CloudExpert: An Intelligent System for Selecting Cloud System Simulators

Alberto Núñez[1], Pablo C. Cañizares[2] and Juan de Lara[2]

[1]*Software Systems and Computation Department, Complutense University of Madrid, Spain*

e-mail: alberto.nunez@pdi.ucm.es

[2]*Computer Science Department, Autonomous University of Madrid, Spain*

e-mail: pablo.cerro@uam.es, juan.delara@uam.es

Abstract

During the last decade, the research community has developed different simulation tools to model and study cloud systems. However, current cloud simulators focus on specific features that typically do not fully cover all aspects of the cloud infrastructure. The ever-growing number of existing simulators increases the difficulty to properly choose the most appropriate one. Moreover, in certain situations, these simulators must be combined to analyze the features required by the user, which leads to investing a considerable time and effort for their selection.

In this paper, we propose CloudExpert, an intelligent system based on metamorphic testing that selects the most appropriate simulator covering the features of interest for the user. In contrast to our previous work, where metamorphic testing is applied to improve models representing a cloud, in this work we analyse the underlying features of several well-known cloud simulators to generate metamorphic rules, which are applied to represent the properties of the simulator. To show the applicability of CloudExpert, we conducted an empirical study where the adequacy of six well-known cloud simulators was analyzed. In this experiment, CloudExpert recommended the most appropriate simulator for eight scenarios involving different aspects of the cloud (energy, storage, network, memory, CPU) and simulator performance; and could also identify strengths and weaknesses of these simulators. Then, we further validated CloudExpert in two different ways. Firstly, the effectiveness of CloudExpert was measured using different faulty cloud simulators. Secondly, we designed a questionnaire based on the results provided by CloudExpert for some of the scenarios of the first experiment. The questionnaire was answered by eight experts in cloud simulation, confirming the usefulness of the tool.

Keywords: Intelligent Systems, Cloud Systems, Cloud Simulators, Simulation, Metamorphic Testing

1. Introduction

The first concepts of cloud computing were proposed in 1997 (Mei et al., 2008). However, the rise of cloud computing started in 2007, when the cloud was adopted and promoted by several top companies, like Amazon (Amazon Elastic Compute Cloud, 2020), IBM (IBM Cloud, 2020), Google (Google Cloud, 2020) and Microsoft (Microsoft Azure, 2020). Since then, billions of dollars have been invested to make available cloud computing solutions. In fact, the market is expected to rise from nearly US$212 billion in 2019 to US$370 billion in 2023 Ng Fred et al (2019).

Cloud computing provides the illusion of infinite storage and computing resources available on demand. This way, a cloud system allows accessing to a *flexible* and *on-demand* computing infrastructure by providing users with virtual machines that are deployed in physical resources. *On-demand* refers to the use of cloud services by end-users according to their computing needs, while *flexibility* concerns the capabilities to offer a wide spectrum of resources using virtualization techniques.

Currently, cloud platforms are increasing their role to perform large-scale computational analysis Keshavarzi et al. (2021); Amani et al. (2020), which has lead both IT companies and the research community to invest much effort for their study. However, testing and analyzing the cloud is especially challenging due to its complex underlying infrastructure. First, cloud systems consist of a large number of physical machines, which are accessed by a vast number of concurrent users, hampering their analysis. Second, the resources provided by the cloud to end users are virtual and, therefore, different VMs can be deployed in a single physical machine, sharing the same resources among the different users. Third, in order to obtain accurate results we need exclusive access to the cloud architecture and to the cloud settings, which is – in most cases – not possible. On the one hand, the main parameters that configure the cloud architecture, like allocation policies of virtual machines, network topologies and the organization of the hosts, are hidden to the end-users. On the other, if we do not have exclusive access to the cloud for executing the experiments, other users may interfere by accessing to shared resources, like the communication network, leading to unreliable results, and making the repeatability of the experiments impossible Schad et al. (2010).

During the last years, simulation techniques have been widely adopted as a cost-effective technique to analyze complex systems Cañizares et al. (2019); Teerasoponpong & Sopadang (2021). Particularly, the use of cloud simulators alleviates the previously described issues, since they are able to easily reproduce and parallelize experiments without requiring the physical hardware resources Filho et al. (2017); Mansouri et al. (2020). However, the wide spectrum of inter-related parameters involved in the modeling of cloud systems requires an expert to design appropriate models for carrying out the experiments. Generally, each simulator focuses on representing an aspect of the cloud, where the behavior of one or several features are simulated, like storage, performance, cost, service-level agreements (SLAs) and energy consumption, among others. Therefore, making the right decision for selecting a simulator is critical to successfully perform an empirical evaluation, avoiding a waste of effort and time Alomair et al. (2015). Conversely, the selection of an inappropriate simulator may lead to significant financial losses and, in some cases, the failure of the research project Gupta (2014).

The problem of selecting an appropriate simulator is not new Hlupic & Mann (1995); Gupta et al. (2009); Nikoukaran et al. (1999). In general, the criteria for evaluating cloud simulators is gathered from related articles found in the current literature, software manuals and by the experience gained when working with existing simulators. This way, the task of evaluating and

selecting a cloud simulator is a multi-criteria decision-making problem, which is complex and time-consuming. In essence, the researcher must analyze the cloud features to be simulated, study the trade-offs of each simulator (e.g. performance vs accuracy, ease of use vs flexibility, etc.) and make a decision. Unfortunately, current approaches for the evaluation and selection of simulation tools cannot be applied to cloud computing simulators. In particular, these solutions are based on analyzing and grouping evaluation criteria, while evaluating a cloud simulator requires to study aspects like the accuracy and performance provided and, thus, more sophisticated techniques are required.

To alleviate these issues, we propose an intelligent system, called `CloudExpert`, which calculates the adequacy of different cloud simulators for a given task. `CloudExpert` receives as input the cloud to be simulated, the workload, and the requirements of the user. These requirements are specified by selecting relevant aspects from a catalogue of metamorphic rules Chen et al. (1998). These include features like the overall performance of the simulator and the capability of the simulator to faithfully capture aspects like the energy consumption, different resource allocation policies and the network system. The output provided by `CloudExpert` is a recommendation of the simulators that best cover the requirements previously indicated by the user.

The main novelty and contributions of this work are the following:

- `CloudExpert` uses a collection of well-known cloud simulators to analyze the features required by the user. The design of `CloudExpert` allows horizontal scaling in the sense that the capacity of the system can be increased by including new simulators.

- The core of `CloudExpert` uses metamorphic rules to model the features of the cloud of interest. Using metamorphic testing (MT) techniques Chen et al. (1998), large and appropriate test suites are automatically generated to evaluate the suitability of each simulator. This way, `CloudExpert` is able to scale vertically by including new metamorphic rules to cover a wider spectrum of cloud features.

- `CloudExpert` automatically generates a high number of test cases targeting the part of the cloud to be studied, which significantly increases the overall system performance. Moreover, the results obtained after the execution of all the test cases are automatically checked using the metamorphic rules. Thus, no human intervention is needed to analyze the accuracy of the obtained results. Metamorphic testing, unlike current conventional testing techniques, can be applied for both test case generation and test result verification, making it suitable to face both fundamental problems of testing Chen et al. (1998); Núñez et al. (2020). In essence, the main difference between traditional testing techniques and MT lies in the comparison of the obtained outputs. While traditional techniques compare the output of each individual test case with the one obtained from the oracle, MT checks the relation between multiple test inputs and their outputs.

- The paper reports on an experimental study analyzing the applicability and suitability of `CloudExpert`. In this experiment we considered six well-known cloud simulators, and `CloudExpert` was able to recommend the most appropriate one for eight scenarios involving different aspects of the cloud (energy, storage, network, memory, CPU) and simulator performance. Moreover, `CloudExpert` could also identify strengths and weaknesses of these simulators.

In this paper, we provide some metamorphic rules to analyze different cloud simulators. While most of these rules are new, those rules targeting at energy consumption are inspired by our own previous work Cañizares et al. (2020). It is worth noting that the application field of this work is completely different. Since our previous work uses metamorphic testing to improve cloud models Cañizares et al. (2020) (i.e. increasing the overall performance or reducing the energetic consumption), the present work analyzes the underlying features of several well-known cloud simulators, not the generated models representing a cloud. Hence, we provide an extensible catalogue of metamorphic rules that are applied to represent the properties of the simulator. Additionally, we use the metamorphic rules to automatically generate test cases and measure the appropriateness of each simulator for modeling a cloud proposed by the user. As we will detail in Sections 4.1, 4.2 and 4.3, new rules can be added to the catalogue and existing ones can be combined. This process is automatic in the sense that the user only needs to enter the cloud to be modeled, while the generation of test cases, its execution and the calculations to provide the final recommendation are performed by our proposed system without the intervention of the user. As a result, `CloudExpert` provides a table containing the recommended simulators to model the cloud provided by the user. From the best of our knowledge, there is no current approach to alleviate this issue, we think that this contribution is new and has not been proposed in the past.

The rest of the paper is structured as follows. Section 2 presents an overview of the three disciplines used to design our system: cloud computing, simulation and metamorphic testing. Section 3 discusses related work, stressing the novelty of our approach with respect to it. Section 4 describes in detail the architecture of `CloudExpert`. Section 5 reports on an empirical study to evaluate `CloudExpert`. The potential threats to validity are described in Section 6. Finally, Section 7 presents the conclusions and some directions for future work.

2. Background

In this section we provide background on the three disciplines used to design `CloudExpert`: cloud computing (Section 2.1), simulation of cloud systems (Section 2.2) and metamorphic testing (Section 2.3).

2.1. Cloud computing

There are several definitions of the notion of cloud computing. The U.S. National Institute of Standards and Technology (NIST) provides a formal definition that includes the key elements used within the cloud computing community (Mell & Grance, 2011): "*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*."

According to their deployment model Mell & Grance (2011), clouds can be categorized as *public*, *private*, *community* and *hybrid*. Public clouds – like for example Amazon EC2 Amazon Elastic Compute Cloud (2020) – are used by the general public. In this scheme, the cloud provider has the full ownership of the cloud, which allows the *pay-as-you-go* pricing model, where users pay only for the capacity their applications actually need. In public clouds, low-level configurations cannot be applied to run experiments and, moreover, the overall cloud performance may present significant variations depending on the physical machines used to deploy the VMs Schad et al. (2010). Private clouds are operated exclusively within a single organization. Usually, public clouds provide bigger infrastructures than private ones. However, IT

companies prefer private clouds when security is a concern. The infrastructure of community clouds is provisioned for exclusive use by a specific community of consumers. Finally, hybrid clouds combine two or more distinct cloud infrastructures (private, community or public) that remain unique entities.

2.2. Simulation of cloud systems

Robert E. Shannon defines simulation Shannon (1998) as "*the process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behavior of the system and/or evaluating various strategies for the operation of the system*". In computer science, simulation is the technique of representing the real world by a computer program, which should imitate the internal processes and not merely the results of the system being simulated. Currently, the research community has widely adopted simulation as a technique for analyzing cloud systems Byrne et al. (2017), due to the following advantages:

- Simulations are *cheap* compared to experimentation using the real system. Simulations can be executed on single regular computers or, if available, in small commodity clusters for increasing the overall performance. Hence, when simulating a cloud system, the underlying physical architecture of the modeled system is not required. Moreover, in the cloud domain, many simulators have a GPL license, which does not require an initial investment.
- Simulation allows a high level of *flexibility* for conducting experiments. The modeled system can be easily customized by modifying configuration parameters (in text files or a GUI). However, experimenting with real cloud systems requires making changes in the hardware, which requires significantly more time and effort.
- Experiments can be easily controlled and repeated in simulated environments.
- Cloud models can be easily scaled, because this just implies increasing the parameters related to the cloud size. On the contrary, scaling a real architecture is expensive and time-consuming.
- Simulators and models can be easily shared with other researchers.

During the last decade, a wide variety of cloud simulators have been proposed. Table 1 compares some features of six well-known, widely used simulators within the cloud research community. They have been selected after a careful analysis of the current literature Byrne et al. (2017); Fakhfakh et al. (2017); Ahmed & Sabyasachi (2014); Ismail (2020). The first three columns show the simulator name, the version used in the empirical evaluation, and its implementation language. In general, Java and C++ are the most adopted languages to build cloud simulators. The next two columns (*License* and *GUI*) depict the software license of the simulators and the support for a graphical interface. All the analyzed simulators provide a *free software* license.

The next four columns analyze the functionality offered by each simulator. The *Net* column determines the capacity of the simulator to model the communication network. In general, a cloud is made of a high number of hosts and communication devices, like switches and routers, leading to long execution times to accurately represent the behavior of the network. In this case, GreenCloud Dzmitry Kliazovich (2012) and iCanCloud Núñez et al. (2012); Castañé et al. (2013) completely model the communication network, implementing the TCP/IP protocol. Other

Simulator	Version	Language	License	GUI	Net	Energy	Storage	Cost	SLA
DISSECT-CF	0.9.3	Java	GPL3	No	Limited	Yes	Yes	No	No
GreenCloud	2.1.2	C++, OTcl	GPL	Yes	Full	Yes	No	No	Yes
SimGrid	3.19.1	C/C++	GPL	No	Limited	Yes	Yes	No	No
iCanCloud	1.0	C++	GPL3/GNU	Yes	Full	Yes*	Yes	Yes	No
CloudSim-Plus	4.3	Java	GPL3	No	Limited	Yes	No	Yes	Yes
CloudSim-Storage	1.0	Java	Apache 2	No	Limited	Storage	Yes	Limited	Yes

Table 1: Comparison of current cloud simulators.

simulators provide a limited model to simulate the network, which is acceptable for simulating cloud systems. The *Energy* column shows the capability of the simulators to model the energetic consumption of the cloud. In general, the presented simulators support this feature. CloudSim-Storage Ouarnoughi et al. (2017) only models the energy consumption for the storage devices. The iCanCloud simulator models the energy consumption using the *E-mc*^{2} framework Castañé et al. (2013), and this is why we set an asterisk on the cell. The capability of each simulator to model the *Storage* system is depicted in the next column. The iCanCloud simulator provides a wide variety of configurations for modeling the storage system, like parallel and distributed file systems and RAID. CloudSim-Storage is an extension of CloudSim Filho et al. (2017) that focuses on modeling the behavior of the storage devices, including its energetic consumption. SimGrid Casanova et al. (2008) also provides models to represent the storage devices in a cloud system. DISSECT-CF Kecskemeti (2015) supports both energy and storage features. Finally, the last two columns represent the capability to model *Cost* and *SLAs*. It is worth mentioning that CloudSim-Plus is the only simulator that supports cost and SLAs.

Overall, we can observe that there is no simulator capable of fully modeling a cloud system. In general, each simulator supports a few features and, consequently, different simulators must be appropriately combined for investigating the different aspects of the cloud infrastructure under study.

2.3. Metamorphic testing

Metamorphic testing (MT) alleviates two fundamental problems in traditional testing: the *reliable test set problem* and the *oracle problem* Weyuker (1982). The first problem consists in generating a proper test suite for determining the correctness of a system. In most cases, it is not feasible to execute all possible test cases over the system under test and, therefore, a subset of test cases has to be selected. However, this task is complex and challenging.

The oracle problem refers to the availability of a mechanism to differentiate the correct behavior of the system under test from potentially wrong behaviors. Schematically, let S be a system, I the input domain and $\mathcal{T} = \{t_1, t_2, \ldots, t_n\} \subseteq I$ a set of tests cases. If these tests are sequentially applied to the system S, we obtain a sequence of outputs $S(t_1)$, $S(t_2)$, ..., $S(t_n)$. Thus, if we have an oracle f to calculate the expected output of S when exercised with any test in $\mathcal{T}$, we find an error if there is some $t_i \in \mathcal{T}$ such that $S(t_i) \neq f(t_i)$. In those cases where the system under test is complex – like the cloud – an oracle is not available or is computationally too expensive to be applied and, consequently, alternative approaches must be used.

Metamorphic testing uses expected properties of the target system, relating multiple test inputs with the outputs produced by the system under test. These properties are formulated in the form of metamorphic rules (MRs). An MR is a property of the studied system that involves multiple inputs and their outputs. We represent an MR as a tuple (MR_i, MR_o), where MR_i is a relation between the source test case and a second (follow-up) test case, and MR_o is a relation

that must be fulfilled by the outputs produced by the source and follow-up test cases. The idea is that, instead of checking the output o_1 produced when testing with one input i_1, we test with a second (follow-up) input i_2, which generates the output o_2. It is important to remark that i_1 and i_2 must fulfill MR_i. Thus, we check that o_1 and o_2 are related as expected by the MR.

As an example, consider the problem of checking a program p that implements the trigonometric sine function. Since it may be challenging to check the correctness of applying p to an arbitrary input, we may take advantage of several known properties of the sine function to build a set of MRs. One of these properties is that $sin(-x) = -sin(x)$. In order to check this MR, we test p with both an input and with the negation of the original input. Next, we check if the MR is fulfilled using the produced outputs. If the property does not hold, p must be faulty and it must have failed on at least one of the two inputs (the original value or its negation).

3. Problem description and related work

As we have discussed, the flexibility and scalability of current simulators, combined with the possibility to easily reproduce experiments, makes simulation tools a very suitable option to study cloud systems. Cloud simulators are widely adopted by different research communities, which are applied, among other fields, for improving efficiency in smart healthcare Abdelmoneem et al. (2020); Anuradha et al. (2020), analyzing telecommunication aspects Hegyi & Varga (2019) and providing the basis for building IoT systems Singh et al. (2020).

Despite the advantages of cloud simulators, selecting the right one requires deep, expert knowledge Nikoukaran et al. (1999) due to the ever-growing number of available options, the effort needed for their set-up, the cost of executing large experiments and the underlying complexity of the tools. This decision must be taken after analyzing a wide spectrum of features and criteria. In general terms, the steps required to select a cloud simulator are the following:

1. Analyzing the features of the cloud system to be studied (e.g. storage, energy consumption, cost). Since the final decision must be taken based on the features to be studied, this step is key to make the right decision.
2. Searching for available cloud simulators, typically over the current literature and specialized web sites.
3. Analyzing the properties, features and capabilities of the simulators found in the previous step. The simulators that do not fulfill the requirements of the researcher are discarded.
4. Modeling the target cloud using the previously collected simulators. Generally, the existing cloud simulators have been developed for different purposes. Moreover, each simulator may require to configure a pre-defined set of parameters – using a specific format – to build a cloud model. Consequently, this step requires *translating* the infrastructure of the cloud under study to the different formats readable by each simulator, which needs significant effort and time.
5. Executing experiments using different simulators. In some cases, the results obtained do not accurately represent the system under study. Thus, it is desirable to analyze not only the characteristics of each simulator by reading the documentation but to execute different simulations to observe if the produced outputs represent the system under test with the desired accuracy.
6. Selecting one or several simulators after a careful analysis of the collected data. Additionally, other aspects that are not directly presented in the provided results should be taken into account, like the trade-offs between performance, flexibility and accuracy.

In general, users may lack the expertise to decide which cloud simulator is more suitable for their requirements. Furthermore, the underlying complexity of cloud systems makes this process challenging and time-consuming. There are recent surveys and studies comparing cloud simulators Márkus et al. (2020); Rahman et al. (2019); Ismail (2020), but the optimal choice depends on the requirements of the particular project. This section reviews several methods and techniques – found in the current literature – that can be applied to evaluate simulation tools.

To the best of our knowledge, there is no expert or intelligent system supporting the evaluation and selection of cloud simulators. Liu et al. Liu et al. (2016) presented a decision-making system for the selection of a cloud vendor. In this case, the underlying infrastructure of cloud simulators are not analyzed, but several vendors of real cloud systems were studied instead. Conversely, several expert systems have been specifically developed to be deployed on the cloud N. Tanković and T. Galinac Grbac and M. Žagar (2017); Škrjanc et al. (2018); Wang et al. (2012).

Evaluation criteria have also been developed with the purpose of evaluating simulation tools. The idea is to categorize the criteria involved in the selection process into different groups, and then provide the user with an organized view of the most relevant features and properties. Law and Haider designed a classification based on a vendor survey of 23 simulation packages, where the criteria were categorized into 6 groups Law & Haider (1989). A classification consisting on 8 groups of features based on a questionnaire involving 54 features was presented by Mackulak et al. Mackulak et al. (1994). Davis and Williams Davis & Williams (1994) designed a set of eight criteria, which consider the main issues involved in the selection of simulation software. Hlupic et al. Hlupic et al. (1999) presented a classification where the user can select between 230 evaluation criteria grouped into different categories. Also, several guidelines of the desirable properties of the simulation tool under study were provided.

Multi-Criteria Decision Making (MCDM) methods are useful when decisions need to be made taking into account multiple conflicting criteria. For instance, the overall performance of a simulator is usually in conflict with the accuracy provided. When a cloud is simulated with a high level of detail – explicitly representing most of its features and properties – the required amount of calculations to perform the simulations is high, hence providing a low performance and high accuracy. On the contrary, when the cloud is simulated using a low level of detail – using an abstraction that represents only its most relevant features – the required calculations to perform the simulations are considerably smaller and faster, which provides a high performance and a lower accuracy. In the current literature, several MCDM methods can be found Attri & Grover (2015), like the Analytic Hierarchy Process Davis & Williams (1994) and the Fuzzy Analytic Hierarchy Process Azadeh & Shirkouhi (2009).

Additionally, several frameworks and tools have been developed for helping researchers to evaluate and select simulation tools. The hierarchical framework of Hlupic et al. Nikoukaran et al. (1999) contains 7 fundamental criteria that are used to evaluate and select the most appropriate simulation software. SimSelect Hlupic & Mann (1995) has a database that stores information concerning the purpose of the simulation and the type, price and name of 20 simulation packages. SimSelect allows users to select and prioritize the desired requirements from a list. If a simulator matching the selected requirements is not found, recommendations to alternative simulators are offered instead. A similar approach, called Smart Sim Selector, helps in the selection of simulation software for the automotive domain Gupta et al. (2009). This tool includes a database containing 11 simulation packages, which are analyzed using 210 different criteria.

Table 2 shows a comparison between the current approaches for selecting simulation tools and our proposed intelligent system `CloudExpert`. In general, these approaches are based on analyzing and grouping evaluation criteria. However, they cannot be applied to selecting a cloud

simulator. First, these approaches do not take into account the *level of detail* used to model the different features of the cloud, which has a direct impact on the resulting simulation accuracy. For example, if a simulator fully models the energy consumption of the CPU system but provides a poor network model, the obtained simulation results may not accurately reflect the behavior of the target system. In order to evaluate this feature, it is required to execute some simulations and study the provided outputs. Nevertheless, some of these approaches group the criteria involved in the selection process to provide the user with the most relevant characteristics of the simulators. This process is carried out in a *static* way in the sense that the studied characteristics are not based on the results provided by the simulator, but on a list previously classified by the expert. Second, *representing the performance* of the simulator in the form of evaluation criteria is complex and in some cases, unpractical. In general, the obtained performance heavily depends on other features, like the level of detail of each modeled feature, the size of the target cloud and the workload to be processed. Some of the current tools for selecting simulators provide a list of features that shows a general view of the performance of the simulators. Unfortunately, this information is not accurate enough to analyze the real performance of cloud simulators. Third, to make a correct decision, the researcher must execute different experiments to evaluate, among other aspects, the *ease of use* for modeling a cloud using each simulator and the accuracy obtained. Thus, for each simulator, different experiments must be designed and executed, which requires a significant amount of effort and time.

Therefore, to fill these gaps, we built `CloudExpert`, which we present in the next section. In essence, `CloudExpert` automatically generates a wide variety of simulations focusing on the features reflected in the metamorphic rules. Moreover, the provided outputs are also automatically checked, which provide accurate information of the analyzed simulators, like the level of detail provided and the performance to execute the generated environments.

Approach	Level of detail	Representing performance	Measuring ease of use
Decision making	No	No	No
Evaluation criteria	Static	No	Static
MCDM	Static	No	Static
Tools	Static	Static	No
`CloudExpert`	Yes	Yes	Yes

Table 2: Comparison between `CloudExpert` and current approaches.

4. Proposed Intelligent System

This section provides a description of our proposed intelligent system, called `CloudExpert`, aimed at evaluating and selecting cloud simulators. For the sake of clarity, we first introduce some notation in Section 4.1. The basic architecture of `CloudExpert` is presented in Section 4.2. Finally, the expert knowledge representation in the form of metamorphic rules is described in Section 4.3.

4.1. Notation

In the first place, `CloudExpert` needs to represent cloud models in a neutral way, so that they can later be translated into the format of each simulator. For this, we have designed a generic template that represents a wide-spectrum of parameters. Thus, `CloudExpert` translates the architecture of the cloud model and the workload stored in a template to the specific format

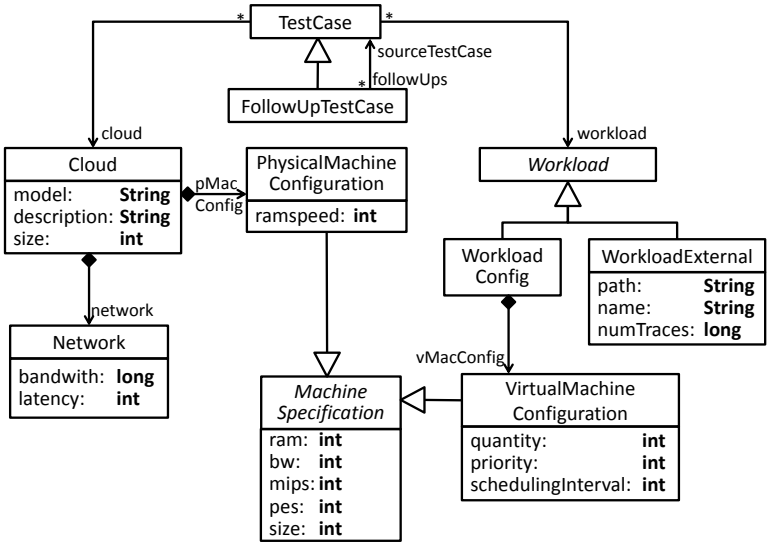


Figure 1: `CloudExpert`'s structure for clouds, workloads and (follow-up) test cases.

```
1  #cloud model
2  cloud.model = cloud_0000
3  cloud.description = 1024 nodes cloud.
4  cloud.size = 1024
5
6  #Hosts
7  host.ram = 16384
8  host.ramspeed = 1600
9  host.bw = 2000000
10 host.sto = 1000000
11 host.mips = 600000
12 host.pes = 8
13
14 #Network
15 bandwidth = 10250000000
16 net.latency = 100
```

Listing 1: Cloud model example in the `CloudExpert` format.

files readable by the target simulator. Figure 1 shows a conceptual class diagram that depicts the common structure of that template to represent cloud models, workloads and test cases.

A cloud is characterized by a *name*, a *description*, its *size* (the number of physical machines), and the main features of the physical machines. These are captured by class *PhysicalMachineConfiguration*, which contains the specification of the the RAM size (*ram*), and its speed (*ramspeed*); the number of cores in the CPUs (*pes*), and their speed (*mips*); the storage size (*size*) and its bandwidth (*bw*).

Listing 1 shows an excerpt of a cloud model in the `CloudExpert`format. Lines 2-3 show the name and description of the cloud model, line 4 refers to the size of the cloud (1024 physical machines), lines 7-12 represent the main features of each physical machine (16GB of RAM, 8-core CPUs at 600k MIPS and 1TB of disk space) and lines 15-16 defines the communication network.

A workload is a set of operations to be executed by the cloud. Generally, each simulator must be configured to process a workload in a specific format and, therefore, one workload

```
1  #VM
2  vm.quantity = 512
3  vm.mips = 250
4  vm.pes = 1
5  vm.ram = 128
6  vm.bw = 100000
7  vm.size = 2500
8  vm.priority = 1
9  vm.schedulingInterval = 300
10
11 Specific:
12 #Workload
13 work.path = none
14 work.name = none
15 work.numTraces = 9000000
```

Listing 2: Workload in the `CloudExpert` format.

must be generated for each simulator. We consider two ways of defining workloads. The first one is resorting to an external file readable by the target simulator (class *WorkloadExternal*) for which we need to specify its *name* and *path*, along with the number of traces. The second one is by configuring the number of VMs (class *VirtualMachineConfiguration*), and the number of instructions – per second – to be executed by each application.

As an example, Listing 2 shows an excerpt of the file generated when the user designs a workload. This template allows the workload to be defined by setting up the corresponding parameters (see lines 2-9) or alternatively by configuring a path containing an external file (see lines 13-15). In this case, the workload is defined by configuring the number of VMs (512) and the number of instructions to be executed by each application (250 MIs). Each VM requires 1-core CPU, 128 MB of RAM and 2.5 GB of disk space to be deployed.

A *test case* is a tuple *(cm, ω)*, which represents the execution of the *workload* ω over the cloud *cm*. Since our system targets cloud simulators, we denote the execution of *workload* ω over the cloud *cm* using the simulator S by $S(cm, \omega)$.

A *follow-up test case* is a test case that is generated by a slight modification of a source test case so that a given metamorphic rule is satisfied. Using this technique, `CloudExpert` generates large test suites – using random testing – to evaluate the different simulators Segura et al. (2011). As an example, Listing 3 shows an excerpt of a test case generated using the cloud model and the workload of Listings 1 and 2. For the sake of clarity, only the most relevant parameters for this example are shown. Let us suppose that we are interested in analyzing the performance of a cloud system when we increase its computing power. For this, `CloudExpert` generates a follow-up test case, denoted by (cm', ω), by applying a slight modification in the CPU system of the source test case, as Listing 4 shows. The test case has been created by increasing the CPU power of *cm*, that is, while the source test cases use 8-core CPUs at 600k MIPS, the follow-up test case uses 16-core CPUs at 700k MIPS. The idea is to execute these test cases and then observe whether the provided outputs are as expected (e.g the cloud *cm'* is faster than the cloud *cm*).

```
1  cloud.size = 1024
2
3  #Hosts
4  host.ram = 16384
5  host.ramspeed = 1600
6  host.bw = 2000000
7  host.sto = 1000000
8  host.mips = 600000
9  host.pes = 8
```

Listing 3: Example of source test case (cm, ω)

```
1  cloud.size = 1024
2
3  #Hosts
4  host.ram = 16384
5  host.ramspeed = 1600
6  host.bw = 2000000
7  host.sto = 1000000
8  host.mips = 700000 ◄
9  host.pes = 16 ◄
```

Listing 4: Example of follow-up test case (cm', ω)

Metamorphic rules (MRs) are designed to model the underlying behavior of the cloud. MRs

are used to generate follow-up test cases, and to check that their outputs and that of the original test cases are related as expected. In the previous paragraph, we used a MR stating that increasing the computer power of one cloud should make it faster. We can express such MR using the notation defined so far: Given two clouds models cm and cm', and a workload ω, if the CPU speed and number of cores of cm' is bigger than those in cm, then the time required to execute ω over cm' must be less than the time required to execute ω over cm. This way, MRs can be defined using the following template:

$$MR(cm, \omega) = \left\{ \left(\begin{array}{l} (cm, \omega), \\ (cm', \omega') \\ S(cm, \omega), \\ S(cm', \omega') \end{array} \right) \middle| \begin{array}{c} MR_i((cm, \omega), (cm', \omega')) \\ \Downarrow \\ MR_o(S(cm, \omega), S(cm', \omega')) \end{array} \right\}$$

where (cm, ω) is the source test case provided by the user, (cm', ω') is a follow-up test case generated by `CloudExpert`, MR_i is a relation over the source test case and a follow-up test case, and MR_o is a relation over the results obtained from the execution of these test cases.

4.2. Architecture of `CloudExpert`

In general, evaluating cloud simulation tools entails two fundamental problems. First, to generate a high number of appropriate test cases. Second, to check if the provided results are correct or not. Typically, the users must manually create the test cases, execute them and check if the results provided properly represent the expected behavior of the system under study. In this process, the user acts both as a tester and as an oracle, requiring expertise in cloud computing and a considerable effort and time to accomplish this task.

Simulating cloud systems is especially challenging due to the complexity of its underlying infrastructure. In order to face the problem of selecting and evaluating cloud simulators, we propose a novel design for an intelligent system that combines cloud simulators and metamorphic testing. `CloudExpert` consists of five main modules, as depicted in Figure 2: a *Rule Engine*, a *Knowledge Base*, a *Testing Engine*, a *Simulation Engine*, and a *Grahpical User Interface* (GUI). `CloudExpert` supports two different user roles: regular and expert users. The former uses `CloudExpert` to find a suitable cloud simulator, and may not have the required expertise to do a proper selection without using `CloudExpert`. As Figure 2 shows, these users only need to provide a cloud model, a workload, and and the aspects of interest for the evaluation (by selecting suitable MRs). Then, `CloudExpert` provides full automation for suggesting the most appropriate simulator for the given requirements. In their turn, expert users provide knowledge to configure `CloudExpert`, for example incorporating new simulators or new metamorphic rules.

Since the *Knowledge Base* (in short, *KB*) contains the knowledge of the expert, it is the key module of the intelligent system. In `CloudExpert`, this knowledge is expressed in the form of MRs. A catalog of MRs is described in Section 4.3 but can be extended with new rules.

The *Rule Engine* (in short, *RE*) implements the reasoning of the expert using the available data from the system to reach a possible solution to the problem. The *RE* receives the data generated from the simulation of a cloud model using different cloud simulators. This data is compared against the different rules from the *Knowledge Base*. As a result, the *RE* calculates the adequacy of each simulator to simulate the features requested by the user, which is reported in the GUI.

The *Simulation Engine* (in short, *SE*) contains a collection of cloud simulators and a generic template for modeling cloud systems, as described in Section 4.1. In general, the parameters to

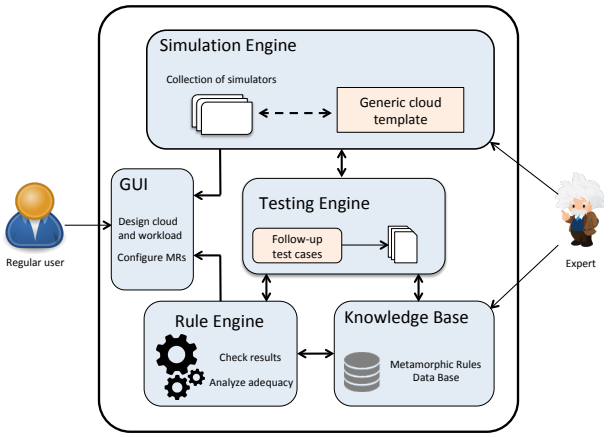


Figure 2: Architecture of `CloudExpert`.

model a cloud differ from the different cloud simulators, which hamper the process of repeating the same experiment using different tools. To alleviate this issue, we provide a generic template where all the parameters used in all the simulators are summarized and presented to the user. This template is required to *translate* the model provided by the user to the different formats readable by the different cloud simulators.

Since each simulator focuses on representing one, or several, aspects of the cloud, like the dynamism for allocating hardware resources, scalability and elasticity, non-experts users can select different simulators – from the *Simulation Engine* – to simulate the required features. Let us remark that there is no current simulator that fully simulates the cloud and, consequently, different simulators must be used to cover the user requirements. In order to analyze the scalability and elasticity of a cloud, it is required to generate several workloads and different configurations of the cloud (i. e. by adding more physical machines to the data-center) and then evaluate how these workloads are processed by these clouds. `CloudExpert` automatically generates the workloads to be processed by the different cloud configurations. Similarly, `CloudExpert` generates different configurations of virtual machines, to evaluate, among other features, how the simulators deal with the dynamic management to allocate resources in the cloud.

The *Testing Engine* (in short, *TE*) is the central piece of `CloudExpert`. It automatically generates and executes – without the intervention of the user – follow-up test cases using metamorphic testing techniques, as described in Section 4.1. In order to generate a *reliable* test suite, `CloudExpert` uses the rules previously selected by the user. The features and properties of the cloud are reflected in the input part of each rule (see Section 4.1), which must be satisfied by the generated test cases. Therefore, the generated test cases aim at the part of the system reflected in the selected rules. This way, not only the search space is reduced by focusing the testing target on specific features, but each test aims at a relevant part of the system previously defined by an expert, hence providing appropriate and quality test cases.

The *GUI* provides a user-friendly interface that aids users to design a cloud system and to select those features of the cloud to be studied. This module has been implemented in Java, which favours portability. Figure 4 shows its main window, which is divided into two frames. The left one shows the current configuration: the source cloud model, the workload and the

loaded MRs. These elements are stored into a repository. This way, users can manage different clouds, workloads and MRs by loading and editing them using the GUI. The right frame of the window shows two log areas. The log area at the top shows a general log indicating the progress of evaluating the adequacy of each simulator, while the log area at the bottom shows the results obtained during the testing process.

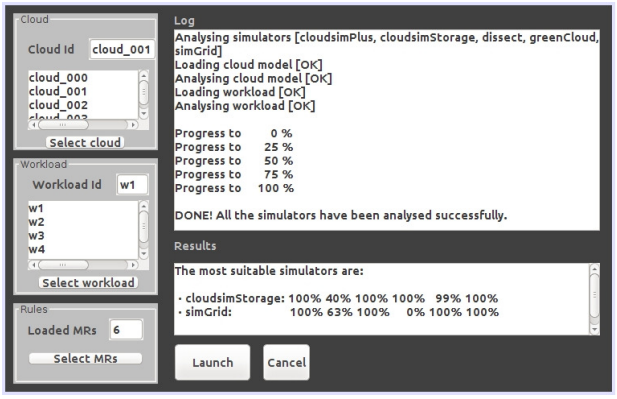


Figure 3: Main menu of `CloudExpert` GUI.

Figure 4 shows the editor of cloud models. In this window, users can provide the required parameters for modeling the underlying cloud architecture. The left area shows a tree containing the currently modeled clouds, which can be loaded into the system to be edited. The right part of the editor shows the required parameters for modeling the cloud. Once the cloud model is configured, a file containing the cloud model is created using the generic template.

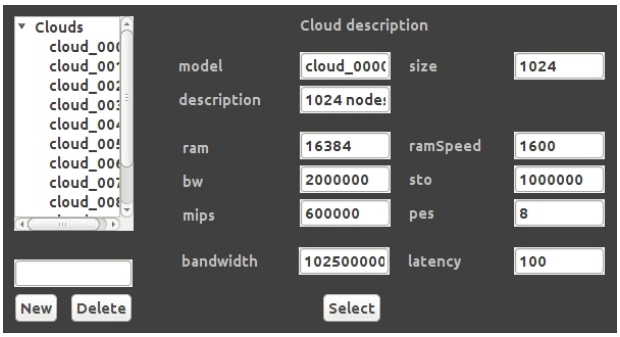


Figure 4: Cloud Editor of `CloudExpert`.

Figure 5 shows a viewer for managing the metamorphic rules in `CloudExpert`. These rules may be (un-)selected to configure the criteria for simulation selection. The set of rules is extensible, since new rules can be created by the expert, but specifying their behavior requires programming in Java.

We next summarize the main steps required to check the adequacy of cloud simulators. Please note that Step 0 is performed by the expert user, and Step 1 by the regular user, but all other steps are automatic.

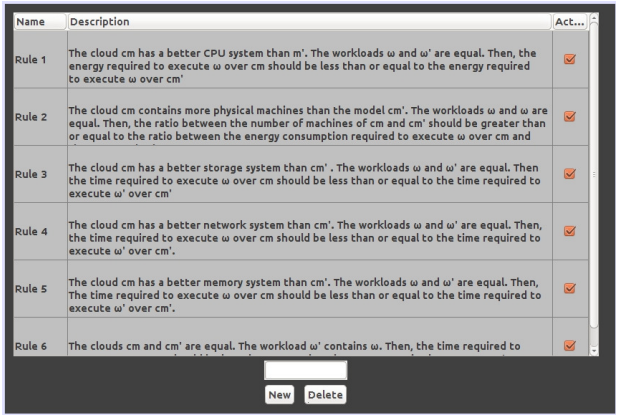


Figure 5: Viewer/chooser of metamorphic rules

Step 0 This is the setting-up phase. Initially, the expert must design a proper catalog of MRs that accurately models the underlying behavior of the cloud, which is stored in the *KB*. Additionally, the expert must deploy a collection of cloud simulators in the *SE*.

Step 1: The regular user interacts with the GUI to design a cloud model and a workload to be processed by the cloud. Once these elements are provided, the GUI displays the list of MRs in the *KB*. The user must select those MRs representing the features that must be evaluated in the simulators, like storage, energy consumption and CPU, among others.

Step 2: The *SE* translates the cloud model provided by the user to the formats readable of the specific simulators. In some cases, it is possible that a simulator is not capable of simulating a specific feature and, consequently, the corresponding MRs representing these features are discarded from the testing process.

Step 3: The *TE* generates, for each selected MR in Step 1, a large number of follow-up test cases. These test cases focus on the features represented in the MRs. Next, these follow-up test cases are executed.

Step 4: The *RE* processes the data generated from the execution of the follow-up test cases. For this, the MRs contained in the *KB* are used to check if the obtained results satisfy the constraints reflected in each rule.

Step 5: Finally, a report containing the adequacy of each simulator is generated. This report is presented to the user through the GUI.

4.3. Metamorphic rules

In this section, we present the catalog of MRs that are currently available within `CloudExpert`'s *KB*. As explained in Section 4.1, MRs use two cloud models and two workloads, denoted by cm, cm', ω and ω' respectively, where cm is the original cloud model designed by the user, ω is the workload – defined by the user – to be processed by cm, and cm' and ω' represent a variant model and workload, respectively, generated automatically by the *Testing Engine*.

Table 3 shows a detailed description of each MR, where the first column refers to the name of the MR and the main cloud aspect tackled (Energy, Storage, Network, Memory and CPU), the second column shows both the input (MR_i) and output (MR_i) relations and the last columns shows a description of each relation. Some of these rules, which target at energy consumption, are inspired by previous work focused on improving the overall energy consumption in cloud systems Cañizares et al. (2020). The rest of the rules are new.

Rule	Constraints	Rule description
MR1 Energy	$MR1_i$	The cloud *cm* has a better CPU system than m'. The workloads ω and ω' are equal.
	$MR1_o$	The energy required to execute ω over *cm* should be less than or equal to the energy required to execute ω' over cm'
MR2 Energy	$MR2_i$	The cloud *cm* contains more physical machines than the model cm'. The workloads ω and ω' are equal.
	$MR2_o$	The ratio between the number of machines of *cm* and cm' should be greater than or equal to the ratio between the energy consumption required to execute ω over *cm* and the one required to execute ω' over cm'.
MR3 Storage	$MR3_i$	The cloud *cm* has a better storage system than m'. The workloads ω and ω' are equal.
	$MR3_o$	The time required to execute ω over *cm* should be less than or equal to the time required to execute ω' over cm'
MR4 Network	$MR4_i$	The cloud *cm* has a better network system than m'. The workloads ω and ω' are equal.
	$MR4_o$	The time required to execute ω over *cm* should be less than or equal to the time required to execute ω' over cm'
MR5 Memory	$MR5_i$	The cloud *cm* has a better memory system than m'. The workloads ω and ω' are equal.
	$MR5_o$	The time required to execute ω over *cm* should be less than or equal to the time required to execute ω' over cm'
MR6 CPU	$MR6_i$	The clouds *cm* and cm' are equal. The workload ω' contains ω.
	$MR6_o$	The time required to execute ω over *cm* should be less than or equal to the energy required to execute ω' over cm'

Table 3: Catalog of Metamorphic Rules

As mentioned in the previous section, the catalog of rules is extensible, to cover more details of existing aspects (energy, storage, network, memory, CPU) or new aspects (e.g., virtualization). Additionally, the existing rules can be combined to create new ones. Let us illustrate this with an example. Let suppose that we have two basic rules, namely r_1 and r_2, where the former establishes that "if the computing power of a cloud system is increased, then the provided performance increases as well", while the latter establishes that "if the network bandwidth increases, the energy consumption is reduced". Then, we can create a new rule, namely r_{12}, which establishes that "if the computing power and the network bandwidth of a cloud increases, then the obtained performance is increased and the energy consumption reduced". Please note that we can combine the input parts of the rule, that is, relations of its underlying features, and the outputs like, in this case, the performance and the energy consumption of the cloud. It is important to remark that an expert with deep knowledge in cloud computing is needed to design proper and valuable rules, as well as to properly combine them.

4.4. Configuring the simulator adequacy score

Assessing the adequacy of a simulator is a complex task that involves considering different inter-related parameters and user requirements. On the one hand, users may be especially interested in the behavior of specific aspects of the cloud under study (i.e., storage, energy consumption and networking) while others might be less relevant for them. On the other hand, the

simulator performance might be an important factor to take into account in the selection. This way, in order to increase the flexibility provided by `CloudExpert`, we allow users to weight the importance of the MRs used, and the period of time that is considered as *acceptable* for executing the simulations.

Regarding the relevance of each MR, let us suppose that a simulator provides accurate results for simulating the network system of a data-center, but the results provided to simulate the energy consumption are not as accurate as expected. In this case, for those users interested in simulating the network of a data-center, the capability of the simulator for representing the energy consumption is useless. Thus, these users may indicate a low – or null – relevance for this feature when calculating the adequacy of the simulator. For this purpose, each MR is assigned a weight, so that the greater is the relevance of the MR, the greater is its weight.

With respect to performance, we can configure how important for the user is the simulator execution time. It is well known that, in simulation, there is usually a correlation between the execution time and the accuracy of the results. Depending of the simulation platform, these values might differ, to a greater or lesser extent.

To cope with these scenarios, we propose an adequacy score, called *simScore*, to measure how suitable is a simulator to represent the properties reflected in the MRs. More concretely, *simScore* uses intermediate results that are calculated by `CloudExpert` using an average of normalized weights to calculate the suitability of each simulator. This score is calculated using Equations 1 and 2. If the user requirements only focus on the capability of the simulators for representing the different properties of the system, *simScore* is calculated using Equation 1 as follows:

$$simScore = \frac{\sum_{i=1}^{n} x_i * w_i}{\sum_{i=1}^{n} w_i} \tag{1}$$

where x_i is to the percentage of test cases that satisfy the *i-th* MR. Therefore, obtaining a value of 100% means that the results provided by the simulator satisfy the MR in all the executed test cases. Additionally, w_i denotes the weight assigned by the user to this MR and n is the number of selected MRs.

When the user considers that the performance of the simulator is relevant, we extend *simScore* with the factor shown in Equation 2:

$$simScore = \frac{\sum_{i=1}^{n} x_i * w_i}{\sum_{i=1}^{n} w_i} * \frac{\frac{1}{max(t,t_{max}))}}{t_{max}} \tag{2}$$

where t is the total execution time of the simulator and t_{max} is the maximum execution time – set by the user – allowed to execute the simulations. The right part of the equation balances the *simScore* value of the simulator depending on its execution time. This way, accurate simulators provide high *simScore* values, but this is decreased in proportion to their efficiency.

In order to clarify the calculation of the *simScore* for a given simulator, we present a synthetic example where two simulators, sim_A and sim_B, are analyzed. In this process, the six MRs presented in Table 3 are used.

We denote by $\mathcal{ES}(R, sim, W, t_{max}) = (X, t)$ the application of `CloudExpert` to calculate the adequacy of a simulator, where $R = (MR_1, \ldots, MR_n)$ is a tuple with the MRs used during the evaluation process, sim is the simulator being evaluated, $W = (w_1, \ldots, w_n)$ is a tuple with the weights assigned to each MR, t_{max} is the maximum time allowed to execute the simulations during the evaluation process, $X = (x_1, \ldots, x_n)$ is a tuple with the percentage of test cases that satisfy each MR_i, and t is the total time required for the simulator to execute all the test cases.

For this example, we use $W = (10, 100, 50, 100, 100, 100)$. First, we apply `CloudExpert` to calculate the adequacy of sim_A obtaining the following results:

$$\mathcal{ES}(R, sim_A, W, 50) = ((100, 100, 100, 100, 100, 100), 200)$$

In this case, sim_A satisfies 100% of the simulated test cases for the six MRs, which means that this simulator is totally capable of simulating all the properties expressed by all MRs. The total time required for simulating all the test cases involved in the evaluation process is 200 time units. Next, we apply our intelligent system to evaluate the adequacy of sim_B.

$$\mathcal{ES}(R, sim_B, W, 50) = ((90, 100, 85, 90, 100, 100), 50)$$

In this case we observe that not all the MRs are completely satisfied by all the test cases. For instance, MR_1 and MR_4 are satisfied by the 90% of the test cases, while 85% of the test cases fulfill MR_3. The total time required for this simulator to simulate the test cases is 50 time units.

Once the intermediate results are calculated, we can differentiate two scenarios. In the first one the user is not interested in the performance of the simulator and, so Equation 1 is used to calculate *simScore*. We obtain a *simScore* of 100 and 96 for evaluating sim_A and sim_B respectively. In this case, `CloudExpert` recommends sim_A over sim_B. For the second scenario, simulator performance is important, and `CloudExpert` uses Equation 2 to calculate *simScore*, which is calculated by using the total execution time required by each simulator and the parameter t_{max} provided by the user. Figure 6 shows the obtained results using different values of t_{max}. These results show that increasing t_{max} has a direct impact on the *simScore* obtained. Moreover, the *simScore* remains the same when $t_{max} = t$.

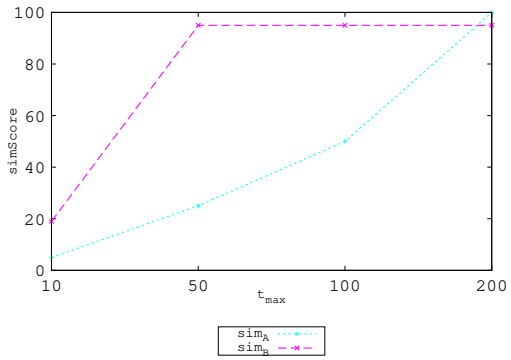


Figure 6: *simScore* using different values of t_{max}

5. Empirical evaluation

In this section we report on an empirical evaluation to check the suitability of `CloudExpert` for measuring the adequacy of six well-known cloud simulators (see Table 1) in different scenarios. This evaluation has been divided in different parts. First, in Section 5.1, we formulate two research questions. Next, Section 5.2 presents the experimental evaluation. Section 5.3 presents a validation process that seeks to verify the obtained results. Finally, the discussion of the obtained results and the answers to the research questions are presented in Section 5.4.

5.1. Research questions

The empirical study described in this section seeks to answer the following two questions:

RQ1 *Is it possible to properly recommend the most appropriate cloud simulator for specific user requirements?*

As mentioned, one of the challenges for modeling and simulating cloud systems lies in the selection of an *appropriate simulator*. In general, researchers invest significant effort and time for installing, configuring and evaluating current simulation tools before performing the experimental process. Moreover, in order to properly selecting the most appropriate simulator, features like performance and accuracy, which can only be studied by executing the simulators and analyzing the provided results, must be considered.

In this work, we are interested in investigating if an intelligent system is capable of properly alleviating this task for the researchers. We face this problem by providing an intelligent system, called `CloudExpert`, which automatically measures the adequacy of different cloud simulators to model and represent cloud scenarios.

RQ2 *Can `CloudExpert` identify the strengths and weaknesses of a cloud simulator?*

There is no cloud simulator that fully simulates each aspect of a cloud computing system. Instead, each cloud simulator focuses on specific parts of the system. The documentation of each simulator and the papers found in the current literature are unfortunately not enough to give a clear view of the accuracy and limitations of each simulator. Hence, in this work we study whether `CloudExpert` is able to identify the main strengths and weaknesses of different cloud simulators.

5.2. Experiments

In this section we presents three experiments carried out to show the applicability, effectiveness and validation of `CloudExpert`.

`CloudExpert` needs to be configured by an expert user. Basically, the expert must design and code the appropriate MRs representing the most relevant features and properties of the cloud, and configure the cloud computing simulators to be analyzed. For this study, we use the six MRs presented in Section 4.3, and use the six cloud simulators in Table 1.

After the setting-up step, non-expert users can use `CloudExpert` to evaluate the different cloud simulators for specific scenarios. In this study, we evaluate `CloudExpert` from the point of view of the non-expert user, which only needs to provide the cloud model to be analyzed, and the workloads. For this study, we manually generate ten different workloads and design a cloud model that consists of 1024 nodes interconnected through a 10-Gigabit Ethernet network. Each node provides a CPU at 32K MIPS, 16 GB of RAM, and a disk drive with a capacity of 1TB.

During the evaluation process, 300 follow-up test cases were automatically synthesized by `CloudExpert` to analyze each simulator, which means that approximately 2100 simulations were executed. The experiment was performed on a desktop PC with an AMD Ryzen 7 1700 CPU at 3,7 GHz, 16 GB of DDR4 memory and a Seagate HHDD with a capacity of 1 TB.

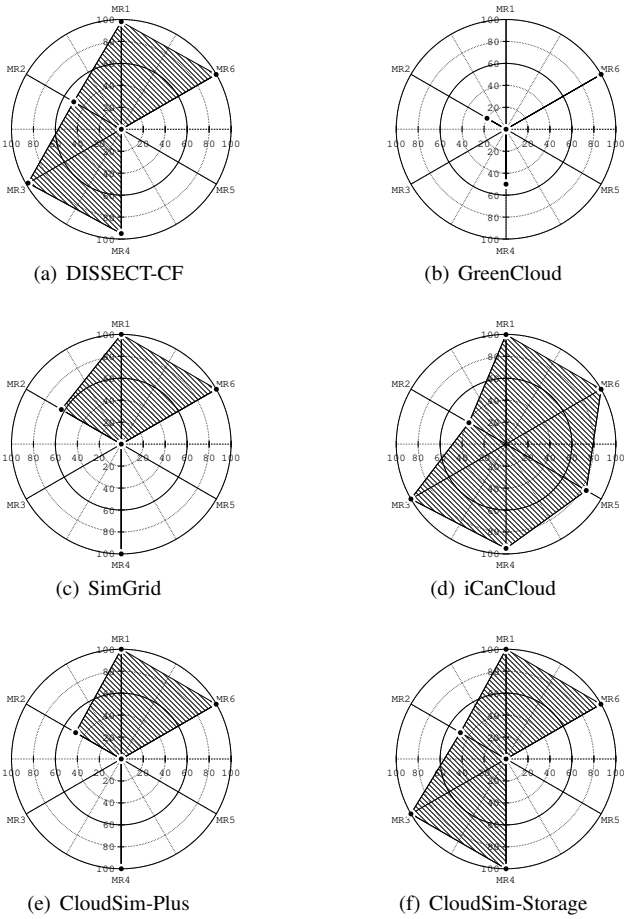
5.2.1. Experiment 1: Preliminary analysis of different cloud simulators

The main objective of this experiment is to render a preliminary view of how `CloudExpert` is applied to check the appropriateness of different cloud simulators to simulate a cloud scenario. For this, we first use `CloudExpert` to calculate the adequacy of each simulator (its *simScore*, see

Section 4.4) when all the MRs are equally involved in the evaluation process (i.e., with a weight of 100) and time restrictions are not applied.

Figure 7 shows the intermediate results provided by `CloudExpert` in this experiment, where a spider chart represents the results of each analyzed simulator. Each chart contains six axes, corresponding to each MR. Each axis ranges from 0 to 100, refereeing to the percentage of test cases that fulfill each MR. Table 4 shows the *simScore* obtained by the simulators using Equation 1.

Since two of the selected features (MR_3 and MR_5 in charts 7.b, 7.c, and 7.e) cannot be represented by *GreenCloud*, *SimGrid* and *CloudSim-Plus*, these simulators obtain poor adequacy scores. Although *iCanCloud* obtains promising results, it only validates fully three MRs (MR_1, MR_3 and MR_6 in chart 7.d). The features reflected in the rest of the MRs provide acceptable results. Although *DISSECT-CF* and *CloudSim-Storage* obtain similar scores, *CloudSim-Storage* fully validates 4 out of 6 features of the cloud, hence obtaining the best score.



(a) DISSECT-CF (b) GreenCloud

(c) SimGrid (d) iCanCloud

(e) CloudSim-Plus (f) CloudSim-Storage

Figure 7: Results of the evaluation process using all the MRs.

DISSECT-CF	GreenCloud	SimGrid	iCanCloud	CloudSim-Storage	CloudSim-Plus
74	27.5	60.5	85.3	73.1	56.5

Table 4: *SimScore* obtained without time restrictions.

5.2.2. Experiment 2: Analyzing the sensitiveness of `CloudExpert` using different configurations

The cloud is a complex system for which many different aspects can be studied. However, different aspects may have different importance for a simulation study. To model each scenario, it is necessary to provide the set of weights for establishing the relevance of the MRs. Let us remind that the greater is the importance of the properties of the rule, the greater is the weight associated with it.

In this second, experiment, we model different scenarios, providing different importance to the energy, storage, network, memory and CPU aspects of a cloud infrastructure. The weight configurations are shown in Table 5. The first configuration (ω_1) uses the same weight for all the MRs. In the second one (ω_2) the weights have been established by selecting the maximum score – from the intermediate results – obtained by all simulators in each MR. Hence, MR_1, MR_3, MR_4 and MR_6, have associated the weight 100, the MR_2 has associated the weight 60 (corresponding with the maximum value of adequacy of this rule, which has been achieved by the *SimGrid* simulator) and MR_5 has associated the weight 80 (corresponding with the maximum value of adequacy achieved by the *iCanCloud* simulator). Finally, the configurations ω_3 to ω_8 have been designed to highlight each specific aspect of the cloud modeled by the MRs.

Weight config	Energy		Storage	Network	Memory	CPU
	MR1	MR2	MR3	MR4	MR5	MR6
ω_1	100	100	100	100	100	100
ω_2	100	60	100	100	80	100
ω_3	100	50	50	50	50	50
ω_4	50	100	50	50	50	50
ω_5	50	50	100	50	50	50
ω_6	50	50	50	100	50	50
ω_7	50	50	50	50	100	50
ω_8	50	50	50	50	50	100

Table 5: Configurations of the weights associated to each MR.

Tables 6-13 present the *simScore* of each simulator by using different t_{max} values and the configurations depicted in Table 5. The *simScore* of each simulator is also represented in Figure 9, where the x-axis of each chart represents the value of t_{max} defined by the user and the y-axis shows the value of the *simScore*. Additionally, a summary of the results is presented in Table 14. We can observe that, using a range of t_{max} between 50 and 200, the most suitable simulator is *CloudSim-Storage* (CS). Then, using a range of t_{max} between 400 and 800, *DISSECT-CF* (D) is considered as the best option. However, if the user does not have time limitations, or the t_{max} parameter can be equal or greater than 1600, the most suitable option is the *iCanCloud* (ICC) simulator.

Figure 8 shows a comparison of the performance provided by the simulators that obtained the best *simCore* in the previous experiment. The x-axis represents the ID of the follow-up test cases executed by each simulator, while the y-axis shows the time, measured in seconds, required to execute the test cases. This chart shows that the fastest simulators are *SimGrid* and *CloudSim-Storage*, providing similar performance. *DISSECT-CF* requires 437 seconds, on average, for executing the simulations. The slowest simulator is *iCanCloud*, requiring 1137 seconds, on average, for executing the simulations. In this particular case, *iCanCloud* provides a full network

Simulator	t_{max} = 50	t_{max} = 100	t_{max} = 200	t_{max} = 400	t_{max} = 800	t_{max} = 1600
DISSECT-CF	9.87	19.73	39.47	**74.00**	**74.00**	74.00
GreenCloud	27.50	27.50	27.50	27.50	27.50	27.50
SimGrid	50.42	60.50	60.50	60.50	60.50	60.50
iCanCloud	3.79	7.59	15.17	30.34	60.68	**85.33**
CloudSim-Storage	**60.97**	**73.17**	**73.17**	73.17	73.17	73.17
CloudSim-Plus	37.67	56.50	56.50	56.50	56.50	56.50

Table 6: *SimScore* obtained by applying the configuration ω_1 and different values of t_{max}.

Simulator	t_{max} = 50	t_{max} = 100	t_{max} = 200	t_{max} = 400	t_{max} = 800	t_{max} = 1600
DISSECT-CF	10.46	20.92	41.84	**78.44**	**78.44**	78.44
GreenCloud	29.07	29.07	29.07	29.07	29.07	29.07
SimGrid	52.13	62.56	62.56	62.56	62.56	62.56
iCanCloud	3.95	7.89	15.79	31.58	63.16	**88.81**
CloudSim-Storage	**65.28**	**78.33**	**78.33**	78.33	78.33	78.33
CloudSim-Plus	39.88	59.81	59.81	59.81	59.81	59.81

Table 7: *SimScore* obtained by applying the configuration ω_2 and different values of t_{max}.

Simulator	t_{max} = 50	t_{max} = 100	t_{max} = 200	t_{max} = 400	t_{max} = 800	t_{max} = 1600
DISSECT-CF	10.34	20.69	41.37	**77.57**	**77.57**	77.57
GreenCloud	23.57	23.57	23.57	23.57	23.57	23.57
SimGrid	55.12	66.14	66.14	66.14	66.14	66.14
iCanCloud	3.89	7.77	15.54	31.09	62.17	**87.43**
CloudSim-Storage	**64.17**	**77.00**	**77.00**	77.00	77.00	77.00
CloudSim-Plus	41.81	62.71	62.71	62.71	62.71	62.71

Table 8: *SimScore* obtained by applying the configuration ω_3 and different values of t_{max}.

Simulator	t_{max} = 50	t_{max} = 100	t_{max} = 200	t_{max} = 400	t_{max} = 800	t_{max} = 1600
DISSECT-CF	9.43	18.86	37.71	**70.71**	**70.71**	70.71
GreenCloud	26.43	26.43	26.43	26.43	26.43	26.43
SimGrid	50.71	60.86	60.86	60.86	60.86	60.86
iCanCloud	3.50	7.00	13.99	27.99	55.97	**78.71**
CloudSim-Storage	**57.02**	**68.43**	**68.43**	68.43	68.43	68.43
CloudSim-Plus	36.10	54.14	54.14	54.14	54.14	54.14

Table 9: *SimScore* obtained by applying the configuration ω_4 and different values of t_{max}.

Simulator	t_{max} = 50	t_{max} = 100	t_{max} = 200	t_{max} = 400	t_{max} = 800	t_{max} = 1600
DISSECT-CF	10.34	20.69	41.37	**77.57**	**77.57**	77.57
GreenCloud	23.57	23.57	23.57	23.57	23.57	23.57
SimGrid	43.21	51.86	51.86	51.86	51.86	51.86
iCanCloud	3.89	7.77	15.54	31.09	62.17	**87.43**
CloudSim-Storage	**64.17**	**77.00**	**77.00**	77.00	77.00	77.00
CloudSim-Plus	32.29	48.43	48.43	48.43	48.43	48.43

Table 10: *SimScore* obtained by applying the configuration ω_5 and different values of t_{max}.

Simulator	t_{max} = 50	t_{max} = 100	t_{max} = 200	t_{max} = 400	t_{max} = 800	t_{max} = 1600
DISSECT-CF	10.27	20.53	41.07	**77.00**	**77.00**	77.00
GreenCloud	30.00	30.00	30.00	30.00	30.00	30.00
SimGrid	55.12	66.14	66.14	66.14	66.14	66.14
iCanCloud	3.82	7.63	15.26	30.53	61.05	**85.86**
CloudSim-Storage	**64.05**	**76.86**	**76.86**	76.86	76.86	76.86
CloudSim-Plus	41.71	62.57	62.57	62.57	62.57	62.57

Table 11: *SimScore* obtained by applying the configuration ω_6 and different values of t_{max}.

Simulator	t_{max} = 50	t_{max} = 100	t_{max} = 200	t_{max} = 400	t_{max} = 800	t_{max} = 1600
DISSECT-CF	8.46	16.91	33.83	**63.43**	**63.43**	63.43
GreenCloud	23.57	23.57	23.57	23.57	23.57	23.57
SimGrid	43.21	51.86	51.86	51.86	51.86	51.86
iCanCloud	3.78	7.57	15.14	30.27	60.55	**85.14**
CloudSim-Storage	**52.26**	**62.71**	**62.71**	62.71	62.71	62.71
CloudSim-Plus	32.29	48.43	48.43	48.43	48.43	48.43

Table 12: *SimScore* obtained by applying the configuration ω_7 and different values of t_{max}.

Simulator	t_{max} = 50	t_{max} = 100	t_{max} = 200	t_{max} = 400	t_{max} = 800	t_{max} = 1600
DISSECT-CF	10.36	20.72	41.45	**77.71**	**77.71**	77.71
GreenCloud	37.86	37.86	37.86	37.86	37.86	37.86
SimGrid	55.12	66.14	66.14	66.14	66.14	66.14
iCanCloud	3.89	7.77	15.54	31.09	62.17	**87.43**
CloudSim-Storage	**64.17**	**77.00**	**77.00**	77.00	77.00	77.00
CloudSim-Plus	41.81	62.71	62.71	62.71	62.71	62.71

Table 13: *SimScore* obtained by applying the configuration ω_8 and different values of t_{max}.

ω/t_{max}	t_{max} = 50	t_{max} = 100	t_{max} = 200	t_{max} = 400	t_{max} = 800	t_{max} = 1600	Selection
ω_1	CS	CS	CS	D	D	ICC	CS
ω_2	CS	CS	CS	D	D	ICC	CS
ω_3	CS	CS	CS	D	D	ICC	CS
ω_4	CS	CS	CS	D	D	ICC	CS
ω_5	CS	CS	CS	D	D	ICC	CS
ω_6	CS	CS	CS	D	D	ICC	CS
ω_7	CS	CS	CS	D	D	ICC	CS
ω_8	CS	CS	CS	D	D	ICC	CS
Selection	CS	CS	CS	D	D	ICC	CS

Table 14: Summary of simulators recommended by `CloudExpert`.

model that requires a vast amount of computing power to be processed.

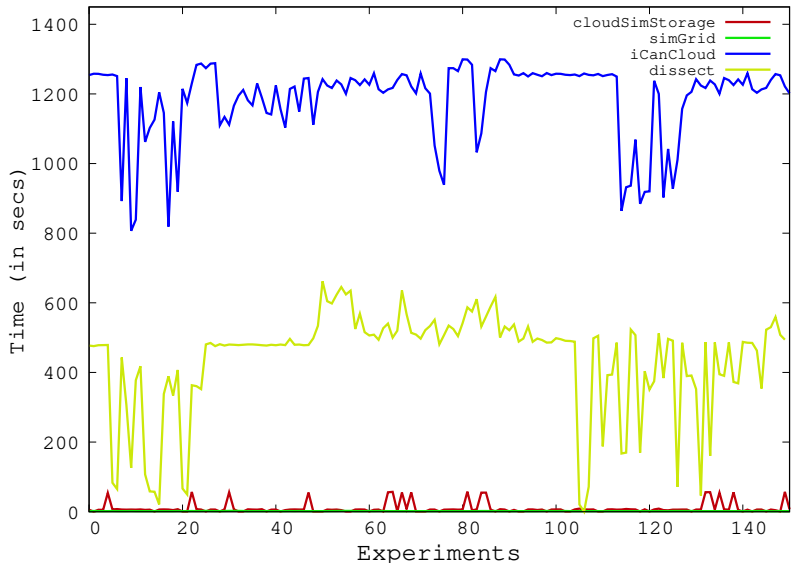
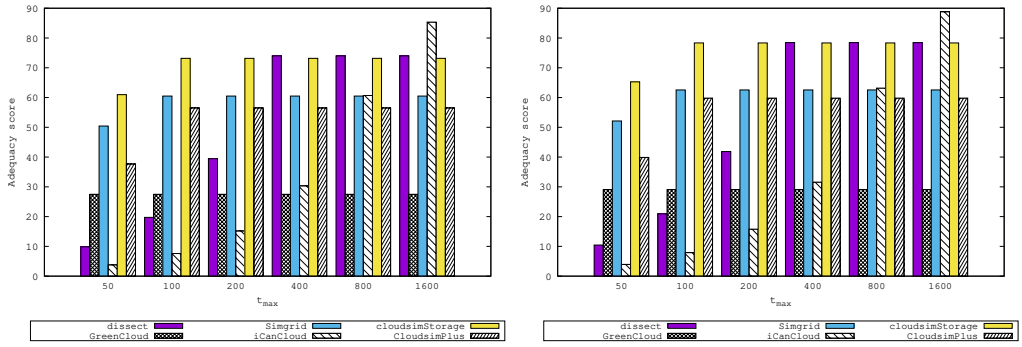
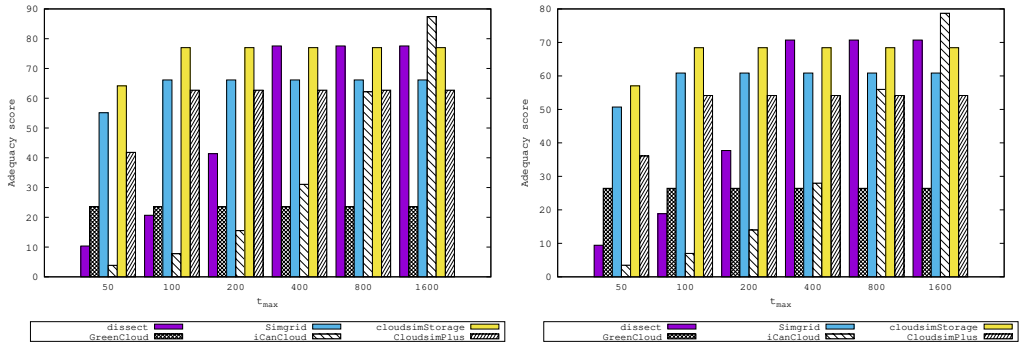


Figure 8: Performance of different simulators analyzed using `CloudExpert`



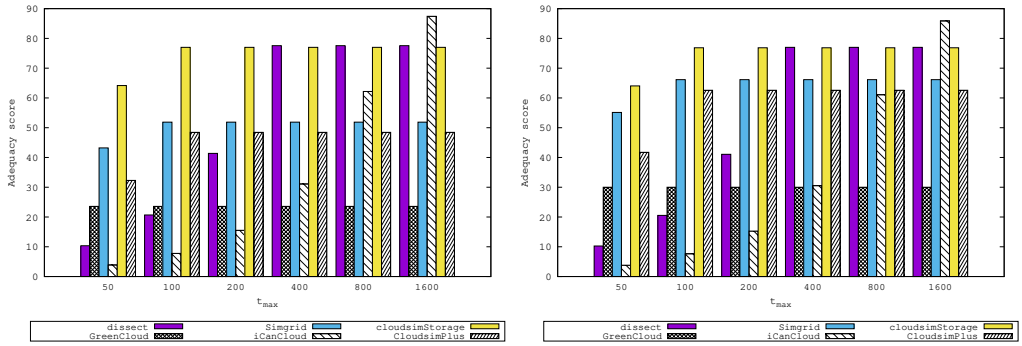
(a) Weights set ω_1

(b) Weights set ω_2



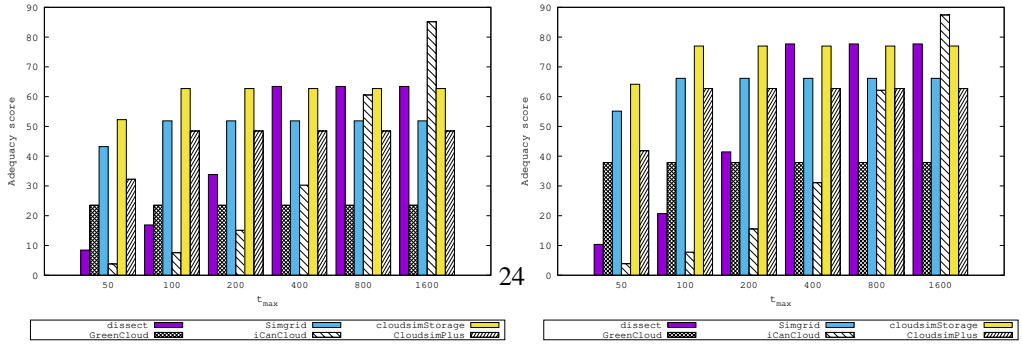
(c) Weights set ω_3

(d) Weights set ω_4



(e) Weights set ω_5

(f) Weights set ω_6



(g) Weights set ω_7

(h) Weights set ω_8

Figure 9: Provided *simScore* by `CloudExpert` to calculate the adequacy of each simulator.

5.2.3. Experiment 3: Checking the suitability of cloud simulators for simulating the different parts of the cloud

In the third experiment, the suitability of the cloud simulators to simulate a specific part of the systems is measured. For this experiment, we have generated one million of configurations, with different combinations of weights (using intervals of 10) to establish the relevance of each MR.

Table 15 presents the number configurations in which `CloudExpert` recommends each simulator for a specific t_{max} value. It can be observed that each simulator is recommended under some configuration. The table shows in bold face the simulators that are recommended more frequently for each t_{max}. Similarly to the results from the previous experiment, we can notice that the simulators recommended more frequently are CloudSim, DISSECT-CF and iCanCloud. Nevertheless, it is important to note that the time restriction for executing the simulation plays an important role in the final decision.

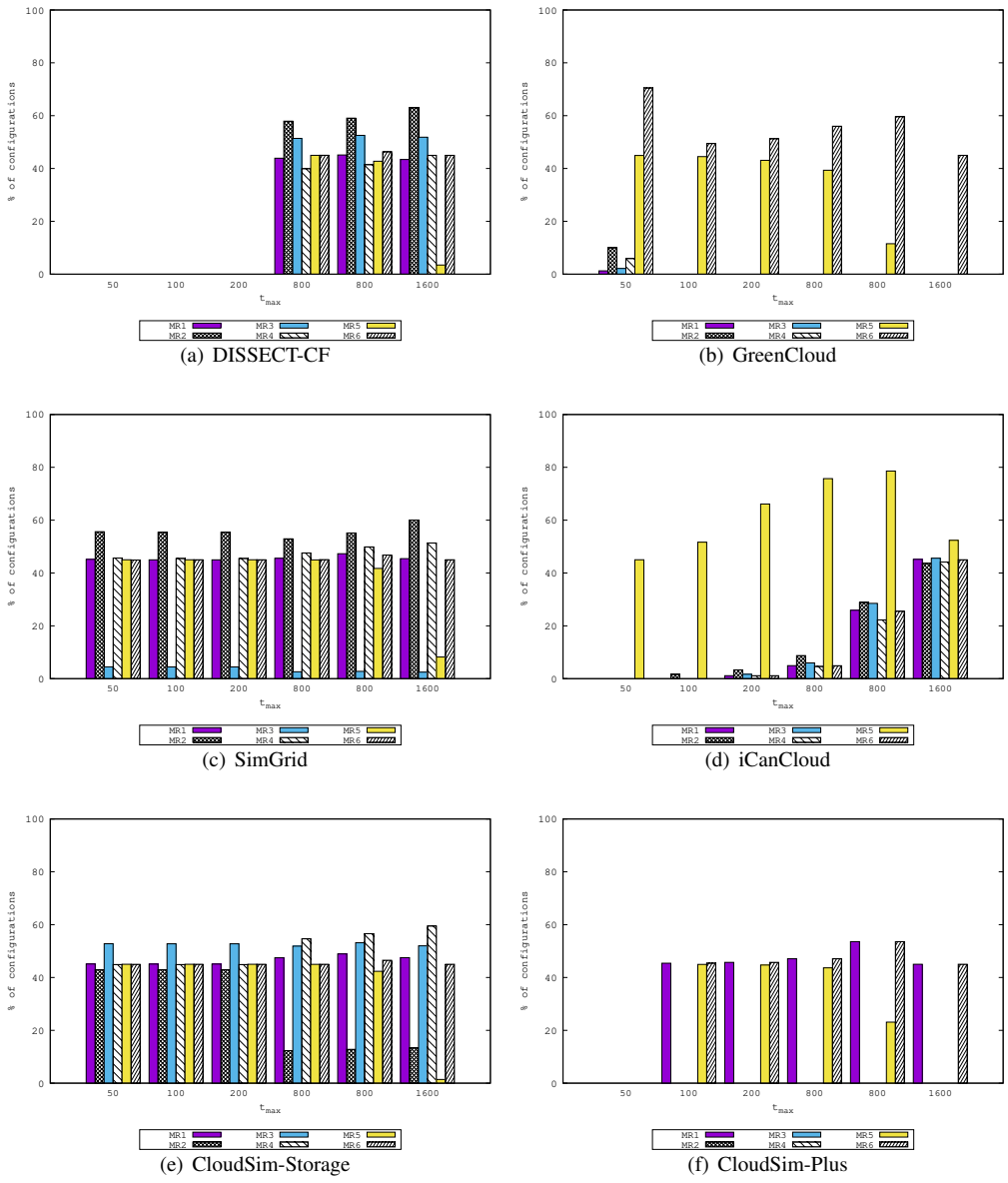
t_{max}/*simulator*	DISSECT-CF	GreenCloud	SimGrid	iCanCloud	CloudSim-Storage	CloudSim-Plus
t_{max} = 50	0	881	161307	10	**839681**	0
t_{max} = 100	0	91	161989	12	**839999**	991
t_{max} = 200	0	87	161971	36	**839985**	983
t_{max} = 400	**599700**	75	135193	368	274668	945
t_{max} = 800	**562856**	26	122741	68323	254637	476
t_{max} = 1600	91790	10	31680	**847748**	31950	100

Table 15: Number of times that `CloudExpert` recommends each simulator

Figure 10 presents the percentage of configurations that makes `CloudExpert` – given an MR – to choose a simulator. Each chart of this figure shows the results for a specific simulator, where the x-axis represents the t_{max} value and the y-axis shows the percentage of configurations where the simulator is recommended. Each chart illustrates the major weakness and strength of each simulator for representing the properties reflected in each MR. For example, we can see that DISSECT-CF is never recommended when the time allowed to execute the simulations is below 800 seconds, since there are other faster simulators. GreenCloud is mostly recommended when MR_5 (memory) and MR_6 (CPU) are of interest. SimGrid is recommended for every t_{max}, but the number of recommendations in which MR_3 (storage) is involved is very low. For low t_{max} values (when performance is a strong requirement), iCanCloud is recommended in configurations only involving MR_5 (memory). However, for higher values of t_{max}, this simulator is recommended in configurations in which each MR is represented. CloudSim-Storage is recommended in configurations with a variety of MR involved, even for low t_{max} values. Interestingly, it is recommended more times than SimGrid when MR_3 (storage) is involved. For higher values of t_{max}, we can see that the number of recommendations involving MR_2 (energy) and MR_5 (memory) decreases. Finally, CloudSim-Plus is never recommended for configurations involving MR_2 (energy of physical machines), MR_3 (storage) or MR_4 (network).

5.3. Validation process

This section presents a study to validate the results obtained from the previous experiments, which consists of two different parts. In section 5.3.1 we show some experiments where the effectiveness of `CloudExpert` is measured using different faulty cloud simulators. Next, in Section 5.3.2, we present some data – acquired from human experts in the field of cloud simulation – to verify the results provided by `CloudExpert`.



(a) DISSECT-CF (b) GreenCloud

(c) SimGrid (d) iCanCloud

(e) CloudSim-Storage (f) CloudSim-Plus

Figure 10: % of configurations that makes `CloudExpert` recommending a simulator for each MR.

5.3.1. Measuring the effectiveness of `CloudExpert` using faulty version of cloud simulators

In order to check the effectiveness of `CloudExpert`, we have repeated some experiments using faulty simulators. Since the faulty simulators are clearly not a suitable choice for accurately and efficiently simulating cloud systems, we expect that these simulators are not considered as

a relevant recommendation made by `CloudExpert`. Thus, we have seeded different faults in DISSECT-CF and CloudSim-Storage. These faults are known errors – discovered in previous versions of these simulators – that have been extracted from the official gitHub repositories. In particular, DISSECT-CF have been seeded with three errors: *allows reducing the number of calls to the actual consolidation* method[1], *minor performance improvement to the simple consolidator*[2], and *VM scheduling refactoring and minor performance improvements*[3]. It is important to mention that these errors only affect the performance of the simulator, that is, the output provided by the faulty version is the same than the one provided by the simulator used in the previous experiments. Overall, the time required to finish the execution of the simulations is increased – approximately – by a 20%. We have repeated the same experiment described in Section 5.2.3 using the faulty version of DISSECT-CF, where we analyze the number of times that `CloudExpert` recommends a simulator to simulate the cloud provided by the user. These results are shown in Table 16. We observe that both versions obtain the same results when $t_{max} \leq 200$, which means that this simulator is not recommended because the simulations require more than 200 seconds to be executed. However, when $t_{max} = 400$, the number of times that `CloudExpert` recommends this simulator drastically drop – when the faulty version is used – from 599,700 to 409. However, when the constraints to provide a recommendation are relaxed, that is, when $t_{max} \geq 800$, the results obtained are the same for both versions. In this case, although the faulty version is 20% slower, the requirements of the user allows that this drop of performance does not affect to the final recommendation.

simulator	$t_{max} = 50$	$t_{max} = 100$	$t_{max} = 200$	$t_{max} = 400$	$t_{max} = 800$	$t_{max} = 1600$
DISSECT-CF 0.9.3	0	0	0	599700	562856	91790
DISSECT-CF 0.9.3 with faults	0	0	0	**409**	562856	91790

Table 16: Number of times that `CloudExpert` recommends each simulator

Similarly, CloudSim-storage have been seeded with several minor bugs[4]. In this case, the injected bugs affect the functional behavior of the simulator in the sense that the output provided by both versions of the simulator – CloudSim-Storage 1.0 and CloudSim-Storage 1.0 with errors – differs. In order to compare these versions of the CloudSim-Storage, we have repeated the experiment described in Section 5.2.1. Table 17 shows the number of test cases that satisfies each MR using CloudSim-Storage and a faulty version of this simulator. In this case, we observe that the faulty version of ClousSim-Storage provides worse results in *MR1*, *MR2* and *MR4*, which reflect properties related to energy and network. It is important to remark that a wrong output indicates a loss in the accuracy of the simulator and, consequently, `CloudExpert` does not consider these cases as a suitable choice for simulating cloud systems.

Simulator	*MR1*	*MR2*	*MR3*	*MR4*	*MR5*	*MR6*
CloudSim-Storage 1.0	100	40	100	99	0	100
CloudSim-Storage 1.0 with faults	**91**	**39**	100	**79**	0	100

Table 17: Percentage of test cases that satisfies each MR using different version of cloudSim-storage

[1] https://github.com/kecskemeti/dissect-cf/commit/a4ad13982f80b2333304797bd7db1659e0b8ade6
[2] https://github.com/kecskemeti/dissect-cf/commit/d45fde1733a15907eb1ea4885997f7ae81f8490e
[3] https://github.com/kecskemeti/dissect-cf/commit/8d6beb541bd2b1599ca841bbb3b930ca4a057093
[4] https://github.com/Houarnoughi/CloudSimStorage/commit/f6c8c52c8efb962bbbf68065055781d59a454264

5.3.2. Validation results obtained from the knowledge of human experts in cloud simulation

To verify the results provided by `CloudExpert`, we have gathered data from eight users of cloud simulators using an online survey. The expertise level of these participants ranges from regular users with basic knowledge to experts with deep knowledge in cloud simulation. Let us remark that some of these participants are developers of the most widely adopted cloud simulators by the research community, like CloudSim, CloudSim-Plus and SimGrid. We designed the survey to collect data in four stages:

- Stage 1: This stage consists in obtaining data related to the background of each participant. In essence, this data is related to demographic data, including field of expertise, area of expertise (academic or industry), years of experience in the field of cloud simulation, expertise level in this field and participation level – if it is the case – in the development of a cloud simulator.

- Stage 2: In this stage, our aim is to understand the knowledge of each participant on a set of simulators. More precisely, we are interested in analyzing the participants' capability to select the most appropriate cloud simulator to model specific features of the cloud. Let us remark that no additional data – to aid the participant – was provided at this stage. For this, we ask the participants if they know about the six well-known cloud simulators used in this work (see Table 1). Additionally, we ask for the cloud features supported by each cloud simulator without specifying its version and, next, we ask the same indicating a specific version of each simulator. Finally, we present a cloud scenario and ask to each participant which is the most appropriate cloud simulator to simulate such scenario.

- Stage 3: In this stage, the recommendations calculated by `CloudExpert` are facilitated to aid the participants for answering questions, which are the same than the ones presented in the previous stage. This way, he/she may use – or not – this information to respond the questions.

- Stage 4: In this final stage, the participants must evaluate the recommendations made by `CloudExpert` to simulate specific cloud scenarios.

In summary, most of the participants of this study have been involved in the development of a cloud simulation tool, and have more than five years of expertise in this field. However, it is important to remark that none of the participants know in depth all the presented cloud simulation platforms. In fact, there is no simulator that is known – in depth – by all the experts, despite in the current literature some of these tools reach 1,000 cites, like GreenCloud (>1000) and SimGrid (> 2000). The rest of the simulators have a significant number of cites, like Dissect-cf (>100), iCanCloud (>400), CloudSim-Storage (>25) and CloudSim-Plus (>50).

After a careful analysis based on the data submitted by the participants, we can draw the following conclusions.

First (and obviously) the participants that have been involved in the development of cloud simulators show a high knowledge in those simulators. In general, the participants agree with the recommendations provided by `CloudExpert`. However, there are slight variations with regard to the suggestion made by `CloudExpert`, when the participants select the version of the simulator to represent specific cloud features, these being clarified by an e-mail sent by the expert, which denotes that this work is interesting for them. On the contrary, the participants show a superficial knowledge in the rest of the simulators. This way, we believe a system like `CloudExpert` can

help experts to choose the best simulator – which is perhaps out of their knowledge – for a given scenario.

Second, in those situations where the participant has a limited knowledge of the cloud features that can be represented by a cloud simulator, `CloudExpert` aided to chose a proper version of the simulator.

Third, we have detected a significant number of mistakes when participants had to select a version of the simulator to represent specific cloud features. This reinforces the view that a manual selection of a simulator to solve a specific scenario – even by an expert – can be erroneous, and a system like `CloudExpert` can help in this task.

Fourth, the final evaluation of the recommendations made by `CloudExpert` has been satisfactory. It is worth mentioning that some of the participants are highly qualified experts in this field, who have developed some of the simulators used in this work, hence providing valuable feedback information. Finally, in general, the participants were satisfied with their final recommendations.

5.4. Discussion of the results and answers to the research questions

In this section we discuss the obtained results and answer the research questions formulated in Section 5.1.

RQ1: *Is it possible to properly recommend the most appropriate cloud simulator for specific user requirements?*

In order to answer this question, we have carefully analyzed the results obtained in the experiments presented in Section 5.2.1 and 5.2.2. These experiments have been designed to check the adequacy of `CloudExpert` for recommending the most appropriate simulator for simulating cloud scenarios. Additionally, we have performed a validation study in Section 5.3 to certify the results obtained in the previous experiments.

In these experiments, several scenarios have been modeled by providing different importance to the energy, storage, network, memory and CPU aspects of the cloud. We observe that `CloudExpert` generates different recommendations depending of the user requirements. In summary, taking into account the adequacy calculated by `CloudExpert` and the performance of each simulator, *CloudSim-Storage* appears as the most appropriate simulator to tackle the aspects of the cloud modeled by the six MRs (see Table 14), which also provides a good trade-off between accuracy and performance.

To verify these results, we have performed a validation study where users with different expertise levels in the field of cloud simulation evaluate the recommendations made by `CloudExpert`. In general, we can conclude that the participants are satisfied by the suggestions provided by `CloudExpert`. Additionally, this study has been completed with an experiment where faulty versions of the simulators are used. The idea is to observe whether the final recommendation, made by `CloudExpert`, varies when using a less suitable simulator (caused by a real failure). In this case, we show that `CloudExpert` does not consider the faulty versions of the simulators as a relevant recommendation.

Hence, we can conclude that the answer to **RQ1** is yes, *it possible to automatically recommend a cloud simulator for specific user requirements.* In essence, `CloudExpert` uses both the accuracy of each simulator and the time required to execute the simulation to provide a recommendation. Hence, in those cases where the time for finishing the simulation is limited, `CloudExpert` clearly recommends CloudSim-Storage, which provides a good accuracy/performance ratio. However, when the time constraint is less restrictive, `CloudExpert` recommends

DISSECT-CF as the most appropriate simulator. In those cases where the time is not a relevant restriction, iCanCloud seems to be the best choice, providing high accuracy at the cost of requiring a long execution time for providing the results.

RQ2 *Can* `CloudExpert` *identify the strengths and weaknesses of a cloud simulator?*

In order to answer this question, we have carefully analyzed the results obtained from the third experiment presented in Section 5.2.3.

The results presented in Table 15 and Figure 10 show that DISSECT-CF, SimGrid and CloudSim-Storage provide good results for simulating the major part of the properties reflected in the MRs, but DISSECT-CF is not recommended when the time allowed to execute the simulations is below 800 seconds. On the contrary, when the execution time restriction is relaxed, the memory system (MR_5) is not accurately represented. SimGrid and CloudSim-Storage provide similar results. The main difference between these simulators lies in their weakness. While the main weakness of SimGrid is the storage system (MR_3), a noticeable weakness of CloudSim-Storage is the energy consumption (MR_2). The rest of the simulators provide acceptable results for a reduced number of MRs. GreenCloud shows good results – independently of time restrictions – for simulating the memory system and the CPU. The major strength of iCanCloud is clearly the memory system. However, this simulator requires a long execution time to generate the results, and only when performance is not an issue it is recommended for other aspects of the cloud. CloudSim-Plus provides acceptable results for simulating the energy consumption of CPUs (MR_1) independently of the performance required.

Overall, we can conclude that the answer to **RQ2** is yes, `CloudExpert` *is able to automatically identify the strength and weakness of different cloud simulators*.

6. Threats to validity

In this section, we discuss the threats to validity of our empirical study.

6.1. Internal threats

Internal validity concerns whether our findings, which are based on the obtained results from the empirical study, truly represent a cause-and-effect relationship. Thus, the internal validity of our study lies in the implementation of our experiments.

The design and implementation of `CloudExpert` has been performed by three experts and experienced programmers. In order to provide a high level of flexibility, allowing the execution of `CloudExpert` in a wide spectrum of platforms, the system has been implemented in Java. The source code has been thoroughly examined and tested – by hand – to check the correctness of the implementations.

The MRs for representing the behavior of cloud systems have been designed by two experts in this field. These rules are based on a previous work focusing on improving the overall energy consumption in cloud systems Cañizares et al. (2020). We are aware that it is possible to obtain different results if other MRs are used instead. Nevertheless, it is important to remark the relevance of the expert, who is key for accurately and appropriately design the MRs used in `CloudExpert`.

In order to verify the obtained results, we have performed a validation study where different participants – experts and regular users – show how appropriate are the commendations made by `CloudExpert` to represent a cloud scenario. In some cases – at stage 2 – where the participants do not have access to the suggestions provided by `CloudExpert`, the recommendations provided

by CloudExpert do not match with the ones provided by the participants. However, this result is modified – at stage 3 – when the participants have access to such information, which denotes that, when the users do not have a clear choice, CloudExpert provides valuable information that can be used by the participant to improve the final answer.

Additional issues affecting our findings might arise due to the cloud simulators used in the empirical study. We mitigate this threat by using six-well known simulators widely adopted by the community and, thus, we expect these simulators are correct and do not contain important errors.

6.2. External threats

External validity concerns the extent to which the results of a study can be generalized.

We have designed – by hand – eight different configurations, covering a wide-range of user requirements to analyze the adequacy of six well-known cloud simulators. We think that these configurations are representative. However, we also generated – automatically – one million configurations to mitigate this issue, carefully analyzing the number of cases where CloudExpert recommends each simulator. These results are summarized in Table 15.

We have used a cloud configuration that reflects a wide range of features to be represented by the simulators, like performance, energy consumption, and the capability to represent hardware resources. Although we believe that this model is representative, we cannot guarantee that the obtained results are the same for all the possible scenarios.

6.3. Constructs threats

Construct validity concerns whether the used measures are representative or not.

The adequacy of each simulator is calculated by using two measures widely adopted in the community, accuracy and performance. Although it is possible that the simulators used in the empirical study contain defects, we control this threat by executing a large number of test cases.

7. Conclusions and future work

In this paper we have presented CloudExpert, an intelligent system for evaluating and selecting cloud simulators. The system has a novel design that manages a collection of cloud simulators and uses metamorphic techniques to automatically generate test cases and to check if the provided outputs represent the expected behavior of the target system. The design of CloudExpert allows vertical scaling, by including new cloud simulators into the system, and horizontal scaling, by adding new rules for accurately modeling the underlying cloud infrastructure.

In order to check the applicability of CloudExpert we have carried out an empirical study where six well-known cloud simulators have been evaluated using six MRs. After a careful analysis of the obtained results, we can summarize the following achievements that have been reached with CloudExpert. First, the carried out experiments show promising results, since CloudExpert recommends the most appropriate cloud simulator by not only analyzing the user requirements for simulating the target cloud architecture, but also taking into account the performance of each simulator for simulating a wide range of cloud configurations. Second, CloudExpert is able to automatically identify the weakness and strengths of each simulator. In those cases where the evaluation is focused on a reduced number of features, CloudExpert is able to quickly make a decision to select the most appropriate simulator. However, in those cases

where the number of features to be evaluated increases, CloudExpert requires to generate a high number of follow-up test cases, requiring a considerable CPU power to completely execute the evaluation process. Moreover, other aspects should be taken into account to make the final decision, like the performance of each simulator, which is calculated by CloudExpert, and the difficulty to model a cloud, where the intervention of the user is required. Finally, we show that the flexible and scalable design of CloudExpert allows including new cloud simulators into the system, widening the range of possible clouds for being studied.

Additionally, it's important to remark the main limitations that have been identified during the experimental analysis. CloudExpert is written in Java using a single-thread design. Consequently, CloudExpert is not able to be executed in parallel, hence using the available CPU cores of the computer where it is executed and, therefore, all the simulations are executed sequentially. Furthermore, we would like to remark that the role of the expert is crucial to accurately evaluate the cloud simulators, since the designed MRs must be carefully designed and coded into the system. Otherwise, a wrong design of the MRs may lead to incoherent results.

We think that CloudExpert provides a significant contribution to the scientific community. Although CloudExpert is targeted at researchers dealing with cloud simulators, our approach is specially useful for non-experts users that analyze and study cloud systems using simulation tools. First, CloudExpert allows users to focus on those simulators that cover the features of interest, discarding the rest and, consequently, saving time and effort. Second, CloudExpert is able to automatically generate a high number of test cases, providing relevant information of the cloud to the user. Third, CloudExpert automatically translates the cloud model provided by the user to the format of each simulator, keeping the user unaware of the underlying configuration of each simulator, which in most cases consist of large plain-text files difficult to understand.

As future work, we plan to include new cloud simulators and to automatically generate workloads adapted to the user needs.

In order to increase the performance of CloudExpert, we also plan to add support for multi-thread executions. Hence, simulations would be launched in parallel, thus exploiting the parallelism of the CPU cores provided by the computer – or virtual machine – where CloudExpert is executed. Furthermore, CloudExpert can be easily executed in the VMs provided by current clouds providers, like Amazon (Amazon Elastic Compute Cloud, 2020), Google (Google Cloud, 2020) and Microsoft (Microsoft Azure, 2020). There are only two requirements for successfully executing CloudExpert in a VM. First, the VM must provide a recent version of the JRE (Java Runtime Environment). Second, the VM must provide an Operating System compatible with the simulators deployed by CloudExpert. In summary, a VM containing Linux and a JRE would be enough to execute CloudExpert. We plan to deploy CloudExpert on some of these clouds to increase its performance, and make it accessible as a service.

Moreover, we are currently working on combining MRs to evaluate, with greater precision, the adequacy of each simulator. For this, we are evaluating the possibility of using artificial intelligence techniques for using a guided search that allows the system to provide reliable results in a reduced time period. Finally we are also considering the extension of these techniques to fog computing simulators Márkus & Kertész (2020).

Acknowledgments

This work has been supported by the Spanish MINECO-FEDER projects FAME (grant number RTI2018-093608-B-C31) and MASSIVE (grant number RTI2018-095255-B-I00), the Re-

gion of Madrid project FORTE-CM (grant number S2018/TCS-4314) and the Madrid Government (Comunidad de Madrid-Spain) under the Multiannual Agreement with the Complutense University as part of the Program to Stimulate Research for Young Doctors in the context of the V PRICIT (Regional Programme of Research and Technological Innovation) under grant PR65/19-22452.

References

References

Abdelmoneem, R., Benslimane, A., & Shaaban, E. (2020). Mobility-aware task scheduling in cloud-fog IoT-Based healthcare architectures. *Computer Networks*, (p. 107348).

Ahmed, A., & Sabyasachi, A. S. (2014). Cloud computing simulators: A detailed survey and future direction. In *IEEE International Advance Computing Conference (IACC'14)* (pp. 866–872).

Alomair, Y., Ahmad, I., & Alghamdi, A. (2015). A Review of Evaluation Methods and Techniques for Simulation Packages. *Procedia Computer Science*, *62*, 249 – 256. Proceedings of the 2015 International Conference on Soft Computing and Software Engineering (SCSE'15).

Amani, M., Ghorbanian, A., Ahmadi, S. A., Kakooei, M., & et al., A. M. (2020). Google earth engine cloud computing platform for remote sensing big data applications: A comprehensive review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, (pp. 5326–5350).

Amazon Elastic Compute Cloud (2020). Web page at `http://aws.amazon.com/ec2/`. Accessed: 2020-12-23.

Anuradha, M., Jayasankar, T., Prakash, N., Sikkandar, M., Hemalakshmi, G., Bharatiraja, C., & Britto, A. (2020). IoT enabled cancer prediction system to enhance the authentication and security using cloud computing. *Microprocessors and Microsystems*, (p. 103301).

Attri, R., & Grover, S. (2015). Application of preference selection index method for decision making over the design stage of production system life cycle. *Journal of King Saud University - Engineering Sciences*, *27*, 207 – 216.

Azadeh, M. A., & Shirkouhi, S. N. (2009). Evaluating simulation software using fuzzy analytical hierarchy process. In *Proceedings of the 2009 Spring Simulation Multiconference (SpringSim'09)* (pp. 41:1–41:9).

Byrne, J., Svorobej, S., Giannoutakis, K., Tzovaras, D., Byrne, P., Östberg, P., Gourinovitch, A., & Lynn, T. (2017). A review of cloud computing simulation platforms and related environments. In *7th International Conference on Cloud Computing and Services Science (CLOSER'17)* (pp. 651–663).

Cañizares, P. C., Núñez, A., & de Lara, J. (2019). An expert system for checking the correctness of memory systems using simulation and metamorphic testing. *Expert Systems with Applications*, *132*, 44–62.

Cañizares, P. C., Núñez, A., de Lara, J., & Llana, L. (2020). MT-EA4Cloud: A Methodology For testing and optimising energy-aware cloud systems. *Journal of Systems and Software*, *163*, 110522.

Casanova, H., Legrand, A., & Quinson, M. (2008). SimGrid: A generic framework for large-scale distributed experiments. In *10th Int. Conf. on Computer Modeling and Simulation, UKSIM' 08* (pp. 126–131).

Castañé, G., Núñez, A., Llopis, P., & Carretero, J. (2013). E-mc^2: A formal framework for energy modelling in cloud computing. *Simulation Modelling Practice and Theory*, *39*, 56–75.

Chen, T. Y., Cheung, S. C., & Yiu, S. M. (1998). *Metamorphic testing: a new approach for generating next test cases*. Technical Report HKUST-CS98-01.

Davis, L., & Williams, G. (1994). Evaluating and selecting simulation software using the analytic hierarchy process. *Integrated Manufacturing Systems*, *5*, 23–32.

Dzmitry Kliazovich, S. U. K., Pascal Bouvry (2012). GreenCloud: A packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, *62*, 1263–1283.

Fakhfakh, F., Kacem, H. H., & Kacem, A. H. (2017). Simulation tools for cloud computing: A survey and comparative study. In *IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS'17)* (pp. 221–226).

Filho, M. C. S., Oliveira, R. L., Monteiro, C. C., Inácio, P. R. M., & Freire, M. M. (2017). Cloudsim plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *IFIP/IEEE Symposium on Integrated Network and Service Management (IM'17)* (pp. 400–406).

Google Cloud (2020). Web page at `https://cloud.google.com/`. Accessed: 2020-12-23.

Gupta, A. (2014). How to select a simulation software. *International Journal of Engineering Research and Development*, *10*, 35 – 41.

Gupta, A., Verma, R., & Singh, K. (2009). Smart sim selector: A software for simulation software selection. *International Journal of Engineering*, *3*, 175 – 185.

Hegyi, P., & Varga, J. (2019). Telco cloud simulator. In *24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD'19)* (pp. 1–7). IEEE.

Hlupic, V., Irani, Z., & Paul, R. J. (1999). Evaluation framework for simulation software. *The International Journal of Advanced Manufacturing Technology*, *15*, 366–382.

Hlupic, V., & Mann, A. S. (1995). Simselect: a system for simulation software selection. In *Winter Simulation Conference Proceedings.* (pp. 720–727).

IBM Cloud (2020). Web page at `https://www.ibm.com/cloud`. Accessed: 2020-12-23.

Ismail, A. (2020). Energy-driven cloud simulation: existing surveys, simulation supports, impacts and challenges. *Clust. Comput.*, *23*, 3039–3055.

Kecskemeti, G. (2015). DISSECT-CF: A simulator to foster energy-aware scheduling in infrastructure clouds. *Simulation Modelling Practice and Theory*, *58*, 188–218. Special issue on Cloud Simulation.

Keshavarzi, A., Haghighat, A. T., & Bohlouli, M. (2021). Clustering of large scale QoS time series data in federated clouds using improved variable Chromosome Length Genetic Algorithm (CQGA). *Expert Systems with Applications*, *164*, 113840.

Law, A. M., & Haider, S. W. (1989). Selecting simulation software for manufacturing applications: Practical guidelines & software survey. *Industrial Engineering*, *21*, 33–46.

Liu, S., Chan, F. T., & Ran, W. (2016). Decision making for the selection of cloud vendor: An improved approach under group decision-making with integrated weights and objective/subjective attributes. *Expert Systems with Applications*, *55*, 37 – 47.

Mackulak, G., Savory, P. A., & Cochran, J. (1994). Ascertaining important features for industrial simulation environments. *SIMULATION*, *63*.

Mansouri, N., Ghafari, R., & Zade, B. M. H. (2020). Cloud computing simulators: A comprehensive review. *Simulation Modelling Practice and Theory*, *104*, 102144.

Márkus, A., Gacsi, P., & Kertész, A. (2020). Develop or dissipate fogs? evaluating an iot application in fog and cloud simulations. In *Proceedings 10th International Conference on Cloud Computing and Services Science, CLOSER* (pp. 193–203). SCITEPRESS.

Márkus, A., & Kertész, A. (2020). A survey and taxonomy of simulation environments modelling fog computing. *Simul. Model. Pract. Theory*, *101*, 102042.

Mei, L., Chan, W. K., & Tse, T. H. (2008). A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues. In *3rd IEEE Asia-Pacific Services computing conference, (APSCC'08)* (pp. 464–469). IEEE Computer Society.

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing. NIST Special Publication 800-145, September 2011.

Microsoft Azure (2020). Web page at `https://azure.microsoft.com/`. Accessed: 2020-12-23.

N. Tanković and T. Galinac Grbac and M. Žagar (2017). ElaClo: A framework for optimizing software application topology in the cloud environment. *Expert Systems with Applications*, *90*, 62 – 86.

Ng Fred et al (2019). Forecast: Public Cloud Services, Worldwide, 2017-2023, 1Q19 Update. `https://www.gartner.com/en/documents/3906899`. Accessed: 2020-12-17.

Nikoukaran, J., Hlupic, V., & Paul, R. J. (1999). A hierarchical framework for evaluating simulation software. *Simulation Practice and Theory*, *7*, 219 – 231.

Núñez, A., Cañizares, P. C., Núñez, M., & Hierons, R. M. (2020). Tea-cloud: A formal framework for testing cloud computing systems. *IEEE Transactions on Reliability*, (pp. 1–24).

Núñez, A., Vázquez-Poletti, J. L., Caminero, A. C., Castañé, G. G., Carretero, J., & Llorente, I. M. (2012). iCanCloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing*, *10*, 185–209.

Ouarnoughi, H., Boukhobza, J., Singhoff, F., & Rubini, S. (2017). Integrating I/Os in Cloudsim for Performance and Energy Estimation. *ACM SIGOPS Operating Systems Review*, *50*, 27–36.

Rahman, U. U., Bilal, K., Erbad, A., Khalid, O., & Khan, S. U. (2019). Nutshell - simulation toolkit for modeling data center networks and cloud computing. *IEEE Access*, *7*, 19922–19942.

Schad, J., Dittrich, J., & Quiané-Ruiz, J. (2010). Runtime measurements in the cloud: Observing, analyzing, and reducing variance. *PVLDB*, *3*, 460–471.

Segura, S., Hierons, R. M., Benavides, D., & Ruiz-Cortes, A. (2011). Automated metamorphic testing on the analyses of feature models. *Information and Software Technology*, *53*, 245 – 258.

Shannon, R. E. (1998). Introduction to the art and science of simulation. In *Proceedings of the 30th conference on Winter simulation (WSC'98)* (pp. 7–14). Los Alamitos, CA, USA: IEEE Computer Society Press.

Singh, S., Chana, I., & Buyya, R. (2020). Agri-Info: Cloud based autonomic system for delivering agriculture as a service. *Internet of Things*, *9*, 100131.

Teerasoponpong, S., & Sopadang, A. (2021). A simulation-optimization approach for adaptive manufacturing capacity planning in small and medium-sized enterprises. *Expert Systems with Applications*, *168*, 114451.

Wang, W., Zeng, G., Tang, D., & Yao, J. (2012). Cloud-dls: Dynamic trusted scheduling for cloud computing. *Expert Systems with Applications*, *39*, 2321 – 2329.

Weyuker, E. J. (1982). On testing non-testable programs. *The Computer Journal*, *25*, 465–470.

Škrjanc, I., Andonovski, G., Ledezma, A., Sipele, O., Iglesias, J. A., & Sanchis, A. (2018). Evolving cloud-based system

for the recognition of drivers' actions. *Expert Systems with Applications*, *99*, 231 – 238.