Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

# Simcan2Cloud: a discrete-event-based simulator for modelling and simulating cloud computing infrastructures

Pablo C. Cañizares[1*], Alberto Núñez[2], Adrián Bernal[3], M. Emilia Cambronero[3] and Adam Barker[4]

**Abstract**

Cloud computing is an evolving paradigm whose adoption has been increasing over the last few years. This fact has led to the growth of the cloud computing market, together with fierce competition for the leading market share, with an increase in the number of cloud service providers. Novel techniques are continuously being proposed to increase the cloud service provider's profitability. However, only those techniques that are proven not to hinder the service agreements are considered for production clouds. Analysing the expected behaviour and performance of the cloud infrastructure is challenging, as the repeatability and reproducibility of experiments on these systems are made difficult by the large number of users concurrently accessing the infrastructure. To this, must be added the complications of using different provisioning policies, managing several workloads, and applying different resource configurations. Therefore, in order to alleviate these issues, we present Simcan2Cloud, a discrete-event-based simulator for modelling and simulating cloud computing environments. Simcan2Cloud focuses on modelling and simulating the behaviour of the cloud provider with a high level of detail, where both the cloud infrastructure and the interactions of the users with the cloud are integrated in the simulated scenarios. For this purpose, Simcan2Cloud supports different resource allocation policies, service level agreements (SLAs), and an intuitive and complete API for including new management policies. Finally, a thorough experimental study to measure the suitability and applicability of Simcan2Cloud, using both real-world traces and synthetic workloads, is presented.

**Keywords** Cloud computing, Simulation, SLAs, Pricing schemes

## Introduction

Over the last few years, cloud computing has become a reference for on-demand computing. The high level of flexibility, security, and cost savings have led companies to use this computing paradigm for the provision of the services they require. According to the Right-Scale 2019 State of the Cloud Report from [1], 94% of enterprises use at least one cloud service, and spending on such services reached $227.8 billion. In order to satisfy this demand, there exist several cloud service providers, such as Amazon Web Services (AWS), Azure, Google Cloud, VMWare Cloud, and Oracle Cloud Infrastructure, among others.

Market competition has led service providers to seek elements of differentiation, such as performance, quality of service, and cost. Thus, one of the main goals of cloud providers is to achieve a good balance between system performance and usage of computational resources while maintaining profits. However, achieving a balanced

*Correspondence:
Pablo C. Cañizares
pablo.cerro@uam.es
[1] Computer Science Department, Autonomous University of Madrid, Madrid, Spain
[2] Software Systems and Computation Department, Complutense University of Madrid, Madrid, Spain
[3] Department of Computer Science, Universidad de Castilla-La Mancha, Albacete, Spain
[4] School of Computer Science, University of St. Andrews, St. Andrews, Scotland

architecture that accomplishes this goal is challenging. Considering an emerging company that provides cloud services, considerable growth in the number of users accessing the services may lead to experiencing system bottlenecks, which may force profit drops due to the loss of users. The main idea is to provide the data-centre with adequate computational resources to serve the incoming users, avoiding overdimensioning, or underdimensioning, the system.

In order to obtain a good cost-performance ratio it is necessary to perform a thorough analysis of the cloud when processing different workloads, which allows the provider to properly configure the different cloud parameters, such as virtual machines (VMs), resource allocation policies, and the cost of each VM offered [2]. A misconfigured cloud environment may lead to poor overall performance, which will have a significant impact on the quality of service and, consequently, compromise the reputation of the company.

Unfortunately, carrying out an experimental analysis on production-ready environments is complex, expensive and, in some cases, not possible due to the necessity of having dedicated access to the system. Furthermore, applying configuration changes in a production system, such as adding more machines, replacing computational resources, or setting a new network topology, may affect the behaviour of the system.

In the last ten years, researchers have tackled these issues by using simulation techniques [3–6]. The main features of these techniques allow the creation of simulation tools that are appropriate for modelling, analysing and studying complex systems. In essence, a simulator uses an abstraction of the system under study - namely a model - to imitate its behaviour by representing its most relevant features. Among the most important advantages provided by simulation, we can highlight the following cloud-systems-related ones: (i) The system under study is not required to execute the simulations. In general, simulators can be run on a regular computer; (ii) Experiments can be easily reproduced in a simulated environment. In most cases, there exist a high number of inter-related parameters and variables that cannot be controlled on a real-world production system, such as the users accessing the system concurrently, thus making the repeatability of the experiments impossible [7]. However, simulation allows us to reproduce the same experiment in a controllable way; (iii) Experiments can be run in parallel, improving performance without requiring specific hardware resources [8, 9]. Thus, simulations can be run on a standard desktop – using the available CPU cores – or, in order to significantly increase the number of simulations executed in parallel, on a computer cluster; and (iv) Simulation provides more flexibility when applying changes to the configuration settings. While modifying the configuration of a cloud system is a time-consuming and expensive task, simulation only requires us to modify the configuration of the model by setting up the correct parameters, such as the network topology, or the resource allocation policy.

Currently, there exists a broad spectrum of simulation platforms for modelling cloud computing systems. However, most of the cloud simulators are focused on representing the behaviour of the system from the users' perspective, and do not consider the cloud provider part. For instance, DISSECT-CF [10] is considered as one of the most relevant cloud computing simulators. However, different aspects related to the cloud provider, such as allocation policies, user management, and costs are not taken into consideration. Additionally, there exist several proposals focused on different cloud provider aspects, such as pricing features [11–13], cloud deployments [14], modelling resources [15], and services offered by the cloud provider [16]. Nevertheless, these works are not targeted at considering the underlying hardware of a cloud platform.

To the best of our knowledge, there are few simulation platforms aimed at describing the cloud provider, with a reasonable level of detail, while considering the infrastructure support. In these terms, CloudSim [8] offers several policies for the management of the available cloud resources, supporting different host selection strategies, service deployment, and VM provisioning. However, the resources of the cloud infrastructure, and both the management and the behaviour of the users, are not particularly detailed. In order to overcome these issues, we present Simcan2Cloud, a discrete-event-based framework for modelling and simulating cloud systems. Simcan2Cloud mainly focuses on the cloud provider, supporting the modelling of cloud infrastructures and the interaction of the users with the cloud. In addition, for analysing how Simcan2Cloud is aligned with the real world, the platform includes a trace representation module that allows to execute real-world traces collected from production-ready systems. Thus, we can compare the simulated system with the real – target – system to find potential inconsistencies. Below, we highlight the most relevant and novel features of our proposed simulation platform:

1. Flexible SLAs. Simcan2Cloud considers different SLA definitions in cloud computing environments. Hence, the requested resources are allocated to the users according to the different parameters established in the SLA: availability of the resources, rental time, and a configurable cost model that covers several aspects, such as discounts for delays, an extra

cost for additional time, and compensation for unavailability.

2. A cloud provider waiting queue. In terms of user management, the platform provides a queue system to handle users upon their arrival in the cloud. This mechanism enables users to wait for the requested service – by subscribing to the system – instead of leaving the system immediately.
3. Priority users. In order to enrich the behaviour of the system, the platform supports the management of users with different priority levels. Hence, high-priority users are not required to wait in the cloud provider queue, since their requested resources are allocated on reserved machines, which are exclusively dedicated to these users.
4. A renting time extension offer. With respect to the service rentals, the platform supports extending the rental time of the VMs when some services are still running, and the rental time of the requested resources has expired. This feature is designed to cover a common behaviour in cloud environments.
5. Resource usage. This platform includes a module for monitoring the usage of the computational resources at the data-centre, such as CPUs, RAM memory and storage. This feature enables users to analyse usage patterns and detect disruptive behaviours in these key subsystems.
6. An API to easily include new management policies. Simcan2Cloud supports different scheduling policies for resource allocation. The cloud provider can select the most appropriate algorithms for maximising both the percentage of resource usage and the cloud provider profits. In addition, the platform provides templates to facilitate the creation of custom scheduling policies and user behaviours.

This paper is organised as follows. Firstly, Section "Related work" introduces and analyses the state of the art of cloud computing simulators. Section "Simcan2Cloud" presents the architecture and the implementation details of Simcan2Cloud. Then, we present an empirical study in Section "Empirical study", in which the performance of Simcan2Cloud is analysed and discussed. Finally, Section "Conclusions and future work" contains our conclusions and some lines for future work.

## Related work

In the last few decades, simulation techniques have been adopted by the research community as a valuable way to study and analyse cloud computing environments. As a result, a significant number of cloud computing simulators have appeared in the literature [6, 17–21]. The noticeable growth in the state-of-the-art surveys – from an average of 10 in 2012 [21] to up to 30 in 2020 [6] – is a clear indicator of the increasing interest in designing cloud simulators.

### Cloud computing simulators

In the current literature, we found several simulation platforms focused on the cloud provider. The CloudNetSim++ simulator [22] is a cloud simulator, built on OMNeT++, that uses the INET Framework to model a complete network layer. This simulator allows users to describe SLA policies, scheduling algorithms, and billing costs, and offers the built-in OMNeT++ user interface. Thus, users must learn the basics of the OMNeT++ environment to create cloud scenarios. Another proposal is the Data Centre Simulator (DCSim), a simulation framework for modelling and simulating data centre infrastructures [23]. In general terms, DCSim focuses on the IaaS layer, which is used for providing services to multiple clients. It is also worth mentioning that DCSim supports the modelling of cost and SLAs. DISSECT-CF is a simulation platform focused on modelling resource sharing and the cloud infrastructure with a high level of detail [10]. This approach presents quite a detailed IaaS stack simulation and supports energy-aware techniques for cloud infrastructures, hence allowing the inclusion of new metrics for analysing different resources. SCORE [24] is a simulator based on Google Omega and written in Scala. SCORE simulates parallel scheduling, energy consumption, and synthetic workloads, as well as offering shutting-down and powering-on computational node mechanisms. In the same line, SCORE-GAME is an extension of SCORE that includes an energy scheduling policy based on the Stackelberg game [25]. The model of this simulator includes two roles, namely the *Scheduling Manager* and the *Energy-Efficiency Manager*. The former processes the tasks as quickly as possible, while the latter is targeted at minimising the overall energy consumption. In this way, this proposal is based on a competition between those roles, where the main goal is to balance the trade-offs between energy consumption and performance. iCanCloud is a simulation platform built on the OMNeT++ framework [26]. In essence, this simulator represents the behaviour of cloud systems by modelling the physical machines supporting the cloud, the configuration of the VMs provided and different resource allocation policies. Additionally, the E-mc$^2$ framework [9] has been developed to include support for measuring the energy consumption of the different hardware components of the system, such as the memories, the CPUs of disk drives, etc. Thus, iCanCloud can be used to estimate the trade-offs between cost and performance in a wide range of cloud scenarios.

The surveys of cloud simulation tools found in the current literature claim that CloudSim [8] is one of the leading cloud simulators [6]. CloudSim uses SimJava as the simulation core and allows the modelling of hosts in data centres, virtual machines, user tasks, and resource provisioning policies. CloudSim focuses on service broker scheduling algorithms and implements space-shared and time-shared allocation policies to manage computing resources. This simulator provides a limited network model, as it only considers transmission delays and lacks a realistic network topology. Since several researchers found limitations in carrying out experiments with CloudSim in areas of study, the research community has extended the capabilities of the tool by implementing other simulators based on CloudSim. Among such simulators extending CloudSim functionalities, we can highlight NetworkCloudSim, CloudAnalyst, CDOSim, WokflowSim, CloudExp, and UCloud.

NetworkCloudSim [27] improves the network layer of CloudSim by implementing switches at several aggregation levels and providing communication models with different levels of detail. These implementations allow developers to model parallel applications. CloudAnalyst [28] provides a GUI for CloudSim, presenting geographical factors that allow the configuration of user and data centre locations. Basically, the location feature enables the simulator to calculate the response and processing time of the requests. CDOSim [29] simulates the cost and performance characteristics of cloud deployment scenarios, and allows developers to model delays and SLA violations, helping them to choose a deployment strategy. Although this simulator implements VM migration, it still inherits a limited network model from CloudSim.

WokflowSim [30] introduces the modelling of scientific workflows in a cloud environment and job clustering, which allows researchers to study the impact of job failures on workflows. This simulator is not suitable for data-intensive applications, since it does not model the performance of the storage system. The simulator CloudExp [31] improves CloudSim by including complex network models and a Map-Reduce processing model. CloudExp offers SLA definition based on measurable terms, and also supports a workload generator toolkit to model real workloads. One of the main weaknesses of CloudExp is the static model for measuring the performance of the VMs. UCloud [32] is a hybrid cloud simulator – for university environments – focusing on scenarios that require the services of public clouds when the private cloud is full. In addition, UCloud implements performance monitoring, university activities, and security management, as well as considering the cost of using the public cloud, but not the cost of the data centre communications.

## Comparison of Simcan2Cloud and SoTA solutions

In this section, we present a comprehensive comparison between Simcan2Cloud and some of the well-known cloud simulators. It is important to highlight the effort and time invested by the research community to create and maintain a broad spectrum of simulation tools, a fact that has led to the existence of a wide range of cloud simulators. In order to choose those simulators that have been widely adopted by the community, we have carefully analysed papers available in the current literature, surveys – such as those of [6] and [17] – and public repositories.

Table 1 analyses the main differences between the existing cloud simulators and the approach presented in this work, namely Simcan2Cloud, by highlighting the main contribution of our proposal. The table consists of five sections aggregating several aspects of the cloud simulation platforms. The first section (labelled *Main features*) provides basic information about each platform, such as the name of the tool, creation date (*Year*), programming language (*Lang.*), availability of the tool (*Avail.*) and the framework used as a basis for creating the simulator (*Platform*). The second block, labelled *General aspects*, shows general features of the simulation platform, that is, the possibility of using a graphical user interface (*GUI*), the level of detail to represent the communication network (*Comm. model*), and the capabilities of the platform for designing the network topology (*Network topology*). The third block, called *Cloud provider*, shows features for modelling the behaviour of the cloud provider. In this case, we analyse the service level agreement (*SLA*), the cost model (*Cost*), and the renting time extension offer (*Rent ext.*), which refers to those scenarios in which some services are still running when the renting time of the requested resources ends. In this particular case, the users have the possibility of extending the renting time of the resources by paying an extra charge on top of the initial cost. Additionally, this block covers scheduling policies (*Sched. policies*), and the management of user queues (*Waiting queue*), which refers to the mechanism that allows users to wait if the requested resources are not available. The fourth block (*Users/workload*) presents the features for modelling a cloud environment from the perspective of the user that accesses the cloud to request services. In this case, we consider the facilities for managing new workloads (*API*), support for representing the execution of traces extracted from real-world clouds (*Real traces*), the capability to use different statistical distributions to create workloads (*Traffic dist.*). This feature represent the resources requested by the users by indicating

**Table 1** Comparison of the main cloud computing simulators existing in the literature

| Main features | | | | | General aspects | | | Cloud provider | | | | | Users (workload) | | | | DC | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Tool** | **Year** | **Lang.** | **Avail.** | **Platform** | **GUI** | **Comm. model** | **Network topology** | **SLA** | **Cost** | **Rent ext.** | **Sched. policies** | **Waiting queue** | **API** | **Real traces** | **Traffic dist.** | **Prior users** | **HW usage** | **HW detail** |
| CloudSim | 2009 | Java | Open Source | SimJava | ✗ | Limited | Limited | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | ✗ |
| NetworkCloud-Sim | 2009 | Java | Open Source | CloudSim | ✗ | Limited | Limited | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | Limited |
| CloudAnalyst | 2010 | Java | Open Source | CloudSim | ✓ | Limited | Limited | Limited | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | Limited |
| iCanCloud | 2011 | C++ | Open Source | OMNeT++ | ✓ | Limited | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| DCSim | 2011 | Java | Open Source | - | ✗ | ✗ | ✗ | Limited | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| GreenCloud | 2012 | C++ Otcl | Open Source | NS2 | Limited | Full | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ |
| WokflowSim | 2012 | Java | Open Source | CloudSim | ✗ | Limited | Limited | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | Limited |
| CloudNetSim++ | 2014 | C++ | Open Source | OMNeT++ | ✓ | Limited | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ | ✗* (VM) | ✓ | ✗ |
| CloudExp | 2014 | Java | N.A | CloudSim | ✓ | Full | Limited | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | Limited |
| UCloud | 2014 | Java | Open Source | CloudSim | ✗ | Limited | Limited | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | Limited |
| DISSECT-CF | 2014 | Java | Open Source | - | ✗ | Full | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ |
| SCORE | 2018 | Scala | Open Source | Google Omega | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| SCORE-GAME | 2018 | Scala | Open Source | Google Omega | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| CDOSim | 2021 | Java | Open Source | CloudSim | ✓ | Limited | Limited | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ | Limited |
| Simcan2Cloud | 2022 | C++ | Open Source | OMNeT++ | ✓ | Limited | Limited | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

*CloudSimNet++ allows the creation of priority VMs, but not the management of multiple profile users with different priorities the exact time when each user sends the request and the specific configuration of the resources required. The last feature denotes the support for including priority users on the simulation platform (*Prior. users*). The last block, labelled *DC*, is focused on the data centre aspects, which include the percentage of hardware used in the DC (*HW usage*), and the support to model – with a high level of detail – the different hardware components of the platform (*HW detail*).

This comparison covers 15 simulation platforms, including Simcan2Cloud. The year of creation of the platforms ranges from 2009 to 2022. Regarding the programming language, Java is the predominant one, as it has been used to code 60% of the simulators studied in this comparison. In contrast, C++ and Scala have been used to write 26% and 13% of the simulation tools, respectively. All the simulation tools – with the exception of CloudExp – have an open-source licence. Most of the simulation platforms are built upon a base platform, and only DCSim and DISSECT-CF are built as independent tools. Among them, CloudSim is the preferred platform for creating new simulators, being used by 40% of them, followed by OMNeT++, which is used by 20% of the simulation tools. The other solutions are based on Google Omega, NS2 and SimJava. On analysing the general aspects of the platforms, we can see that only 40% provide a complete GUI via which the cloud scenario can be fully modelled and customised, while 13% offer a limited GUI that facilitates the configuration of simulated scenarios with significant restrictions. Regarding communications, only 20% implement a full communication model (i.e. communication protocols, such as TCP and UDP), whilst 60% provide a limited model. Finally, full support for designing network topologies is only included in 20% of the proposals.

With regards to the cloud provider's details, we can find two features that are implemented by most of the platforms under study, namely the cost model and scheduling policies, which are supported by 80% of the solutions. Nevertheless, some features are only fully covered by a relatively small percentage of the simulators, such as SLAs, which are only supported by 40%. Furthermore, other features, such as rental extension and the management of queues for handling user requests, are not implemented by any of the proposals, with the exception of Simcan2Cloud.

However, other characteristics, such as providing a flexible and open API for creating workloads and managing different types of users are not supported by the solutions analysed (again excluding our own proposal). It is worth mentioning that only CloudSimNet++ allows the creation of priority VMs, but not the management of multiple profile users with different priorities. Modelling the underlying cloud infrastructure is an important aspect of the simulation platforms since it affects the reliability of the results obtained. In this case, only 20% of the simulators include a highly-detailed infrastructure, allowing the design of heterogeneous systems. In the same line, the monitoring of the percentage of resource usage is only possible in 26% of the systems.

As is shown by the comparison, there are few simulation platforms aimed at describing the cloud provider – with a reasonable level of detail – while maintaining infrastructure support. In general, the simulation platform that shares most features with Simcan2Cloud is CloudNetSim++. However, CloudNetSim++ does not consider certain important aspects of the cloud, such as the extension time for renting VMs, the implementation of user queues to manage different types of user accessing the cloud, priority resources, support for real traces, and a highly-detailed infrastructure.

## Simcan2Cloud

This section presents a detailed description of the Simcan2Cloud simulator. The meta-data is presented in Table 2, in which the current version of the simulator, the link to the repository containing the source code, the legal code licence, and the code versioning system used are shown in the rows labelled *C1-C4*, respectively. The programming language used to write Simcan2Cloud, the

**Table 2** Meta-data of Simcan2Cloud

| Id | Code Metadata | Description |
|---|---|---|
| C1 | Current code version | 0.1 |
| C2 | Permanent link to code/repository used for this code version | https://github.com/pabloccanizares/Simcan2Cloud |
| C3 | Legal Code Licence | GPL |
| C4 | Code versioning system used | git |
| C5 | Software code languages, tools, and services used | C++, OMNeT++, Java |
| C6 | Compilation requirements, operating environments & dependencies | OMNeT++ 5.0, Java 8, TCL/TK 8.4, Bison, Flex, NetBeans 7 or above |
| C7 | If available, link to developer manual/documentation | |
| C8 | Support email for queries | pablo.cerro@uam.es |

dependencies, the compilation requirements, and the documentation are shown in the rows labelled *C5-C7*, respectively. The support e-mail address is shown in the last row (labelled *C8*).

This section has been divided into five different parts. An overview of Simcan2Cloud is presented in Section "Software description". Section "Service level agreements" shows how the SLAs are modelled using different configuration parameters, and the main architecture and the simulation core of the simulator are presented in Section "Architecture". Next, Section "API" presents the API, which provides the facilities to manage cloud environments. Finally, Section 3.5 introduces the GUI, a component that has been designed for easily creating simulated scenarios.

### Software description

Simcan2Cloud is a modular and flexible discrete-event simulation platform that allows users to model and simulate cloud computing infrastructures. Simcan2Cloud has been coded using C++ on top of OMNeT++, an open-source simulation framework [33]. It is worth mentioning that OMNeT++ is considered as a de facto standard in the simulation of distributed systems. In particular, during the last decade, OMNeT++ has been widely adopted by the research community and industry to simulate a broad spectrum of complex systems [34–36]. Among the main features of OMNeT++, we can highlight the structured programming and event-oriented model, which establish the foundations for a high degree of flexibility in the design of distributed systems. However, simulators built using OMNeT++ require the use of the NED language to configure the simulated environments. This fact, in most cases, is a tedious and error-prone task due to the large number of parameters that must be configured in a plain text file. In order to alleviate this task, Simcan2Cloud provides an easy-to-use GUI that allows the modelling of complex infrastructures without the need for an in-depth knowledge of these systems. Furthermore, the modelling process is enhanced with the inclusion of a component repository. The main idea is to enable the reuse of the components required to build a cloud environment, making it possible to model and configure large complex systems within minutes.

The cloud provider is one of the most important parts in a cloud system and, thus, Simcan2Cloud includes a modular and fully customisable cloud provider module. This module is mainly focused on the management of users, scheduling and allocation policies, financial costs, and SLAs. In addition, the cloud provider module allows the inclusion of both customised and well-known virtual machines. The current version of Simcan2Cloud provides a large collection of VM instances inspired by Amazon EC2 [37]. Regarding the physical resources, Simcan2Cloud provides several mechanisms for modelling the four basic subsystems, namely storage, computing, memory and network. Basically, combining the components of these subsystems (i.e. disk drives, communication networks, CPUs, and memories) allows users to build a wide variety of cloud scenarios. These may range from a small number of physical machines to complex and heterogeneous data-centres.

In order to accurately represent the behaviour of cloud systems, Simcan2Cloud is able to generate and process realistic workloads. These are created by using a large number of users, ranging from just a few to thousands. The arrival of these users at the platform can be determined by using different distribution functions. The behaviour of users interacting with the cloud can be easily modelled by determining, in essence, the requested VMs and the applications executed on these VMs. The users that request resources from the cloud provider are managed by using a fully-customisable queue system. The current version of Simcan2Cloud immediately attends to the user requests if the required resources are available. On the contrary, if the system is not able to provide the requested resources, the user has two possibilities. The first option consists of waiting for a pre-defined period of time, with the expectation that the requested resources will – in due course – become available. The second one, is to leave the system without having used any services. Let us remark that this queue system has been specially designed to provide flexibility for including new scheduling policies. Hence, this feature allows to increase the functionality of the Simcan2Cloud simulator. It is worth mentioning that different priorities for users have been also considered, in such a way that the priority criteria employed by the most common service providers can be replicated. Thus, different policies can be applied to allocate the resources requested by the users. To this end, Simcan2Cloud provides a high level of flexibility, making it possible to balance the user requests between different data-centres. This fact is possible by selecting policies based on the occupancy of the physical machines (i.e. avoiding fragmentation), and allowing developers to implement new and customised policies. Similarly, the number of scheduling policies, in the hypervisor module, can be easily extended and customised. One of the key aspects of the cloud provider module is the capability to define and include new SLAs based on both functional and non-functional aspects. Among them, it is worth mentioning cost-based features and provisioning (i.e. abandon rate and waiting time). The definition of the SLAs is described in detail in Section "Service Level Agreements".

### Service level agreements

In this section, we define how Simcan2Cloud implements the service level agreements (SLAs) that are used in simulated cloud scenarios. It is a requirement that cloud service providers should always sign an SLA with the cloud users who wish to use the cloud services. Hence, a user cannot request a virtual machine (VM) in the cloud if it is not included in the signed SLA. Simcan2Cloud defines SLAs based on the way users interact with the cloud and how they are attended to.

More specifically, the architectural level defines several SLA parameters related to the costs of different VMs depending on user types. The current version of Simcan2Cloud allows distinction between two types of users, namely *regular* and *high-priority* users. The former requests VMs from the cloud provider but can wait until resources become available when these requests cannot be met immediately. If the user decides to wait, then they subscribe to the VM characteristics for a specific period, waiting to be notified when a VM with these features becomes available. Once the notification message is received, the user starts executing their applications. If the subscription period expires without the VM becoming available, the user leaves without being able to run their applications. High-priority users should receive the resources they request immediately, but they pay a higher price and must be compensated when the resources are not provided.

The parameters used to define an SLA in Simcan2Cloud include:

1. The base VM cost (*Base*). This is the cost for one hour of VM services under normal circumstances when a request can be dealt with immediately. If no VM is available (with the requested features), the user will be notified and receive a discount.
2. The discounts on the base VM price for delays (*Discount*). If there are no available resources to attend to the *regular* user demands immediately, they can subscribe and wait for the requested resources to become available. In this case, the price will be lower, so the VM renting price will have a discount applied to the normal cost. The user can decide to wait until the required VM is available or leave.
3. An increase in the base VM cost for high-priority users (*Inc_priority*). If a user decides to have priority behaviour in the cloud, they must pay a price above the base price. This increment ensures that the cloud provider reserves some machines to be used when the *normal* (non-reserved and always running) machines cannot meet the user's requirements. Hence, when the *normal* machines are unavailable, the cloud provider should start up a reserve VM to serve high-priority users.
4. Cost for extra execution time (*Offer*). If the execution of the applications deployed by the user does not finish within the estimated renting time, the cloud provider can make an offer to the user to continue execution. Cloud providers offer users an extension to the rental period at a base price per hour plus a surcharge. Thus, the user can either pay for this extra execution time or decline it and stop interacting with the system.
5. Compensation cost due to resource unavailability (*Compensation*). The *high-priority* users pay an additional price to guarantee service. Thus, they must be compensated for damages caused in the unlikely event that the cloud provider has no available VMs. This situation is very unlikely, but it would only occur in cases in which there is no available VM with these requested features in the pool of *reserved* machines. Essentially, this scenario occurs due to an unexpected number of *high-priority* user requests, or as a consequence of a misconfiguration in the cloud. A solution to this would probably focus on the addition of more racks. Thus, additional VMs may be deployed while maintaining a balanced system.

Some parameters are exclusive to *regular* users, such as the discount for waiting for resources. Other parameters are only defined for *high-priority* users, such as the increased cost to receive immediate attention and compensation if no resources are available. These parameters allow us to perform a cost analysis in the tool. One of the most noteworthy characteristics of Simcan2Cloud is its flexibility. Thus, we intend to enrich the SLAs with new parameters and behaviours, for instance, by studying the procurement schemes of Amazon Web Services and their combinations.

Table 3 illustrates three examples of different possible scenarios in the cloud. These scenarios are achieved by varying the cost parameters defined in an SLA signed between the users and the cloud provider. Based on AWS EC2 on-demand instances of VM [37], a cost of 0.012 USD is established for *base*. The three SLA scenarios directly impact cloud users and providers income. In $SLA_1$, users receive excellent offers, including a 60%

**Table 3** SLAs configurations for several cloud scenarios

| Type | Base (per hour) | Discount (% of the cost) | Inc-priority (% of the cost) | Offer (% of the cost) | Compensation (% of the cost) |
|---|---|---|---|---|---|
| $SLA_1$ | 0.012 | 60 | 10 | 5 | 80 |
| $SLA_2$ | 0.012 | 20 | 50 | 30 | 10 |
| $SLA_3$ | 0.012 | 10 | 70 | 70 | 5 |

discount for initially unavailable resources, a 10% premium for priority behavior, and a 5% cost increase for extending application execution. High-priority users also get 80% compensation for unavailability. $SLA_2$ offers moderate benefits, with a 20% discount, a 50% premium for priority, a 30% cost increase for application continuation, and 10% compensation. In contrast, $SLA_3$ heavily favors the provider with high prices, offering only a 10% discount, 70% premium for priority, 70% cost increase for application continuation, and 5% compensation. The cloud provider income depends on these parameters, striking a balance between user attractiveness and profit maximisation.

## Architecture

In cloud computing, users have access to a delocalised computing infrastructure through the Internet. The simplicity in accessing these platforms and the wide offer of computing configurations encourage the massive use of cloud computing systems. However, the significant increment in the number of users concurrently accessing the system, without considering an appropriate re-scaling of the infrastructure, can lead to the appearance of bottlenecks. The design of the cloud infrastructure plays a key role in avoiding this scenario, but this task is challenging since it is necessary to consider other important factors. Among them, let us mention, just to name a few, obtaining a good cost-performance ratio, analysing the behaviour of the users, and their management when accessing the platform.

Figure 1 depicts the main architecture of Simcan2Cloud. In essence, a cloud scenario modelled in Simcan2Cloud consists of three main modules: a *User generator*, a *Cloud provider*, and a *Data centre*. The behaviour of each one of these modules is coded into a *Manager* submodule. Thus, in order to fully customise the behaviour of the cloud, new managers can be coded using the API showed in Table 4.

The *User generator* module (see left-most module in Fig. 1) generates realistic workloads to be processed by the simulated cloud scenarios. Firstly, the configuration parameters that allow the customisation how the users are created to represent the workload must be provided (label 1). These parameters are processed by the *User Manager* submodule to create the workload, which can be represented by a real-world trace, or by different statistical distributions (label 2). Next, the workload is created at run-time (label 3). It is important to remark that Simcan2Cloud is a discrete-event-based simulator and, consequently, the computation required to create the workload does not affect the operations performed to represent the behaviour of the target system.

The module in the centre of Fig. 1 represents the *Cloud provider*. In general terms, the main tasks of a cloud provider consist in attending to user requests and returning the answers generated by the data-centres to the users. More specifically, the cloud provider is in charge of four main tasks: i) Handling the VM requests from the users; ii) Forwarding the jobs requested by the users to a suitable data-centre, in such a way that the requested VMs are executed on the available physical resources; iii) Managing the subscriptions of the users to the requested resources; and iv) Defining cost policies for each VM instance type. Note that this behaviour can be modified by including a new *CP Manager* submodule, which can be coded using the API provided by Simcan2Cloud. Once

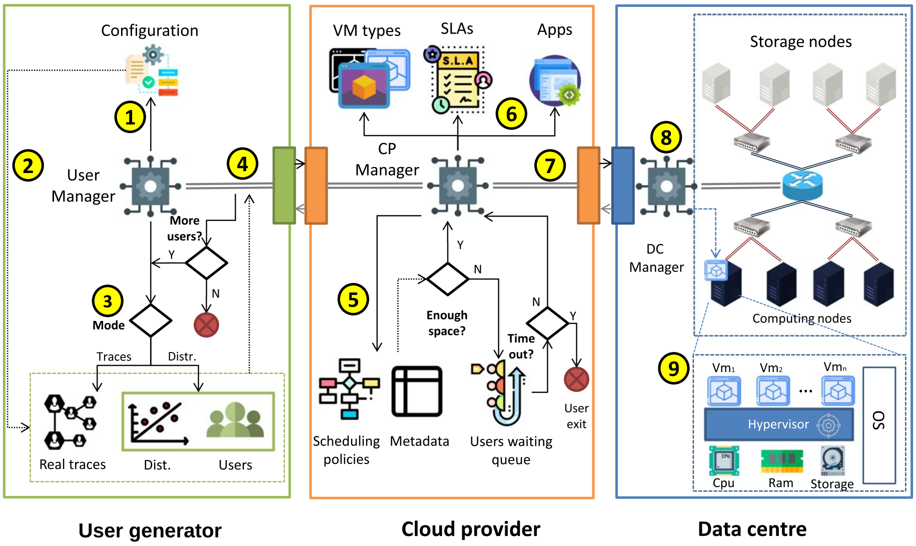


**Fig. 1** Main architecture of Simcan2Cloud

**Table 4** Excerpt from the Simcan2Cloud API

| Component | Id | Method | Description |
|---|---|---|---|
| UserGeneration | 1 | initialise | Initialises the user generation module. |
| UserGeneration | 2 | generateShuffledUsers | Generates a users workload with a random order. |
| UserGeneration | 3 | getNextUser | Obtains the next user to be processed. |
| UserGeneration | 4 | sendRequest | Sends a VM request to the cloud provider. |
| UserGeneration | 5 | subscribe | Sends a subscription request to the cloud provider. |
| UserGeneration | 6 | createVmRequest | Creates a VM request for a specific user. |
| UserGeneration | 7 | handleResponseVmAccept | Handles the accept response of the cloud provider to a specific VM request. |
| UserGeneration | 8 | handleResponseVmReject | Handles the reject response of the cloud provider to a specific VM request. |
| UserGeneration | 9 | updateVmUserStatus | Updates the status of a specific VM. |
| UserGeneration | 10 | submitService | Submits a service to be executed in the cloud. |
| UserGeneration | 11 | createAppRequest | Generates a request for the execution of an application. |
| UserGeneration | 12 | handleResponseAppAccept | Handles an accept response sent by the cloud provider for the execution of a specific application. |
| UserGeneration | 13 | handleResponseAppReject | Handles a reject response sent by the cloud provider for the execution of a specific application. |
| UserGeneration | 14 | handleResponseAppTimeout | Handles a timeout response sent by the cloud provider for the execution of a specific application. |
| UserGeneration | 15 | calculateStatistics | Generates a report with the statistics obtained during the simulation. |
| CloudProvider | 16 | initialise | Initialises the cloud provider module. |
| CloudProvider | 17 | checkVmUserFit | Checks whether the VMs requested by a user fits in a data-centre and sends the request to it. |
| CloudProvider | 18 | updateSubsQueue | Updates the subscription queue. |
| CloudProvider | 19 | notifySubscription | Notifies users that the system is ready to receive their service submissions. |
| CloudProvider | 20 | timeoutSubscription | Notifies users that their subscription time has expired. |
| CloudProvider | 21 | handleUserAppRequest | Forwards the request for the execution of an application to the data-centre. |
| DataCentreManager | 22 | initialise | Initialises the data-centre manager module. |
| DataCentreManager | 23 | checkVmUserFit | Checks whether the VMs requested by a user fits in the system. |
| DataCentreManager | 24 | getTotalCoresByVmType | Returns all the computational cores required by a VM type. |
| DataCentreManager | 25 | acceptVmRequest | Accepts a VM request. |
| DataCentreManager | 26 | rejectVmRequest | Rejects a request for the execution of an application. |
| DataCentreManager | 27 | allocateVM | Allocates a VM in the cloud. |
| DataCentreManager | 28 | handleUserAppRequest | Handles the request for the execution of an application sent by a specific user. |
| DataCentreManager | 29 | acceptAppRequest | Accepts a request for the execution of an application. |
| DataCentreManager | 30 | rejectAppRequest | Rejects a request for the execution of an application. |
| DataCentreManager | 31 | timeoutAppRequest | Notifies the user that the time for the execution of an application has expired. |

a user arrives at the cloud (label 4), the cloud provider manager checks whether this user has previously signed an SLA. Thus, their requests are processed accordingly using the corresponding scheduling policy (label 5).

When the resources can be accessed by the user that requested them, the *CP Manager* submodule (label 6) creates the VMs containing the applications to be executed. The VMs are deployed using the features reflected in the signed SLA. Next, the requested VMs are sent to the *Data centre* module to be deployed on the available physical resources (label 7).

Once a VM arrives in the *Data centre* module, the *DC Manager* locates potential physical machines to deploy the VM, which must satisfy the hardware requirements indicated in the VM settings (label 8). Next, once the hardware to deploy the VM has been located, the VM is set up on the corresponding physical machine and its execution starts (label 9). Finally, the *DC Manager* sends a message to the *CP Manager* to update the list of available resources in the Data centre.

Figure 2 shows a class diagram containing the main classes of the Simcan2Cloud simulator and how these classes are related. Further details can be found at the Appendix B. The main classes of the *User generator* module are shown with a green background. *UserBase* is the main entity and provides the basic functionality, in terms of data management and structures, of the user modules. The *UserGenerator* class provides different methods for creating workloads by specifying the exact moment when the users arrive in the cloud, either randomly, or according to

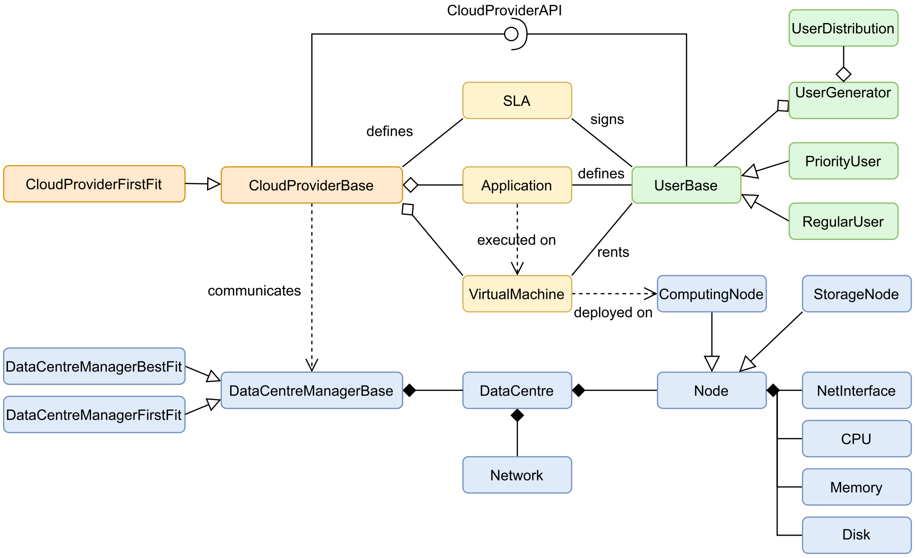


**Fig. 2** Class diagram that represents the main classes of the Simcan2Cloud simulation core

a specific distribution by using the *UserDistribution* class. The generated workloads may consist of different types of user, such as *PriorityUsers* and *RegularUsers*. However, the integration of new user generators and user instances can be performed by creating new classes that inherit from *UserGeneratorBase* and *UserBase*, respectively.

The most important classes that the *Cloud provider* module implements are shown with an orange background in the class diagram (see Fig. 2). *CloudProviderBase* is the main entity of the cloud provider. Some of its most important features are creating the structure and the hierarchy of objects relating to the data-centres, and finding information about a specific data-centre component. The integration of new cloud providers can be performed by creating a new class that inherits from *CloudProviderBase*. The new class must contain a new policy for selecting the data-centres where the VMs requested by the users are deployed. Currently, *CloudProviderFirstFit* inherits from this class and implements an algorithm that selects the first data-centre where the request fits.

In essence, the *Data centre* module contains the physical resources supporting the cloud, which are categorised into storage nodes and computing nodes. A physical machine is defined by setting up the basic sub-systems, that is, CPU, storage, memory, and network. These nodes are interconnected through a communication network and can be modelled independently from each other, thus allowing the composition of heterogeneous data-centres. All the data-centre infrastructure is managed by a component called *data-centre manager*. The main classes of the Simcan2Cloud infrastructure are shown with a blue background in the class diagram depicted in Fig. 2. *DataCentreManagerBase* is the main entity of the data-centre, and its most important features are, among others, allocating the VMs on the physical machines, and scheduling the jobs to be executed on the VM instances. The integration of new data-centre managers can be performed by creating a new class that inherits from *DataCentreManagerBase*. Currently, Simcan2Cloud incorporates two different data-centre managers, namely *DataCentreManagerFirstFit* and *DataCentreManagerBestFit*. The former allocates the VMs requested by the user in the first available slot of the list that contains the available resources. This slot can be a node or a rack, depending on the computational resources required to allocate the requested VMs. The latter is focused on avoiding fragmentation and, therefore, this policy deals with allocating the user requests in the slot that has the smallest quantity of resources available and into which the request fits. Additionally, this class provides different methods for obtaining information from the resources of a data-centre, such as the number of available CPU cores and the total number of CPU cores. The *DataCentre* entity consists of a collection of physical resources required for the proper functioning of the system, such as lists of computational and storage resources instantiated by the *Nodes* class.

## API

For this purpose, it implements the main functionality of the simulation platform, that is, the user generation engine and the cloud provider functionality. Through the use of this API, it is possible to include in the platform new user and cloud provider instances with customized behaviour.

This section describes the API of Simcan2Cloud. The main methods of this API are summarised in Table 4. For the sake of clarity, only the methods belonging to the main modules – *user generation*, *cloud provider* and *data centre manager* – are shown. The first column shows the cloud component that contains the method, and the following columns refer to the ID, name, and description of the method. It is important to point out that these methods are implemented in the current version of Simcan2Cloud and can be overwritten, if necessary, to provide specific functionalities. The idea is that new modules, such as resource allocation policies and user behaviours, can be easily included in the simulator by using this API.

The user generation core is detailed in methods 1-15. Method 1 initialises the data structures required to start the simulation, while methods 2-3 manage the users arriving in the cloud. More specifically, *generateShuffledUsers* generates a workload by randomly establishing the moment when the users arrive in the cloud. This method is particularly useful for reproducing the randomness of user access to the cloud in real environments. Once generated, the workload can be iterated by using the *getNextUser* method, which provides the next user to be processed by the cloud.

The creation and management of the VMs are handled by methods 4-9. The VMs requested by the users are created and sent to the cloud provider with the *createVmRequest* and *sendRequest* methods, respectively. Depending on the resources available in the cloud, the cloud provider sends a notification message that is handled and updated by the user through the *handleResponseVmAccept*, *handleResponseVmReject* and *updateVmUserStatus* methods. If the cloud provider sends a rejection message, which means that the VMs requested by the user do not fit in the data-centre, the user may send a subscription request to the cloud by using the *subscribe* method. At this point, several options exist for the developers to create new user behaviours, such as choosing between subscribing to the cloud provider or leaving the cloud with their request unattended to.

The services required by the users are handled by methods 10-14. The applications to be executed on the VMs are created and sent to the cloud provider by using the *createAppRequest* and *submitService* methods. The response of the cloud provider is managed by the *handleResponseAppAccept*, *handleResponseAppReject*, and *handleResponseAppTimeout* methods. Similarly to the case of the VM requests, it is possible to create new user behaviours by considering the user's decisions depending on whether the cloud provider rejects the request. For example, the user can select another application or reduce the number of applications executed on a VM. Finally, method 15 generates a report containing the statistics obtained during the simulation.

The functionality of the cloud provider is managed by methods 16-21. The creation, initialisation, and configuration of both the data-centre infrastructure and the cloud provider are carried out via method 16, while methods 17-18 manage the VMs requested by the user. Method *checkVmUserFit* analyses whether the available resources of one of the data-centres meet the request requirements. Then, if the request fits in the data-centre, that is, the requested VMs can be executed on the available resources, the cloud provider forwards the request to the selected data-centre.

The subscription of the users to the cloud is managed by methods 18 and 20. The *updateSubsQueue* method analyses the subscription queue to find timeouts that reach the maximum waiting time for users to obtain the requested resources and selects the next requests to be processed. It is worth mentioning that new queue systems for managing users can be coded by overwriting the *updateSubsQueue* method. The *notifySubscription* method notifies the user that the requested resources are available. The *timeoutSubscription* method manages the waiting time for users when the subscription timeout expires and the requested resources remain unavailable. The method 21, namely *handleUserAppRequest*, manages the services submitted by the users and forwards the request to the data-centre where the VMs have previously been allocated.

The functionality of the data-centre manager is reflected in methods 22-31. The creation, initialisation, and configuration of the data-centre infrastructure are carried out by method 22. Let us suppose that a new resource allocation policy, containing new and complex data structures, is included in the simulator. In this case, methods 16 and 22, both named *initialise*, must be overwritten in order to handle the new data structures. Methods 23-27 manage the VMs requested by the user. Method 23, namely *checkVmUserFit*, analyses whether the available resources of the data-centre meet the request requirements, and if the request fits in the data-centre. This check focuses in calculating if the data-centre contains enough machines with free resources to run the requested VMs. In such a case, the data-centre manager allocates the requested VMs and sends the user an acceptance message using the *acceptVmRequest* and *allocateVM* methods, with ID 25 and 27, respectively. Otherwise, if the request does not fit in the system, the data-centre manager sends a rejection message using method 26, namely *rejectVmRequest*. In the current version of Simcan2Cloud, as mentioned in Section "Architecture", two different allocation policies are included, namely *best-fit* and *first-fit*. However, the integration of new policies can be easily performed by implementing new cloud providers with different policies in the *checkVmUserfit* and *allocateVM* methods. Methods 28-31 handle the services requested by the users. Similarly to method 21, *handleUserAppRequest* manages the services submitted by the users and allows request acceptance – or rejection – via methods 29 and 30, respectively. Finally, if the execution of the application exceeds the renting time, a timeout notification is sent to the user via method 31, namely *timeoutAppRequest*.

## GUI

Simcan2Cloud allows the modelling and design of cloud infrastructures with a high level of flexibility by configuring, among other modules, resource allocation policies, data-centres, and workloads. In essence, the configuration of a cloud to be simulated in Simcan2Cloud consists of two plain text files, where one defines the general architecture of the cloud (data-centre, cloud provider, and generation of users) and the other contains all the parameters required to configure each module in the cloud environment (features of each physical machine, configuration of VMs, and distribution of users in the workload, among others). Hence, manually setting all the required parameters to configure the simulated cloud is an error-prone and tedious task.

In order to facilitate both the configuration and the interaction with the simulation engine, Simcan2Cloud provides a graphical user interface (GUI) (see Fig. 3) and consists of three main parts: a tabbed panel, a tree panel and a menu.

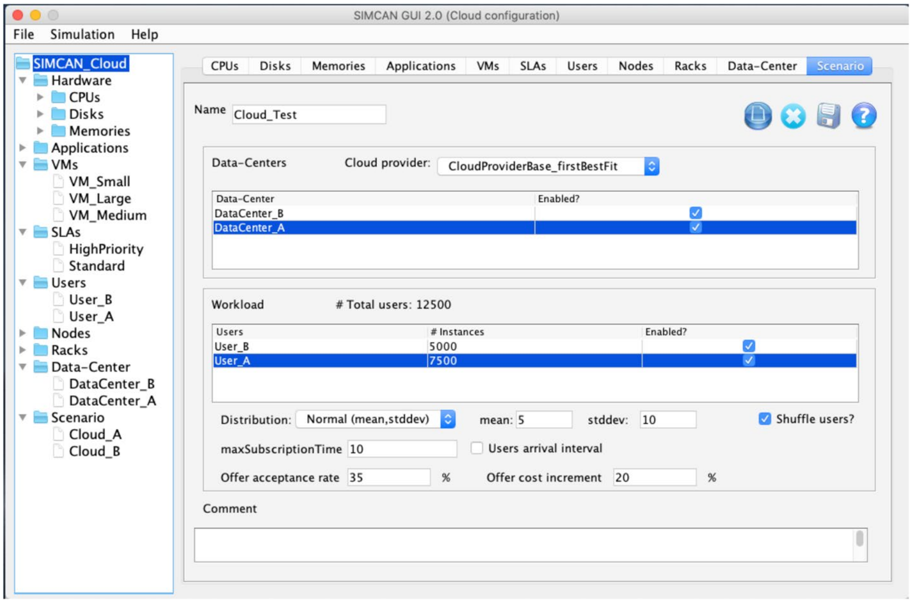
The tree panel, which is located to the left of the tabbed panel, shows the repository of the Simcan2Cloud simulator, which consists of all the elements that have been previously modelled with the tabbed panel. This tree panel also makes it possible to reuse, edit, and remove the components from the repository.

The tabbed panel consists of 10 different editors that allow users to model and configure the different parts of the cloud, that is, CPUs, disks, memories, applications, VMs, SLAs, users, nodes, racks, data-centres, and scenarios. The first three tabs, namely *CPUs*, *Disks* and *Memories*, show the ways of configuring the computational resources of a physical machine. These components are used to customise both the computing and storage nodes.

The applications submitted by the users for execution on the VM can be modelled – in the *Applications* panel – by configuring the total number of CPU and I/O operations to be processed. In this way, it is possible to create different application types, such as data-intensive and CPU-intensive applications, which are focused on processing a large number of I/O and CPU operations, respectively. Alternatively, the number of iterations can be also configured. Thus, the final user is able to model the length of the execution when the application mixes CPU and I/O operations.

The *VMs* panel configures the number of cores, the storage capacity (measured in GB), the cost per hour, the number of cores, and the memory (measured in GB) of each VM.

SLAs are configured in the *SLAs* panel. Thus, when a new SLA is created, the configuration parameters for the existing VMs in the repository are displayed in a table. Figure 4 shows the configuration of *Sla_1*. In this case, the three VMs that are stored in the repository (see left frame in the figure), that is, *VM_large*, *VM_medium*, and *VM_small*, can be configured for the current SLA.



Name Cloud_Test

Data-Centers Cloud provider: CloudProviderBase_firstBestFit

| Data-Center | Enabled? |
|---|---|
| DataCenter_B | ☑ |
| DataCenter_A | ☑ |

Workload # Total users: 12500

| Users | # Instances | Enabled? |
|---|---|---|
| User_B | 5000 | ☑ |
| User_A | 7500 | ☑ |

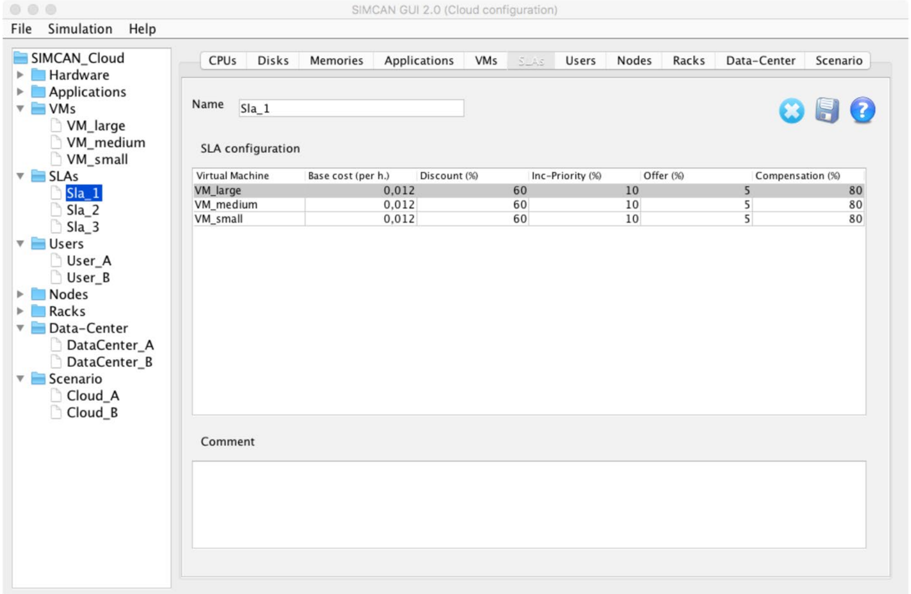
Distribution: Normal (mean,stddev) mean: 5 stddev: 10 ☑ Shuffle users?

maxSubscriptionTime 10 □ Users arrival interval

Offer acceptance rate 35 % Offer cost increment 20 %

Comment

**Fig. 3** Screenshot of the Simcan2Cloud GUI



Name Sla_1

SLA configuration

| Virtual Machine | Base cost (per h.) | Discount (%) | Inc-Priority (%) | Offer (%) | Compensation (%) |
|---|---|---|---|---|---|
| VM_large | 0,012 | 60 | 10 | 5 | 80 |
| VM_medium | 0,012 | 60 | 10 | 5 | 80 |
| VM_small | 0,012 | 60 | 10 | 5 | 80 |

Comment

**Fig. 4** Screenshot of the SLAs configuration in Simcan2Cloud GUI

In the table, the cost and other parameters, like offer and compensation, can be set. Once these parameters are assigned, the user can save the SLA in the repository, which will be displayed in the components tree.

The *Users* tab allows the modelling of the behaviour of users interacting with the cloud. To that end, the number and type of the requested virtual machines, and the application to be executed on these VMs, must be configured.

The *Data-Centre* tab enables the configuration of the data-centre supporting the cloud, which is modelled by configuring the computing and storage racks. In essence, a rack is a collection of nodes interconnected through a communication network. In order to model a rack, the user must choose a node configuration – from the *Nodes* tab – and the number of nodes provided. Let us mention that the modular design of Simcan2Cloud makes it possible to easily model different cloud infrastructures by using the components in the repository (see the left frame of the GUI). Finally, the user must configure a communication network to connect the racks of the data-centre.

The simulation scenario can be modelled in the *Scenario* panel. For this, it is necessary to select the underlying infrastructure of the cloud (data-centres), the workload to be processed, the distribution of the users arriving in the cloud, and several parameters related to the subscription time and costs.

Finally, the menu, which is at the top of the GUI, consists of the management options. This menu creates configuration files, and shows the Simcan2Cloud help and other auxiliary operations.

## Empirical study

This section presents a thorough empirical study in which different cloud configurations are modelled to check the applicability and usability of Simcan2Cloud. Each cloud configuration has a homogeneous data-centre, that is, all the physical machines contain the same features: a quad-core CPU, 64 GB of RAM memory, and a storage device of 500 GB with a read/write bandwidth of 500 Mbps. These physical machines – hosting the VMs requested by the users – are interconnected through a Gigabit Ethernet communication network, and the cloud manager is connected to the cloud through a 40 Gbps Ethernet network. In this study, we use four different configurations for the data-centre supporting the cloud, consisting of 128, 256, 512, and 1024 physical machines.

In order to analyse the different features of the platform, we have divided the empirical study into two parts. The first one focuses on studying the behaviour of cloud systems considering synthetic workloads, while the second one consists in analysing traces extracted from a real-world system.

### Experiment 1: Synthetic workloads and multiple CPU configurations

In this part, we use two different CPU configurations – with different computing power for the CPU cores – to analyse how the computing power affects the overall system performance. In addition, we have created a synthetic workload – based on an exponential distribution – for conducting the experiments. In this way, each cloud processes a workload consisting of 10,000 users requesting resources. This workload has been generated using four different user roles and three different configurations for the virtual machines. All users with the same role exhibit identical behaviour. Table 5 shows the configuration of the VMs, where the first column refers to the name of the VM, and the next columns represent – respectively – the CPU, the memory, and the storage used by the virtual machine. A detailed description of the workload is given in Table 6, in which the first column shows the name of the user role, the second column gives the number of instances created for this user role, and the last column contains the number of VMs requested for a specific time frame. Particularly, this workload contains – among other user instances – 3725 instances of `LemmingUser` users, each one requesting 2 `vmMedium` VMs for two hours. The *timestamp* indicating when users arrive to the system has been calculated using an exponential distribution with *mean=60.5 seconds*. On each requested VM, the users deploy a CPU-intensive appplication that executes five iterations of the following actions: read 10MB of data from disk → execute 79,200,000 MIs → write 5 MB of data to disk. The maximum subscription time is 10 hours, which means that those users that were not able to access the requested resources in this time, left the system unattended. Finally, 90% of those users that needed more time to execute the submitted applications – once the renting time expires – requested an extension to allow the successful execution of the applications.

Figure 5 shows the overall system performance when processing the workload. The y-axis shows the waiting time for each user, that is, the time elapsed from when the user requests resources to the cloud provider, until the moment when the user gains access to them. This waiting time is computed by considering all users that have waited for resources in the system. The x-axis displays the four configurations for the data-centre supporting the cloud, consisting of 128, 256, 512, and 1024 physical machines, which are shown in blue, orange, green, and red, respectively. Each dot represents a user that was attended to, that is, the cloud was able to provide the requested resources. Note that unattended users are not shown in these charts. This experiment was carried out using two different CPU configurations. Thus, Fig. 5a and b show the results for the physical machines containing a CPU with a computing power of 40k MIPS and 70k MIPS, respectively.

**Table 5** Configuration of virtual machines for generating the workload

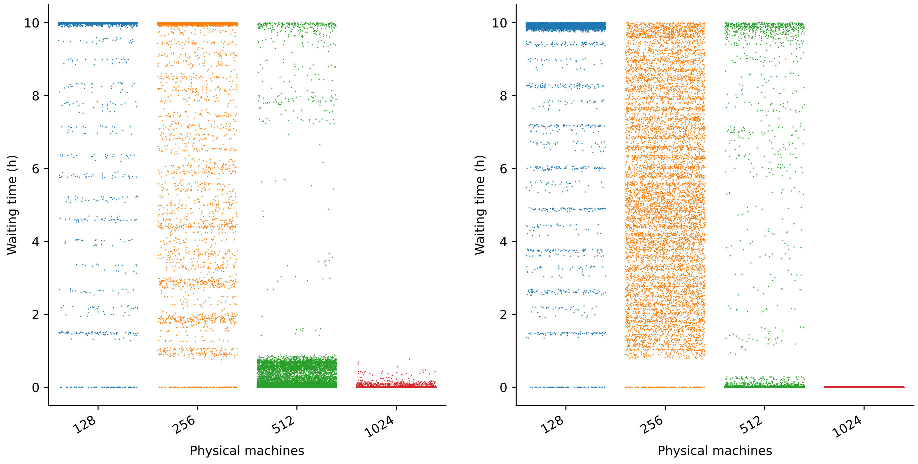
| VM Name | CPU cores / SCU | RAM Memory | Storage |
|---|---|---|---|
| `vmSmall` | 1 | 2 GB | 250 GB |
| `vmMedium` | 2 | 4 GB | 500 GB |
| `vmLarge` | 4 | 8 GB | 1 TB |

**Table 6** Configuration of user roles for generating the workload

| User role | # User instances | Requested resources |
|---|---|---|
| `MinionUser` | 5000 | 5x`vmSmall` 2h |
| `LemmingUser` | 3725 | 2x`vmMedium` 2h |
| `SmurfUser` | 125 | 50x`vmMedium` 2h |
| `FraggleRock` | 1150 | 5x`vmLarge` 3h, 5x`vmMedium` 2h |

Figure 5a shows that when the cloud is supported by 128 physical machines, the dots in the chart are dispersed, which clearly reflects that the system reaches the saturation point and, consequently, few users access the requested resources (blue column). When the number of physical machines increases up to 256 and 512 (orange and green columns), the chart renders a different scenario in which the dots are more condensed, meaning that a greater number of users access the resources, hence reducing the saturation in the system. However, when the cloud contains 512 nodes, there were few users waiting nearly ten hours to access the requested resources. Finally, when the cloud is configured with 1024 physical machines, the stress of the system is significantly reduced, which allows the cloud to provide the resources to all the users in the workload.

Chart Fig. 5b shows the results for processing the workload when the CPU power of the physical machines is increased up to 70k MIPS. In this case, we observe a similar tendency to the one shown in the previous chart, that is, increasing the number of physical resources improves the overall system performance by reducing the waiting time. However, it is worth noting that although the cloud is also saturated when the number of physical machines is equal to or below 512, the waiting time is shorter. The main difference between these scenarios – using a CPU@40k MIPS and a CPU@70k MIPS – lies in the number of applications that were successfully executed before the renting time of the VMs expires. Thus, in those cases in which all the applications submitted by the user are completely executed, the requested VMs are liberated, the user leaves the system, and the requests of new users are processed. On the contrary, when the applications are not completely executed, the user can request an extension and, consequently, these resources are not provided to new users. In this case, the system might become completely saturated if the available resources are not enough to process the workload. When the cloud contains 1024 physical machines, all the users are immediately attended to, which is shown in the red column.



(a) CPUs@40k MIPS (b) CPUs@70k MIPS

**Fig. 5** Performance evaluation using different CPUs in the physical machines

Tables 7 and 8 show detailed information of this experiment. The first column – labelled as *Machines* – shows the number of physical machines supporting the cloud. The following columns, labelled as *U. Attended*, *Average*, *Std*, and *Min*, show – in hours – the number of users attended to, the average waiting time, the standard deviation, and the minimum waiting time for all the users in the workload that were successfully attended to, respectively. The next three columns display the 25*th*, 50*th* and 75*th* percentile, that is, once the data are sorted – from lowest to highest – the waiting time below which the 25%, 50% and 75% of the users are found, respectively.

In this experiment, two simulation parameters have a direct impact on the overall system performance. First, the CPU power provided by the physical machines. When the CPU is improved, we observe an increment in the number of users attended to (see the column labelled *U. Attended*). When the cloud provides 128 physical machines, we observe that using a CPU@70k MIPS allows the cloud to successfully attend to 4399 users (see Table 8). However, when a CPU@40k MIPS is used, only 2163 users are attended to (see Table 7). It is important to point out that this improvement is more noticeable when the cloud uses a small number of physical machines (128 and 256). Using a higher number of physical machines leads to a similar result for both CPUs. In fact, those clouds providing 1024 physical machines, successfully attended to all the users in the workload with both CPUs. Additionally, the average waiting time is only reduced when the cloud provides 512 physical machines, from 0.771 hours using the CPU@70k MIPS to 0.634 hours using the CPU@40k MIPS. In the rest of the cases, using the most powerful CPU led to longer waiting times since the number of users attended to is significantly greater. The second parameter is the size of the cloud – represented by the number of physical machines – which has a direct impact on the number of users that are successfully attended to. In particular, this parameter is directly related to the saturation of the system, especially when the number of physical machines supporting the cloud is small, hence not allowing the system to fully process the workload. Consequently, as the number of physical machines increases, a higher number of VMs are deployed in the system and, therefore, a greater number of users are attended to. We observe that this difference is more noticeable when a small number of physical machines is used. Both tables show that the saturation of the cloud is clearly alleviated when the system contains more than 512 physical machines.

Figure 6 shows the time usage for each CPU core when processing the workload. The x-axis shows the core ID, and the y-axis represents the percentage of time usage for each CPU core, where 100% represents the total time required to fully process the workload. These charts show that when the cloud provides up to 512 physical machines, the time usage for each CPU is nearly 100%. These cases clearly reflect the saturation of the system. However, when the cloud uses 1024 physical machines, we notice that there is a significant number of CPU cores with a low percentage of time usage. In particular, this can be observed in the CPU cores ranging from 3300 to 4096 – in Fig. 6d – and in the CPU cores ranging from 2500 to 4096 in Fig. 6h.

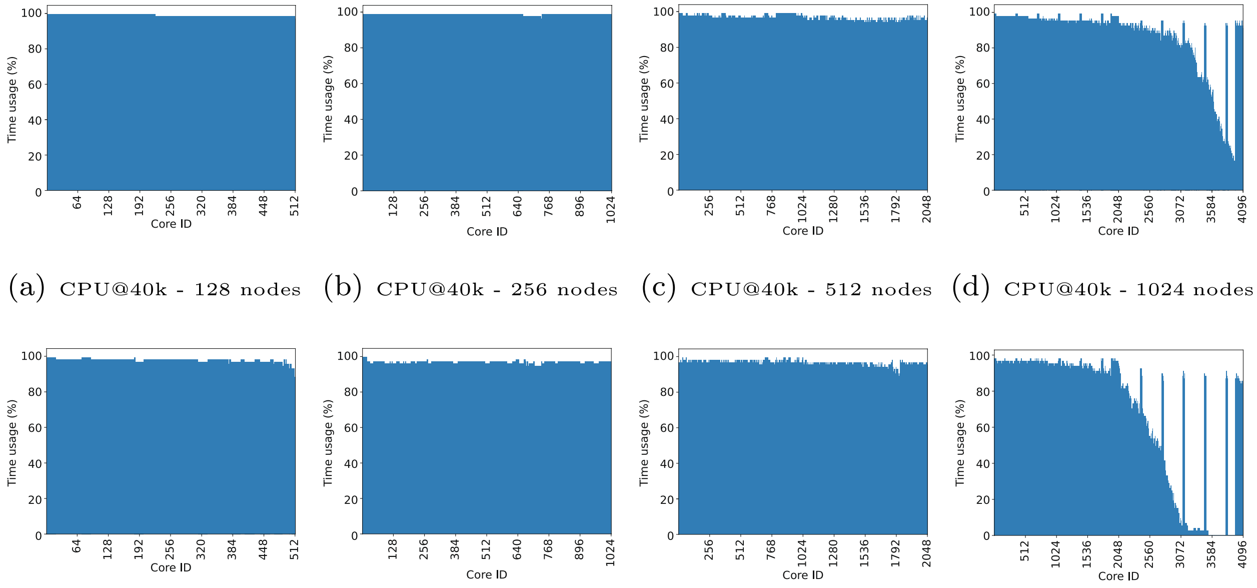
Figure 7 shows the percentage of CPU cores in use when processing the workload. The x-axis shows the timeline and the y-axis represents the percentage of CPU cores in use.

**Table 7** Results obtained when processing the workload using CPUs@40k MIPS

| Machines | U. Attended | Average | Std | Min | 25% | 50% | 75% | Max |
|---|---|---|---|---|---|---|---|---|
| 128 | 2163 | 8.692 | 2.765 | 0.0 | 9.91 | 9.985 | 9.995 | 9.999 |
| 256 | 5089 | 7.791 | 3.212 | 0.0 | 5.39 | 9.963 | 9.990 | 10.0 |
| 512 | 9183 | 0.634 | 1.874 | 0.0 | 0.02 | 0.111 | 0.527 | 9.999 |
| 1024 | 10000 | 0.004 | 0.030 | 0.0 | 0.0 | 0.0 | 0.0 | 0.770 |

**Table 8** Results obtained when processing the workload using CPUs@70k MIPS

| Machines | U. Attended | Average | Std | Min | 25% | 50% | 75% | Max |
|---|---|---|---|---|---|---|---|---|
| 128 | 4399 | 8.908 | 2.350 | 0.0 | 9.81 | 9.898 | 9.953 | 9.999 |
| 256 | 8689 | 5.366 | 2.702 | 0.0 | 3.12 | 5.210 | 7.637 | 9.999 |
| 512 | 9638 | 0.771 | 2.518 | 0.0 | 0.0 | 0.0 | 0.003 | 9.998 |
| 1024 | 10000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |



(a) CPU@40k - 128 nodes (b) CPU@40k - 256 nodes (c) CPU@40k - 512 nodes (d) CPU@40k - 1024 nodes

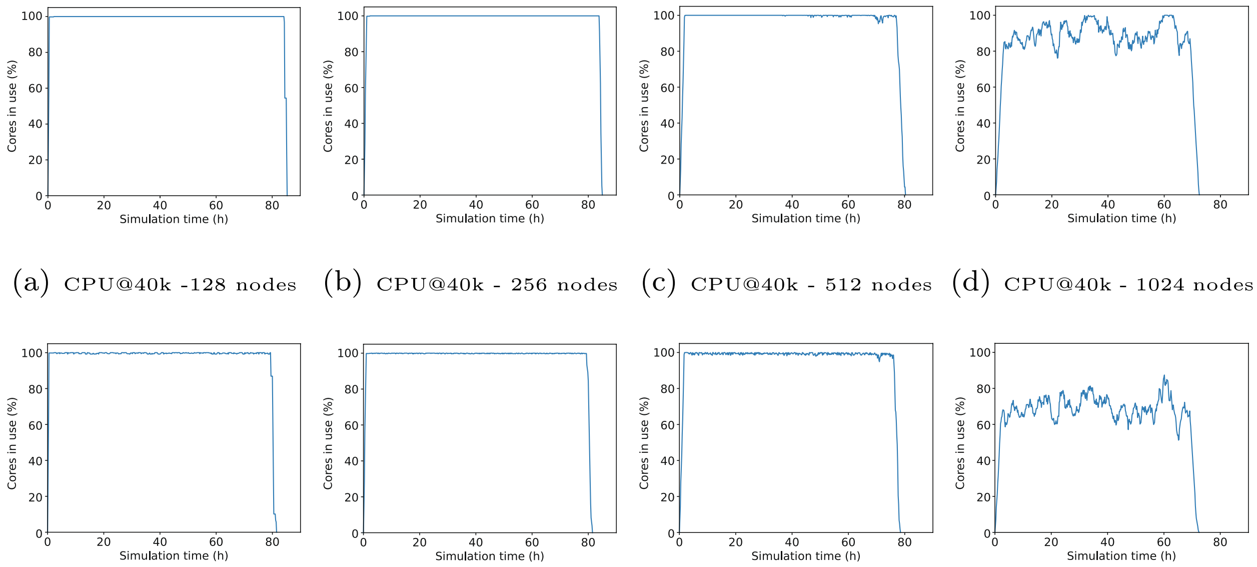
(e) CPU@70k - 128 nodes (f) CPU@70k - 256 nodes (g) CPU@70k - 512 nodes (h) CPU@70k - 1024 nodes

**Fig. 6** Time usage (as %) of each CPU core to process the workload

Similarly to the previous experiment, in this case we can also observe a saturated system when the number of physical machines supporting the cloud is equal to, or below, 512. In particular, almost 100% of the CPU cores were used during the total time required to process the workload. However, increasing the number of physical machines up to 1024 renders a completely different scenario, which provides different results depending on the CPU used. Thus, using CPUs@40k MIPs, the percentage of CPU cores used during the simulation ranged from 75% to 100%. However, using the fastest CPU significantly improves the overall system performance, in such a way that the percentage of CPUs required ranges from 57% to 83%, which allows the cloud to immediately process new user requests, that is, the users are attended to as the resources are requested.

### Experiment 2: Real world traces and SLAs

The main objective of this experiment is to analyse the behaviour of the cloud taking into consideration $SLA_2$, described in Table 3. Specifically, we have modelled four different scenarios, where a different percentage of



(a) CPU@40k -128 nodes (b) CPU@40k - 256 nodes (c) CPU@40k - 512 nodes (d) CPU@40k - 1024 nodes

(e) CPU@70k - 128 nodes (f) CPU@70k - 256 nodes (g) CPU@70k - 512 nodes (h) CPU@70k - 1024 nodes

**Fig. 7** Percentage of CPU cores used to process the workload

priority users (PU) and reserved machines (RM) have been selected:

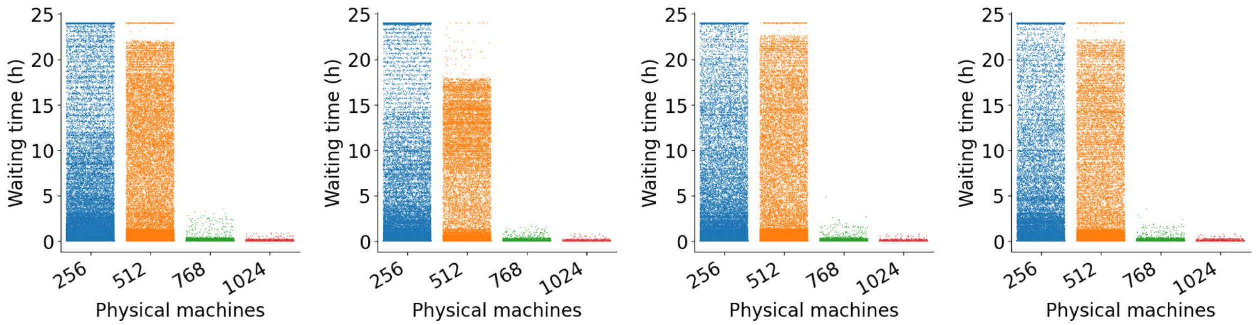
- 0% of priority users and 0% of reserved machines.
- 10% of priority users and 10% of reserved machines.
- 30% of priority users and 10% of reserved machines.
- 30% of priority users and 30% of reserved machines.

In this experiment, in contrast to the previous one where a synthetic workload is used to represent the users attended by the cloud, we use a trace – written in SWF format – obtained from a real-world system[1]. Particularly, this trace consists of 51987 jobs, launched from May 2014 to August 2014, and has been extracted from the GAIA cluster, which is located in the University of Luxemburg. This trace has been pre-processed to remove jobs with non-valid parameters, such as runtime = 0. As a result, 130 jobs were removed, leaving a total of 51857 jobs executed in the simulator. Regarding the data-centre supporting the cloud for processing this trace, we have modelled four different configurations consisting of 256, 512, 768 and 1024 physical machines. Since the trace does not specify the virtual resources requested to execute each job, we use the *vmSmall* instance (described in Table 5) for all users in the simulated environment. This experiment has been conducted on the basis of that each VM must be rented by a minimum of 1 hour. In order to compare the results obtained by using Simcan2Cloud, and those generated in the experiment executed over the GAIA cluster, we have extended the experiments by removing this time limitation.

Figure 8 shows the overall system performance for processing the previously described trace. The y-axis represents the waiting time, while the x-axis shows the number of physical machines supporting the cloud. In addition, we have carried out the experiment varying the percentage of priority users and reserved machines. As we observed in the previous experiments, increasing the number of nodes decreases the waiting time of the users. In particular, this experiment shows that increasing the number of both priority users and reserved machines – in a proportional way – also decreases the waiting time. Specifically, this can be seen in the Fig. 8a and b, where the percentage of priority users (PU) and reserved machines (RM) variates from 0% to 10%, respectively. It is worth mentioning that priority users do not wait in the cloud provider queue.

The details of these experiments are shown in Table 9, where the first column, labelled as *Configuration*, represents the percentage of priority users and reserved machines. The second column denotes the number of machines that compose the cloud. The next three columns refer to the number of users that have been attended to, that is, the number of attended priority users (*Pri. Att.*), the number of priority users that have been attended as regular users (*Pri. Reg. Att.*), and the number of regular users attended (*Reg. Att.*). Next, the following three columns show the number of users that have not been attended, that is, the total number of unattended users (*Total Unatt.*), the unattended regular users (*Reg. Unatt.*)

[1] https://www.cs.huji.ac.il/labs/parallel/workload/l_unilu_gaia/index.html



(a) PU:0%, RM:0% (b) PU:10%, RM:10% (c) PU:30%, RM:10% (d) PU:30%, RM:30%

**Fig. 8** Performance evaluation using different percentages of priority users and reserved machines

**Table 9** Summary of the results obtained in *Experiment 2*

| Configuration | # Machines. | Pri. Att. | Pri. Reg. Att. | Reg. Att. | Total Unatt. | Reg. Unatt. | Pri. Unatt. | Total Income |
|---|---|---|---|---|---|---|---|---|
| NP:0, NR:0 | 256 | 0 | 0 | 40450 | 11407 | 11407 | 0 | 44373.3 |
| | 512 | 0 | 0 | 51673 | 184 | 184 | 0 | 62266.6 |
| | 768 | 0 | 0 | 51857 | 0 | 0 | 0 | 65282.7 |
| | 1024 | 0 | 0 | 51857 | 0 | 0 | 0 | 65998.6 |
| NP:10, NR:10 | 256 | 3328 | 54 | 35887 | 12588 | 10690 | 1898 | 44282.0 |
| | 512 | 4302 | 168 | 46406 | 981 | 171 | 810 | 65201.3 |
| | 768 | 5040 | 126 | 46577 | 114 | 0 | 114 | 68773.2 |
| | 1024 | 5233 | 27 | 46577 | 20 | 0 | 20 | 69589.6 |
| NP:30, NR:10 | 256 | 5335 | 1260 | 30693 | 14569 | 5467 | 9102 | 44868.9 |
| | 512 | 7728 | 3141 | 36156 | 4832 | 4 | 4828 | 68527.2 |
| | 768 | 9835 | 5418 | 36160 | 444 | 0 | 444 | 74518.8 |
| | 1024 | 11176 | 4433 | 36160 | 88 | 0 | 88 | 75875.9 |
| NP:30, NR:30 | 256 | 10039 | 53 | 28072 | 13693 | 8088 | 5605 | 46827.3 |
| | 512 | 13299 | 272 | 36026 | 2260 | 134 | 2126 | 70135.4 |
| | 768 | 15227 | 237 | 36160 | 233 | 0 | 233 | 74933.6 |
| | 1024 | 15647 | 11 | 36160 | 39 | 0 | 39 | 75762.0 |

and the unattended priority users (*Pri. Unatt.*). Finally, the last column shows the total income generated by the cloud provider for each cloud configuration (*Total Income*).

After a careful analysis of these results, we observe that – considering the SLAs features – increasing the percentage of reserved machines of the data-centre, has a direct impact on the overall system performance, causing a decrement in the number of unattended priority users and, at the same time, increasing the number of unattended regular users (see first and second row in Table 9). This situation occurs due to the regular users do not have access to the reserved machines and, therefore, the number of available nodes for this type of users is reduced in this configuration. Similarly, increasing the number of priority users in proportion to the number of reserved machines, positively affects the overall performance, which is reflected in a high number of attended priority users and unattended regular users (see first, second and fourth row in Table 9). In this particular case, priority users can be attended as regular users, hence reducing the available nodes for the remaining regular users. However, increasing the number of priority users without considering the reserved machines leads to increasing the number of unattended priority users (see third and fourth row in Table 9).

Regarding the total income, Fig. 9 depicts a summary of the cloud provider profit, which can be calculated using different configurations of physical machines, and percentages of priority users and reserved machines for processing the real-world trace. In this case we observe that using a reduced number of physical machines, the best income is achieved by not using priority users nor reserved machines. On the contrary, when the number of physical machines

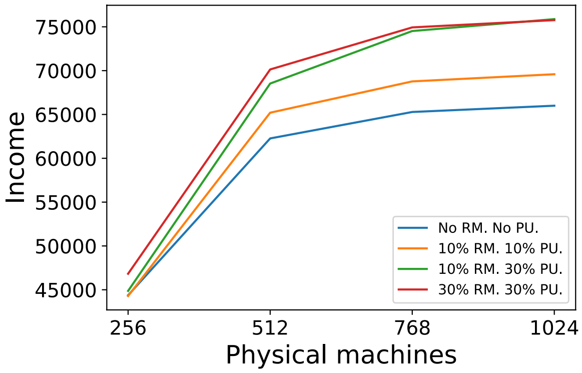
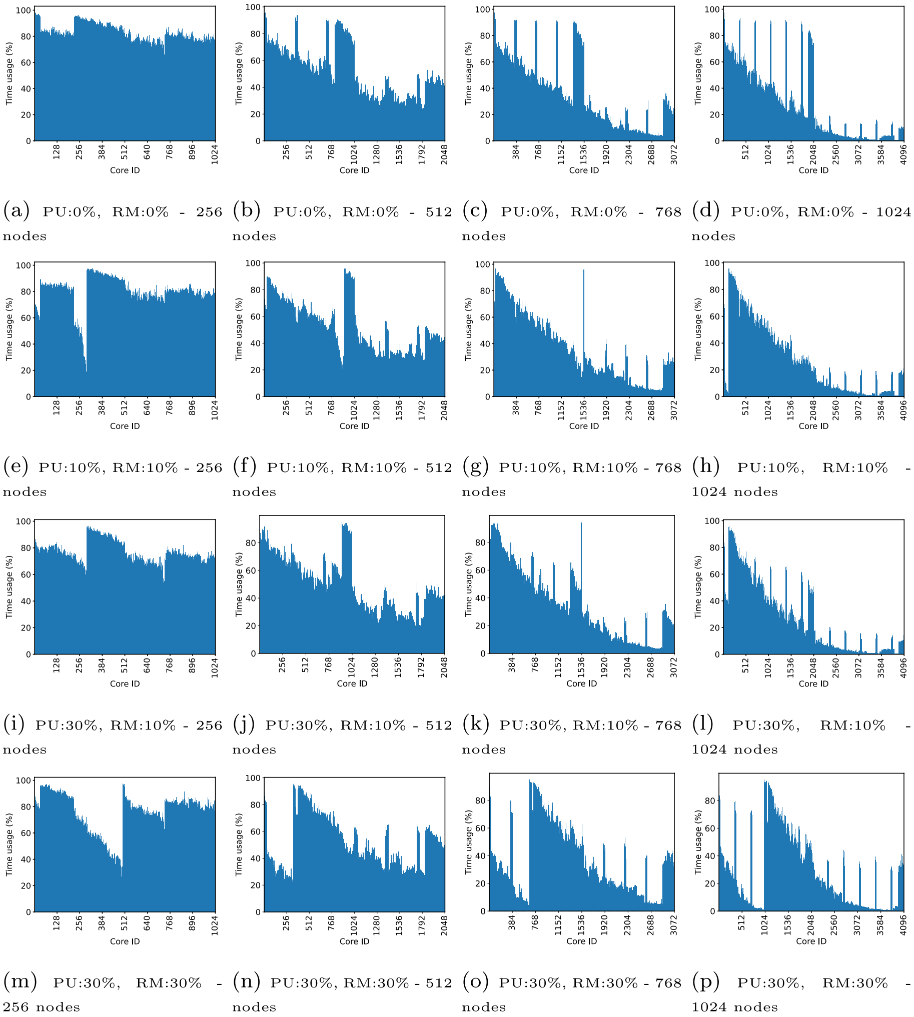


**Fig. 9** Cloud provider income for processing a real-word workload

is increased, and the number of priority users increases in proportion to the number of reserved machines, the results show otherwise, hence achieving higher incomes.

As performed in *Experiment 1*, we have analysed the time usage of each CPU of the data-centre while processing the real trace. Figure 10 shows the CPU usage taking into consideration the four configurations – with different percentages of priority users and reserved machines – designed in this experiment. As expected, increasing the number of physical machines positively impacts on the overall usage of the cloud. Regarding the reserved machines, the greater the ratio between reserved machines and priority users, the lower the percentage of usage of the reserved machines. This fact can be seen in the low peak located in Fig. 10e from CPU 200 to 256. However, when this ratio decreases (increasing the PU and keeping the same number of RM), the usage percentage of reserved machines is increased. In this case, the low peak detected in the previous graph is attenuated, as it can be seen in Fig. 10i.



(a) PU:0%, RM:0% - 256 nodes

(b) PU:0%, RM:0% - 512 nodes

(c) PU:0%, RM:0% - 768 nodes

(d) PU:0%, RM:0% - 1024 nodes

(e) PU:10%, RM:10% - 256 nodes

(f) PU:10%, RM:10% - 512 nodes

(g) PU:10%, RM:10% - 768 nodes

(h) PU:10%, RM:10% - 1024 nodes

(i) PU:30%, RM:10% - 256 nodes

(j) PU:30%, RM:10% - 512 nodes

(k) PU:30%, RM:10% - 768 nodes

(l) PU:30%, RM:10% - 1024 nodes

(m) PU:30%, RM:30% - 256 nodes

(n) PU:30%, RM:30% - 512 nodes

(o) PU:30%, RM:30% - 768 nodes

(p) PU:30%, RM:30% - 1024 nodes

**Fig. 10** Time usage (as %) of each CPU core to process the real-world workload

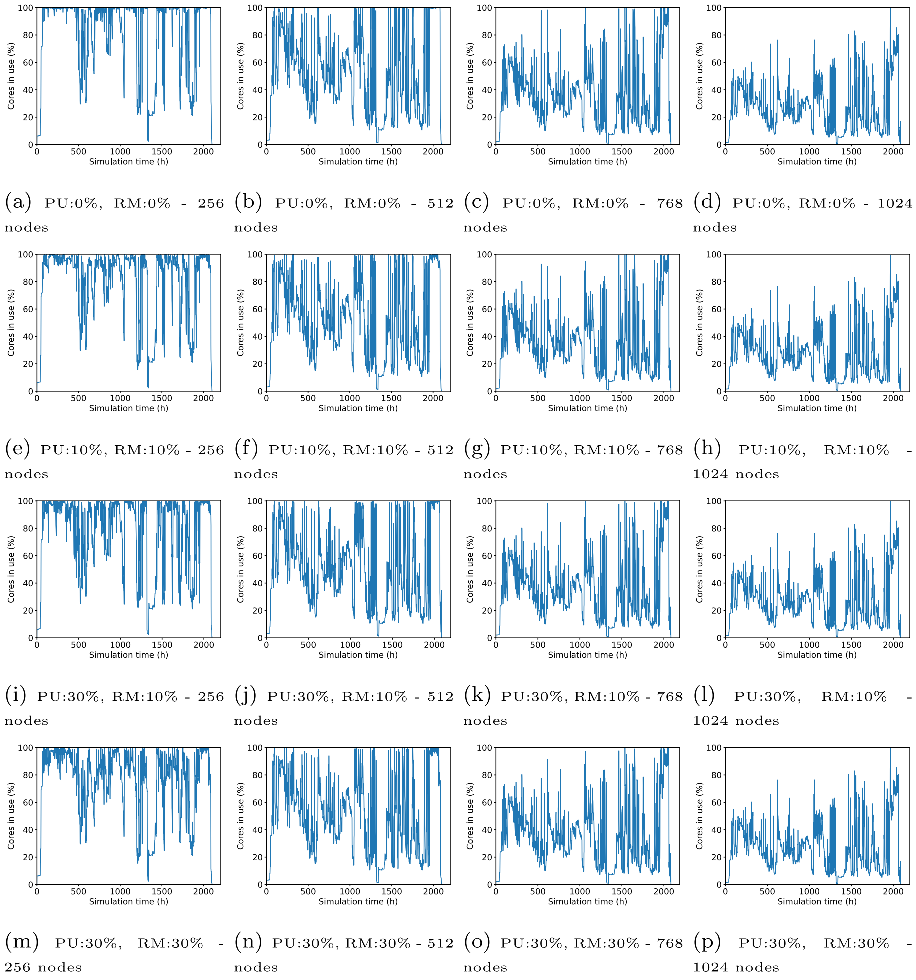
Finally, the percentage of CPU cores used in the datacentre is shown in Fig. 11. As in the previous experiments, increasing the number of nodes causes a decrement in the usage percentage of the cluster. Specifically, we notice that increasing the number of priority users and reserved machines slightly reduces the usage percentage of the platform. This fact can be observed in the first 500 hours, where we can see more prominent saw teeth in the case of priority users. Moreover, this fact is appreciated in the low peaks – among the first 500 hours – of the Fig. 11a, which achieves 35% of usage, while in Fig. 11i the CPU usage is reduced to 25%.

Figure 12 shows a comparison between the CPU usage of Simcan2Cloud (see Fig. 12a), and the one obtained from the GAIA cluster (see Fig. 12b), for processing the trace. In this experiment, we use a data-centre with 640 physical machines, each equipped with a quad-core CPU. The x-axis of these charts show the elapsed time from when the system starts its execution until the trace is fully processed. In general, we appreciate a similar shape in the peaks shown in both charts. For instance, Simcan2Cloud represents the same low peaks that are shown in the behaviour obtained from the real cluster (see low peaks in hours 535, 1036, and 1470). Similarly, high peaks are also represented in the simulated scenario (see high peaks in hours 307, 1256, and 1850).

For the sake of comprehension, we have included additional charts in Section Appendix, which show the results of analysing the behaviour of additional cloud configurations. Nevertheless, we think that the results included in Section "Empirical study" are representative enough to gather sound conclusions.

## Conclusions and future work

In this paper, we have presented Simcan2Cloud, a discrete-event-based framework for modelling and simulating cloud systems. Simcan2Cloud is mainly focused on the cloud provider, supporting the modelling of cloud infrastructures and the interaction of the users with the cloud. Our simulator tool considers monetary costs,



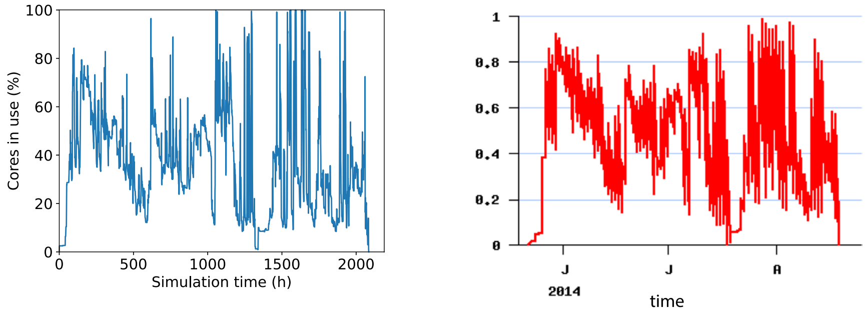
(a) PU:0%, RM:0% - 256 nodes (b) PU:0%, RM:0% - 512 nodes (c) PU:0%, RM:0% - 768 nodes (d) PU:0%, RM:0% - 1024 nodes

(e) PU:10%, RM:10% - 256 nodes (f) PU:10%, RM:10% - 512 nodes (g) PU:10%, RM:10% - 768 nodes (h) PU:10%, RM:10% - 1024 nodes

(i) PU:30%, RM:10% - 256 nodes (j) PU:30%, RM:10% - 512 nodes (k) PU:30%, RM:10% - 768 nodes (l) PU:30%, RM:10% - 1024 nodes

(m) PU:30%, RM:30% - 256 nodes (n) PU:30%, RM:30% - 512 nodes (o) PU:30%, RM:30% - 768 nodes (p) PU:30%, RM:30% - 1024 nodes

**Fig. 11** Percentage of CPU cores used to process the real-world workload



(a) CPU cores used in Simcan2Cloud

(b) CPU cores used in the GAIA cluster

**Fig. 12** Comparison (Simcan2Cloud vs GAIA cluster) focusing on the CPU usage percentage

cloud deployments, a flexible configuration for the VMs offered, and mechanisms for analysing the simulated environments. Furthermore, Simcan2Cloud includes SLAs for modelling two types of users, namely *regular* and *high-priority* users, depending on whether they are willing to wait for the resources they need or not.

A complete case study showing the modelling and evaluation of different cloud scenarios has been presented, and the impact of the parameters considered has been analysed. The main parameters considered were the CPU power provided by the physical machines and the cloud size, that is, the number of physical machines. In this study, several variables, such as the number of users attended to, the average waiting time, the standard deviation, and the minimum waiting time for all the users, were analysed.

The main objective of the empirical study is to assess the suitability of data-centres supporting the cloud for processing a workload, with a strong focus on system performance and scalability. After a thorough analysis of the obtained results, we conclude that the number of machines and the CPU used in the data-centre directly impact the overall system performance. The results clearly demonstrate situations where the data-centre becomes saturated, leading to a significant percentage of users being unable to be attended to. Another interesting parameter is the number of reserved machines (RM). The results clearly indicate that this parameter must be properly configured according to the size of the data-centre to achieve the best results. In addition, we have replicated a trace extracted from the GAIA cluster, a real-world production-ready cluster located at the University of Luxembourg. The results show a similar trend in performance compared to the ones produced by the simulator.

To address the challenge of validating a new simulation platform, we have meticulously designed experiments to explore a wide spectrum of cloud configurations. By varying parameters related to the data-centre, virtualised resources, and user heterogeneity, our experiments provide valuable insights into the system's behaviour across various scenarios. Furthermore, we have successfully conducted an experiment where Simcan2Cloud accurately reproduces a trace from the GAIA cluster, showcasing similar performance trends. In our pursuit of an effective validation approach we plan, as future work, to explore the integration of statistical methods and metamorphic testing (MT) techniques. This involves designing metamorphic relations (MRs) for crucial modules like the user generator, cloud provider, and data-centre, among others. These MRs reflect the underlying properties of the system under test, enabling us to identify scenarios that deviate from expected behaviour due to unfulfilled MRs. Such an approach holds great promise for ensuring the reliability and accuracy of our simulation platform.

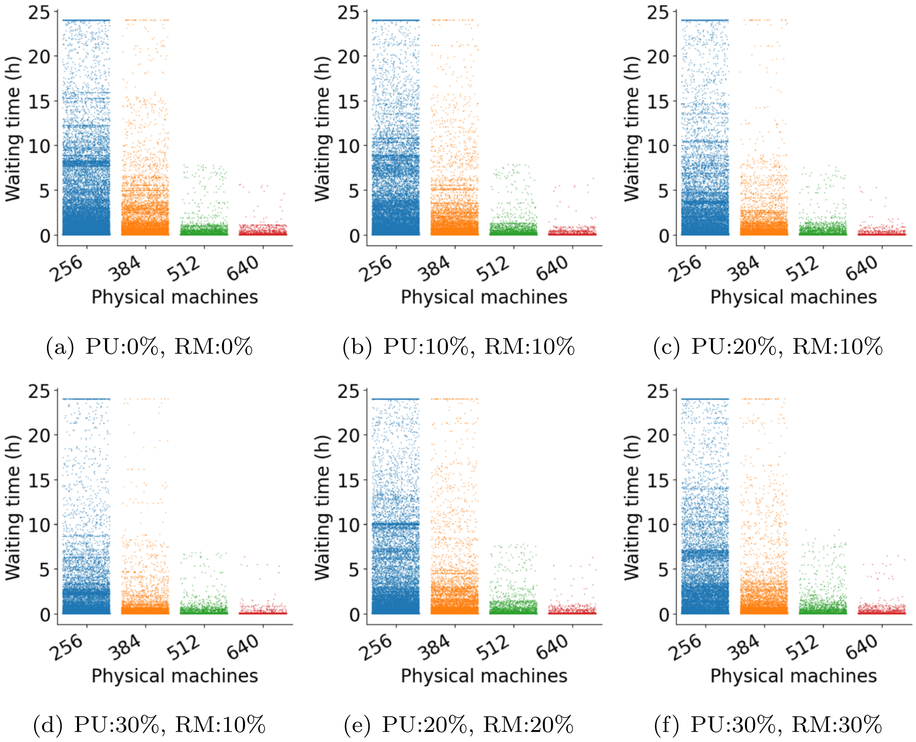
For future work, we are planning an extension of Simcan2Cloud to support one of the latest hot topics in computing: Fog Computing. To that end, we plan to include new modules for supporting a layered distribution of components, such as IoT devices, fog devices, infrastructure monitoring, and IoT applications. Regarding IoT devices, we plan to include sensors and actuators. These devices allows designing a wide variety of *things*, such as hearth monitors, wearables, environmental sensors, and cameras, among others. Fog devices are focused on bridging the gap between the IoT devices and the cloud provider. In this way, the delay in communications will be strongly reduced, which is one of our topics of interests. Moreover, we plan to monitor, in a highly detailed way, the underlying system infrastructure to analyse its scalability. Additionally, IoT application models will be provided to allow the execution of application in the IoT devices.

In addition, we plan to extend the spectrum of possible cloud configurations, increasing the number and CPU power of the physical machines, and also the range of configurations for the hardware, such as the disk space of the hosts. We also intend to analyse the impact of resource costs on the cloud provider's income.

# Appendix

## Appendix A: Graph Appendix

In this appendix, we include an extended set of graphs extracted from the experimental study.



(a) PU:0%, RM:0%

(b) PU:10%, RM:10%

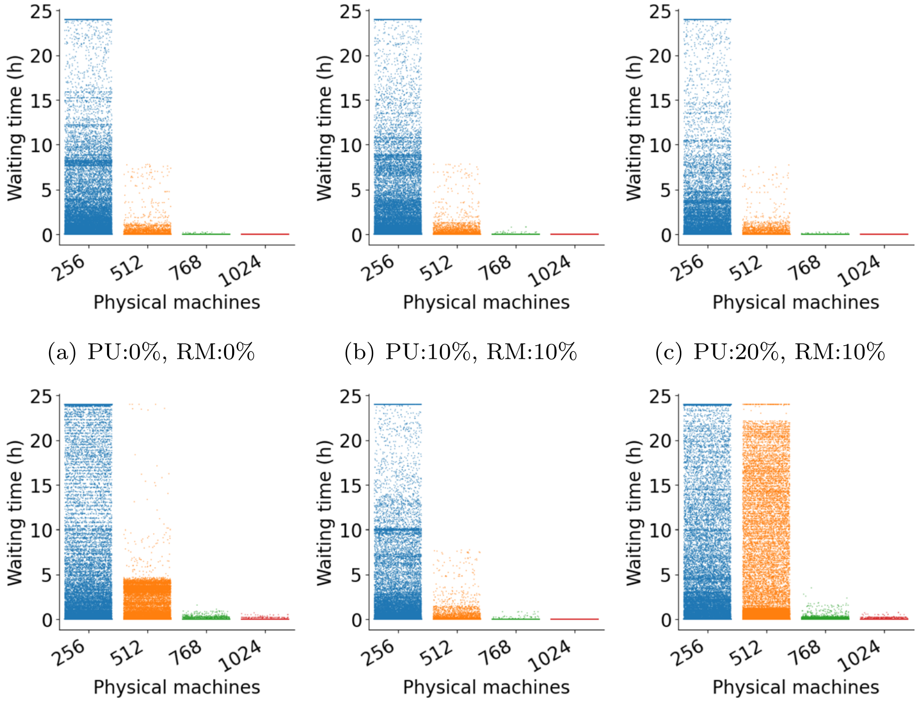
(c) PU:20%, RM:10%

(d) PU:30%, RM:10%

(e) PU:20%, RM:20%

(f) PU:30%, RM:30%

**Fig. 13** Performance evaluation using CPUs@40k MIPS, and different percentages of priority users and reserved machines



(a) PU:0%, RM:0%

(b) PU:10%, RM:10%

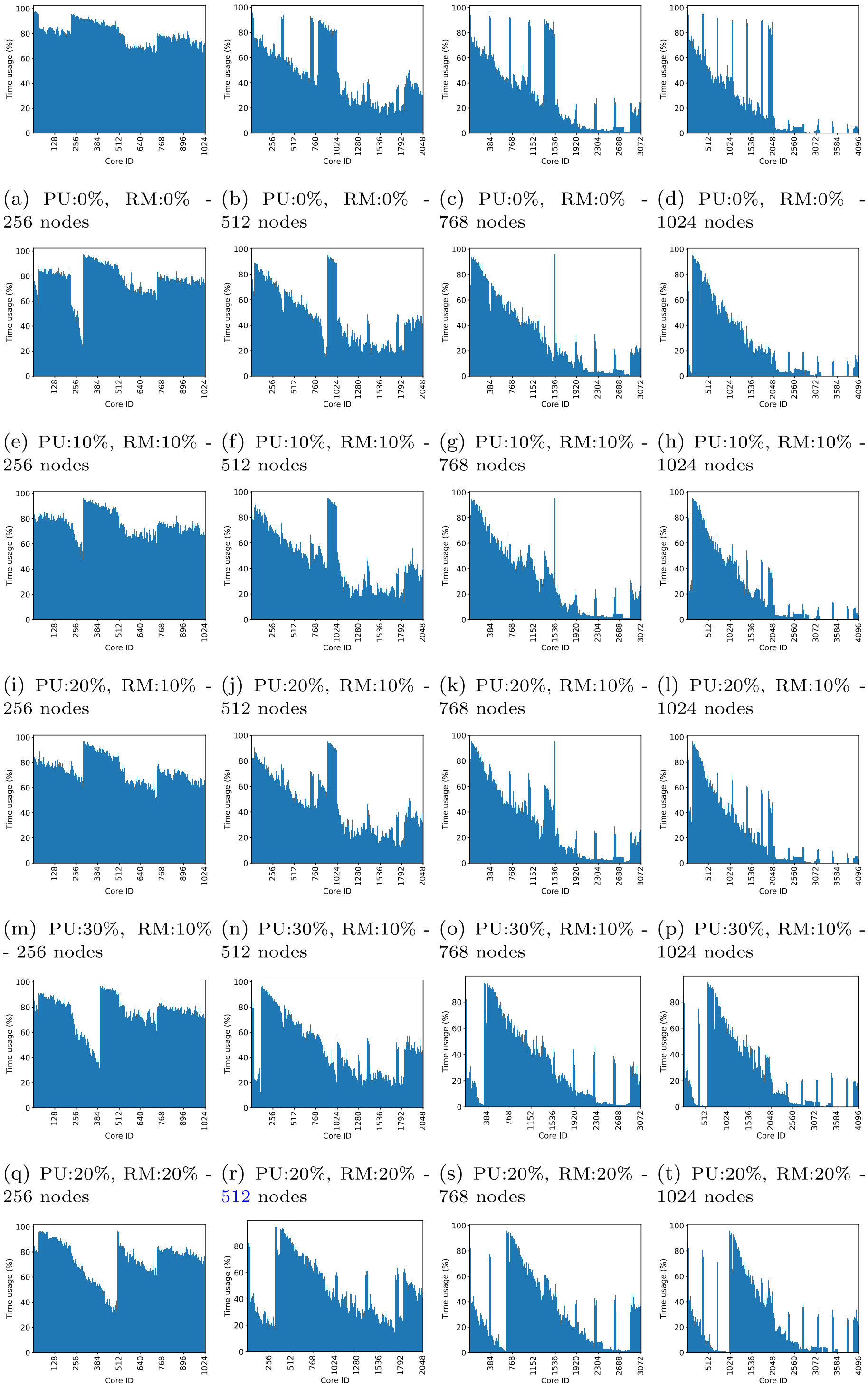
(c) PU:20%, RM:10%

(d) PU:30%, RM:10%

(e) PU:20%, RM:20%

(f) PU:30%, RM:30%

**Fig. 14** Performance evaluation using CPUs@40k MIPS, and different percentages of priority users and reserved machines



(a) PU:0%, RM:0% - 256 nodes

(b) PU:0%, RM:0% - 512 nodes

(c) PU:0%, RM:0% - 768 nodes

(d) PU:0%, RM:0% - 1024 nodes

(e) PU:10%, RM:10% - 256 nodes

(f) PU:10%, RM:10% - 512 nodes

(g) PU:10%, RM:10% - 768 nodes

(h) PU:10%, RM:10% - 1024 nodes

(i) PU:20%, RM:10% - 256 nodes

(j) PU:20%, RM:10% - 512 nodes

(k) PU:20%, RM:10% - 768 nodes

(l) PU:20%, RM:10% - 1024 nodes

(m) PU:30%, RM:10% - 256 nodes

(n) PU:30%, RM:10% - 512 nodes

(o) PU:30%, RM:10% - 768 nodes

(p) PU:30%, RM:10% - 1024 nodes

(q) PU:20%, RM:20% - 256 nodes

(r) PU:20%, RM:20% - 512 nodes

(s) PU:20%, RM:20% - 768 nodes

(t) PU:20%, RM:20% - 1024 nodes

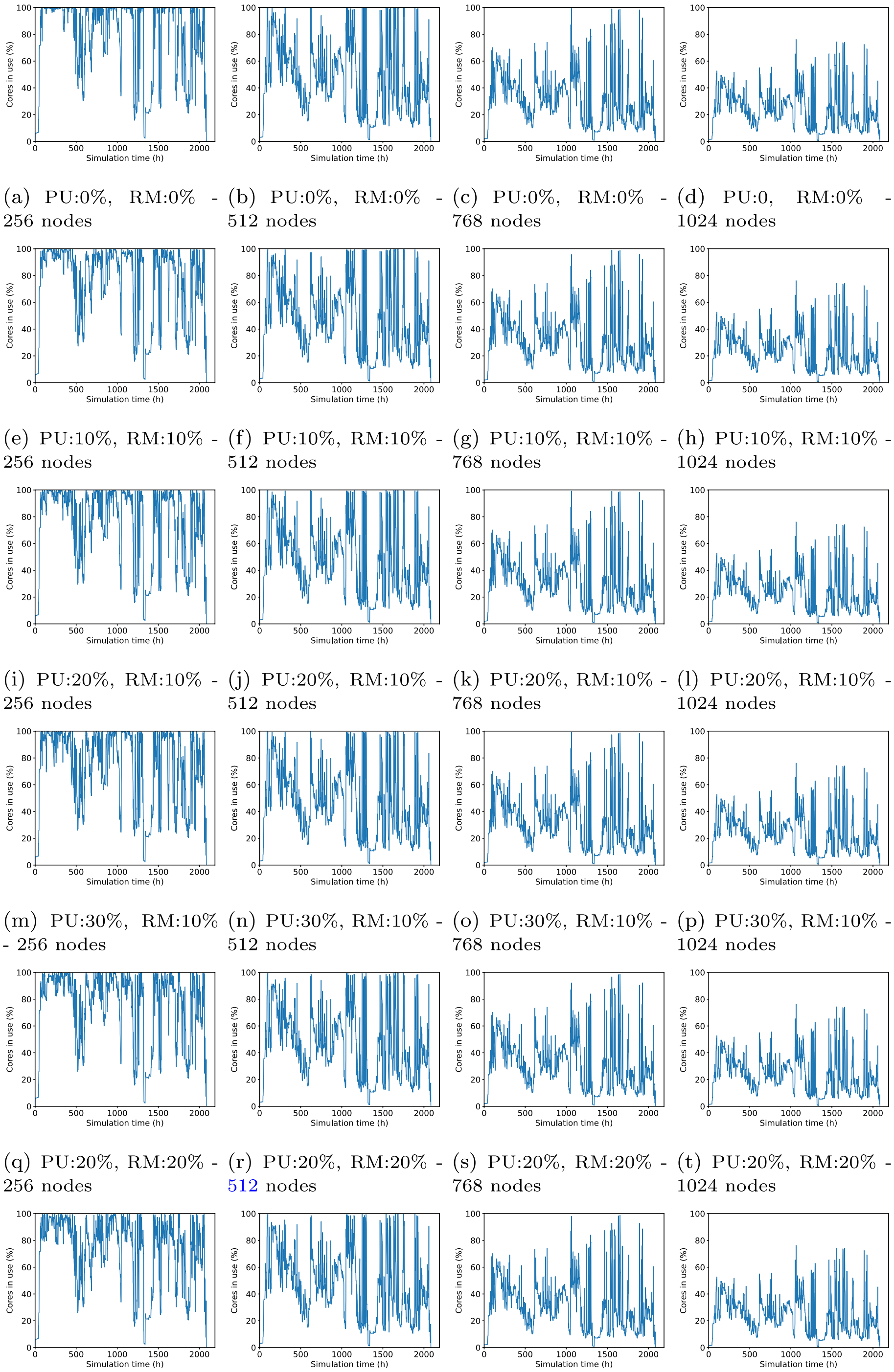
(u) PU:30%, RM:30% - 256 nodes

(v) PU:30%, RM:30% - 512 nodes

(w) PU:30%, RM:30% - 768 nodes

(x) PU:30%, RM:30% - 1024 nodes

**Fig. 15** Time usage (as %) of each CPU core to process the real-world workload



(a) PU:0%, RM:0% - 256 nodes

(b) PU:0%, RM:0% - 512 nodes

(c) PU:0%, RM:0% - 768 nodes

(d) PU:0, RM:0% - 1024 nodes

(e) PU:10%, RM:10% - 256 nodes

(f) PU:10%, RM:10% - 512 nodes

(g) PU:10%, RM:10% - 768 nodes

(h) PU:10%, RM:10% - 1024 nodes

(i) PU:20%, RM:10% - 256 nodes

(j) PU:20%, RM:10% - 512 nodes

(k) PU:20%, RM:10% - 768 nodes

(l) PU:20%, RM:10% - 1024 nodes

(m) PU:30%, RM:10% - 256 nodes

(n) PU:30%, RM:10% - 512 nodes

(o) PU:30%, RM:10% - 768 nodes

(p) PU:30%, RM:10% - 1024 nodes

(q) PU:20%, RM:20% - 256 nodes

(r) PU:20%, RM:20% - 512 nodes

(s) PU:20%, RM:20% - 768 nodes

(t) PU:20%, RM:20% - 1024 nodes

(u) PU:30%, RM:30% - 256 nodes

(v) PU:30%, RM:30% - 512 nodes

(w) PU:30%, RM:30% - 768 nodes

(x) PU:30%, RM:30% - 1024 nodes

**Fig. 16** Percentage of CPU cores used to process the real-world workload

**Table 10** Results obtained when processing the real-world workload using PU:0% and RM:0%

| Machines | Pri. Att. | Pri. as Reg. Att. | Reg. Att. | Total Unatt. | Reg. Unatt. | Pri. Unatt. | Total Income |
|---|---|---|---|---|---|---|---|
| 256 | 0 | 0 | 51322 | 535 | 535 | 0 | 41673.6 |
| 512 | 0 | 0 | 51857 | 0 | 0 | 0 | 55496.6 |
| 768 | 0 | 0 | 51857 | 0 | 0 | 0 | 56299.3 |
| 1024 | 0 | 0 | 51857 | 0 | 0 | 0 | 56382.8 |

**Table 11** Results obtained when processing the real-world workload using PU:10% and RM:10%

| Machines | Pri. Att. | Pri. as Reg. Att. | Reg. Att. | Total Unatt. | Reg. Unatt. | Pri. Unatt. | Total Income |
|---|---|---|---|---|---|---|---|
| 256 | 4519 | 167 | 46094 | 1077 | 483 | 594 | 41818.5 |
| 512 | 5013 | 180 | 46577 | 87 | 0 | 87 | 58318.4 |
| 768 | 5241 | 39 | 46577 | 0 | 0 | 0 | 59429.6 |
| 1024 | 5277 | 3 | 46577 | 0 | 0 | 0 | 59561.4 |

**Table 12** Results obtained when processing the real-world workload using PU:20% and RM:10%

| Machines | Pri. Att. | Pri. as Reg. Att. | Reg. Att. | Total Unatt. | Reg. Unatt. | Pri. Unatt. | Total Income |
|---|---|---|---|---|---|---|---|
| 256 | 7260 | 1615 | 41037 | 1945 | 321 | 1624 | 42833.2 |
| 512 | 9084 | 1217 | 41358 | 198 | 0 | 198 | 60428.6 |
| 768 | 9828 | 669 | 41358 | 2 | 0 | 2 | 61858.0 |
| 1024 | 10223 | 276 | 41358 | 0 | 0 | 0 | 61993.7 |

**Table 13** Results obtained when processing the real-world workload using PU:30% and RM:10%

| Machines | Pri. Att. | Pri. as Reg. Att. | Reg. Att. | Total Unatt. | Reg. Unatt. | Pri. Unatt. | Total Income |
|---|---|---|---|---|---|---|---|
| 256 | 5335 | 1260 | 30693 | 14569 | 5467 | 9102 | 44868.9 |
| 512 | 7728 | 3141 | 36156 | 4832 | 4 | 4828 | 68527.2 |
| 768 | 9835 | 5418 | 36160 | 444 | 0 | 444 | 74518.8 |
| 1024 | 11176 | 4433 | 36160 | 88 | 0 | 88 | 75875.9 |

**Table 14** Results obtained when processing the real-world workload using PU:20% and RM:20%

| Machines | Pri. Att. | Pri. as Reg. Att. | Reg. Att. | Total Unatt. | Reg. Unatt. | Pri. Unatt. | Total Income |
|---|---|---|---|---|---|---|---|
| 256 | 9284 | 230 | 40847 | 1496 | 511 | 985 | 43462.7 |
| 512 | 10221 | 179 | 41358 | 99 | 0 | 99 | 60301.8 |
| 768 | 10480 | 19 | 41358 | 0 | 0 | 0 | 61761.9 |
| 1024 | 10499 | 0 | 41358 | 0 | 0 | 0 | 61993.7 |

**Table 15** Results obtained when processing the real-world workload using PU:30% and RM:30%

| Machines | Pri. Att. | Pri. as Reg. Att. | Reg. Att. | Total Unatt. | Reg. Unatt. | Pri. Unatt. | Total Income |
|---|---|---|---|---|---|---|---|
| 256 | 10039 | 53 | 28072 | 13693 | 8088 | 5605 | 46827.3 |
| 512 | 13299 | 272 | 36026 | 2260 | 134 | 2126 | 70135.4 |
| 768 | 15227 | 237 | 36160 | 233 | 0 | 233 | 74933.6 |
| 1024 | 15647 | 11 | 36160 | 39 | 0 | 39 | 75762.0 |

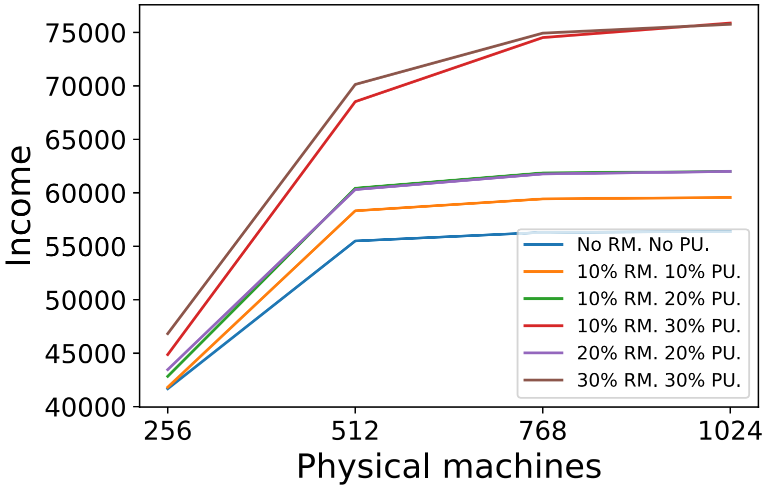


**Fig. 17** Cloud provider income for processing a real-word workload

## Appendix B: Implementation details

This Appendix includes additional information related to the most important components of the Simcan2Cloud architecture: user generator, cloud provider and datacentre. The main classes and features of these components are presented during this section.

Listing 1 shows an excerpt from the *UserGeneratorBase* class. The user instances are parsed from the configuration files and stored in the data structures of the user generation module (see lines 3-4). In addition, there exist other features of the user generation engine that can be configured, such as *allUserArrivesAtOnce*, *startDelay* and *intervalBetweenUsers* (see lines 5-7). These parameters denote the possibility that all the users arrive in the cloud at the beginning of the simulation, the delay time prior to the first user arriving in the cloud, and the time interval between users arriving in the cloud, respectively.

```
 1  class UserGeneratorBase: public CloudManagerBase{
 2      protected:
 3          std::vector<CloudUserInstance*> userInstances;
 4          std::map<std::string, CloudUserInstance*> userHashMap;
 5          bool allUsersArriveAtOnce;
 6          double startDelay;
 7          cPar *intervalBetweenUsers;
 8
 9          virtual void generateUsersBeforeSimulationStarts ();
10          virtual void initialize();
11  }
```

**Listing 1** Excerpt from the *UserGeneratorBase* class

Listing 2 shows the main elements of the base class that represents the cloud provider, namely *CloudProviderBase*. This class contains meta-data for monitoring and managing the resources of the data-centres (see line 4). This meta-data – parsed from the configuration file – is used to locate the most suitable data-centre to allocate the VMs requested by the users (see line 6). The main objective of this class is to process the requests from the users – forwarding them to the data-centres – and manage the user subscriptions.

```
1  class CloudProviderBase : public CloudManagerBase{
2
3      protected:
4          std::vector<DataCentre*> dataCentresBase;
5
6          int parseDataCentresList();
7  };
```

**Listing 2** Excerpt from the *CloudProviderBase* class

```
1   class DataCentreManager: public CloudManagerBase{
2       portected:
3           std::vector<DataCentre*> dataCentresBase;
4
5       public:
6           Hypervisor* selectNode (SM_UserVM*& userVM_Rq,
7                                   const VM_Request& vmRequest);
8           int getNTotalAvailableCores() const;
9           int getNTotalCores() const;
10  };
```

**Listing 3** Excerpt from the *DataCentreManager* class

```
1   class DataCentre{
2       private:
3           std::string name;
4           std::vector <Rack*> computingRacks;
5           std::vector <Rack*> storageRacks;
6
7       public:
8           void addRack (Rack* rack, bool isStorage);
9           Rack* getRack (int index, bool isStorage);
10          int getNumRacks (bool isStorage);
11  };
```

**Listing 4** Excerpt from the *DataCentre* class

Listing 3 shows the main elements of the data-centre manager, which represent the data-centre meta-data (see line 3), the method for choosing the machines on which to allocate the requests (see line 6), as well as the methods for obtaining information (see lines 7 and 8). The integration of new data-centre managers can be performed by creating a new class that inherits from *DataCentreManagerBase*, which must overwrite the method *selectNode* (see line 6) containing the new resource scheduling policy. Currently, Simcan2Cloud incorporates two different data-centre managers, namely *DataCentreManagerFirstFit* and *DataCentreManagerBestFit*.

### Abbreviations

| | |
|---|---|
| API | Application Programming Interface |
| AWS | Amazon Web Services |
| CPU | Central Processing Unit |
| EC2 | Elastic Compute Cloud |
| GB | Gigabyte |
| Gbps | Gigabits per second |
| GUI | Graphical User Interface |
| I/O | Input/Output |
| MB | Megabyte |
| Mbps | Megabits per second |
| MIPS | Millions Instructions per second |
| MIs | Millions Instructions |
| PU | Priority users |
| RAM | Random Access Memory |
| RM | Reserved machines |
| SLAs | Service Level Agreement |
| SoTA | State of The Art |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| VM | Virtual Machine |

### Acknowledgements

Not applicable.

### Authors' contributions

Pablo C. Cañizares authored, reviewed drafts of the paper, prepared figures and tables, and approved the final draft. Alberto Núñez conceived and designed the experiments, authored and reviewed drafts of the paper, and approved the final draft. Adrián Bernal performed the experiments, prepared figures and tables, and approved the final draft. María-Emilia Cambronero analysed the data, authored and reviewed drafts of the paper, and approved the final draft. Adam barker analysed the data, reviewed drafts of the paper, and approved the final draft.

### Authors' information

Not applicable.

### Funding

This work was supported by the Spanish MINECO/FEDER project under grants PID2021-122270OB-I00, TED2021-129381B-C21 and PID2019-108528RB-C22, the Comunidad de Madrid project FORTE-CM under grant S2018/TCS-4314, project S2018/TCS- 4339 (BLOQUES-CM) co-funded by EIE Funds of the European Union and Comunidad de Madrid, 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, and the University of Castilla-La Mancha (cofunded with FSE funds, EU) under the announcement 2018/12504 published in the DOCM.

### Availability of data and materials

The data and materials are available from the corresponding author on reasonable request. The source code is available at GitHub: https://github.com/PabloCCanizares/Simcan2Cloud.

## Declarations

### Ethics approval and consent to participate

Not applicable.

### Competing interests

The authors declare no competing interests.

Received: 16 January 2023 Accepted: 14 August 2023
Published online: 18 September 2023

### References

1. Flexera (2019) RightScale 2019 State of the Cloud Report. Tech. rep
2. Perumal K, Mohan S, Frnda J, Divakarachari PB (2022) Dynamic resource provisioning and secured file sharing using virtualization in cloud azure. J Cloud Comput Adv Syst Appl 11(46):1–12
3. Oren T, Yilmaz L (2012) Synergies of simulation, agents, and systems engineering. Expert Syst Appl 39(1):81–88
4. Khani H, Khanmirza H (2019) Randomized routing of virtual machines in IaaS data centers. PeerJ Comput Sci 5:211
5. Arzymatov K, Sapronov A, Belavin V, Gremyachikh L, Karpov M, Ustyuzhanin A, Tchoub I, Ikoev A (2020) SANgo: A storage infrastructure simulator with reinforcement learning support. PeerJ Comput Sci 6:271
6. Mansouri N, Ghafari R, Zade B (2020) Cloud computing simulators: A comprehensive review. Simul Model Pract Theory 104(102):144
7. Schad J, Dittrich J, Quiané-Ruiz JA (2010) Runtime measurements in the cloud: observing, analyzing, and reducing variance. VLDB Endowment 3:460–471
8. Calheiros RN, Ranjan R, Beloglazov A, Rose CAFD, Buyya R (2011) Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Experience 41(1):23–50
9. Castañé G, Núñez A, Llopis P, Carretero J (2013) E-mc: A formal framework for energy modelling in cloud computing. Simul Model Pract Theory 39:56–75
10. Kecskemeti G (2015) DISSECT-CF: A simulator to foster energy-aware scheduling in infrastructure clouds. Simul Model Pract Theory 58:188–218
11. Rossini A, Kritikos K, Nikolov N, Domaschka J, Griesinger F, Seybold D, Romero D, Orzechowski M, Kapitsaki G, Achilleos A (2017) The cloud application modelling and execution language (CAMEL). Ulm University, Tech. rep
12. Dimitri N (2020) Pricing cloud IaaS computing services. J Cloud Comput Adv Syst Appl 9(14):1–11
13. Sun X, Wang Z, Wu Y, Che H, Jiang H (2021) A price-aware congestion control protocol for cloud services. J Cloud Comput Adv Syst Appl 10(55):1–15
14. Binz T, Breitenbücher U, Kopp O, Leymann F (2014) TOSCA: Portable automated deployment and management of cloud applications. Advanced Web Services. Springer, New York, pp 527–549
15. Silva G, Rose L, Calinescu R (2014) Cloud DSL: A Language for Supporting Cloud Portability by Describing Cloud Entities. In: 2nd International Workshop on Model-Driven Engineering on and for the Cloud, CloudMDE'14. Valencia, Spain, CEUR Workshop Proceedings, 1242;36–45.
16. Guillén J, Miranda J, Murillo J, Canal C (2013) A UML Profile for modeling multicloud applications. In: 2nd European Conference on Service-Oriented and Cloud Computing, Springer, ESOCC'13, pp 180–187
17. Fakhfakh F, Kacem HH, Kacem AH (2017) Simulation tools for cloud computing: A survey and comparative study. In: 16th International Conference on Computer and Information Science, ICIS'17. IEEE Wuhan, China, pp 221–226
18. Byrne J, Svorobej S, Giannoutakis K, Tzovaras D, Byrne P, Östberg PO, Gourinovitch A, Lynn T (2017) A review of cloud computing simulation platforms & related environments. In: 7th International Conference on Cloud Computing and Services Science, CLOSER'17. ACM, Porto Portugal, pp 651–663
19. Bhatia M, Sharma M (2016) A critical review & analysis of cloud computing simulators. Int J Latest Trends Eng Technol 1:29–36
20. Singh R, Patel P, Singh P (2015) Cloud simulators: A review. Int J Adv Comput Electron Technol 2(2):62–67
21. Zhao W, Peng Y, Xie F, Dai Z (2012) Modeling and simulation of cloud computing: A review. In: 2012 IEEE Asia Pacific Cloud Computing Congress (APCloudCC). IEEE, Shenzhen, China, pp 20–24
22. Malik AW, Bilal K, Aziz K, Kliazovich D, Ghani N, Khan SU, Buyya R (2014) CloudNetSim++: A toolkit for data center simulations in OMNeT++. In: 11th Annual High Capacity Optical Networks and Emerging/Enabling Technologies (Photonics for Energy). IEEE, Charlotte, NC, USA, pp 104–108
23. Keller G, Tighe M, Lutfiyya H, Bauer M (2013) DCSim: A data centre simulation tool. In: 2013 IFIP/IEEE International Symposium on Integrated Network Management, IEEE, IM'13, pp 1090–1091

24. Fernández-Cerero D, Fernández-Montes A, Jakóbik A, Kołodziej J, Toro M, (2018) Score: Simulator for cloud optimization of resources and energy consumption. Simul Model Pract Theory 82:160–173. https://doi.org/10.1016/j.simpat.2018.01.004, https://www.sciencedirect.com/science/article/pii/S1569190X18300030
25. Fernández-Cerero D, Jakóbik A, Fernández-Montes A, Kołodziej J (2019) GAME-SCORE: Game-based energy-aware cloud scheduler and simulator for computational clouds. Simul Model Pract Theory 93:3–20
26. Núñez A, Vázquez-Poletti JL, Caminero AC, Castañé GG, Carretero J, Llorente IM (2012) iCanCloud: A flexible and scalable cloud infrastructure simulator. J Grid Comput 10(1):185–209
27. Garg SK, Buyya R (2011) Networkcloudsim: Modelling parallel applications in cloud simulations. In: 2011 Fourth IEEE International Conference on Utility and Cloud Computing. IEEE, Melbourne, VIC, Australia, pp 105–113
28. Wickremasinghe B, Calheiros RN, Buyya R (2010) Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications. IEEE, Perth, WA, Australia, pp 446–452
29. Fittkau F, Frey S, Hasselbring W (2012) Cdosim: Simulating cloud deployment options for software migration support. In: 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems, MESOCA'12. IEEE Computer Society, Trento, Italy, pp 37–46
30. Chen W, Deelman E (2012) Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In: 8th International Conference on E-Science, eScience'12. IEEE, Chicago, IL, USA, pp 1–8
31. Jararweh Y, Jarrah M, kharbutli M, Alshara Z, Alsaleh MN, Al-Ayyoub M, (2014) Cloudexp: A comprehensive cloud computing experimental framework. Simul Model Pract Theory 49:180–192. https://doi.org/10.1016/j.simpat.2014.09.003, https://www.sciencedirect.com/science/article/pii/S1569190X14001464
32. Sqalli MH, Al-saeedi M, Binbeshr F, Siddiqui M (2012) Ucloud: A simulated hybrid cloud for a university environment. In: 1st International Conference on Cloud Networking, CLOUDNET'12. IEEE, Paris, France, pp 170–172
33. Varga A (2010) OMNeT++. Modeling and Tools for Network Simulation. Springer, Berlin, Heidelberg, pp 35–59
34. Baumgart I, Heep B, Krause S (2009) OverSim: A scalable and flexible overlay framework for simulation and real network applications. In: 9th International Conference on Peer-to-Peer Computing, IEEE, P2P'09, pp 87–88
35. TD Nguyen, EN Huh (2018) ECSim++: An INET-Based Simulation Tool for Modeling and Control in Edge Cloud Computing. In: IEEE International Conference on Edge Computing, EDGE'18. IEEE, San Francisco, California, USA, pp 36–45
36. Qayyum T, Malik A, Khattak MK, Khalid O, Khan S (2018) FogNetSim++: A toolkit for modeling and simulation of distributed fog environment. IEEE Access 6:63,570–63,583
37. Amazon (2021) Amazon Elastic Compute Cloud. Web page at http://aws.amazon.com/ec2/. Accessed Mar 26 2021

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.