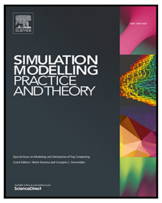
Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

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



Modeling and simulation for cost optimization and performance analysis of transactional applications in hybrid clouds[☆]

Joaquín Entrialgo [*], Manuel García, José Luis Díaz, Javier García, Daniel F. García

Department of Computer Science, University of Oviedo, Campus de Viesques, 33204, Gijón, Spain

A R T I C L E I N F O

Keywords:
Hybrid clouds
Cost optimization
Performance analysis
Virtual machine allocation
Simulation

A B S T R A C T

In the design process of a hybrid cloud with the purpose of cost optimization, finding the optimal distribution between the resources to be deployed in both private and public clouds is a complex task. To help the cloud designer in this task, optimizer and simulation tools can be used. In the research presented in this paper, a state-of-the-art optimizer tool, designed for the optimal allocation of virtual machines in public clouds for cost optimization, is analyzed for its use in hybrid cloud scenarios. In addition, based on the system and workload model implemented by this tool, a simulator is developed and presented. The simulator provides detailed run-time performance information corresponding to the virtual machine allocations generated by the optimizer tool, offering essential analysis capabilities to the cloud designer. A motivating scenario is used to show how the combined utilization of the optimizer and simulation tools can offer fundamental help in the design and configuration process of hybrid clouds. The research questions posed in the motivating scenario are addressed in a set of experiments, which provide meaningful insights into the capabilities of the tools. The tools and all the experimental data carried out in this research have been made publicly available.

1. Introduction

Hybrid clouds combine private clouds (supported by on-premises computing infrastructures) with cloud computing resources offered by public providers, such as Microsoft Azure and Amazon Web Services. In this way, hybrid clouds can provide the benefits of both worlds. While private clouds can take advantage of available on-premises computing infrastructures, as well as offering secure locations for critical data and applications, public clouds make the most of the usual benefits provided by public cloud providers, such as the scalability of resources, the pay-as-you-go cost model, and the multi-region service deployments.

In the field of transactional applications, a common scenario for a hybrid cloud is using a private cloud to support the base workload of the applications and a public cloud configured to support sporadic workload spikes [1,2]. In the design process of a hybrid cloud to support a set of applications, both private and public clouds must be dimensioned and configured to guarantee the required performance of the applications with a minimum cost. Finding the optimal balance between the resources deployed in the private and public clouds of a hybrid infrastructure is not an easy task. If the private cloud is overdimensioned, the average utilization of the computing resources of the private cloud may be too low, wasting resources. However, if the private cloud is not dimensioned with enough resources, to compensate for this shortage, more resources must be acquired in the public cloud. In both

[☆] This work was supported by the Spanish National Plan for Research, Development and Innovation [RTI2018-094849-B-I00].
[*] Corresponding author.
E-mail addresses: joaquin@uniovi.es (J. Entrialgo), mgarcia@uniovi.es (M. García), jldiaz@uniovi.es (J.L. Díaz), javier@uniovi.es (J. García), dfgarcia@uniovi.es (D.F. García).

https://doi.org/10.1016/j.simpat.2021.102311
Received 5 October 2020; Received in revised form 5 March 2021; Accepted 6 March 2021

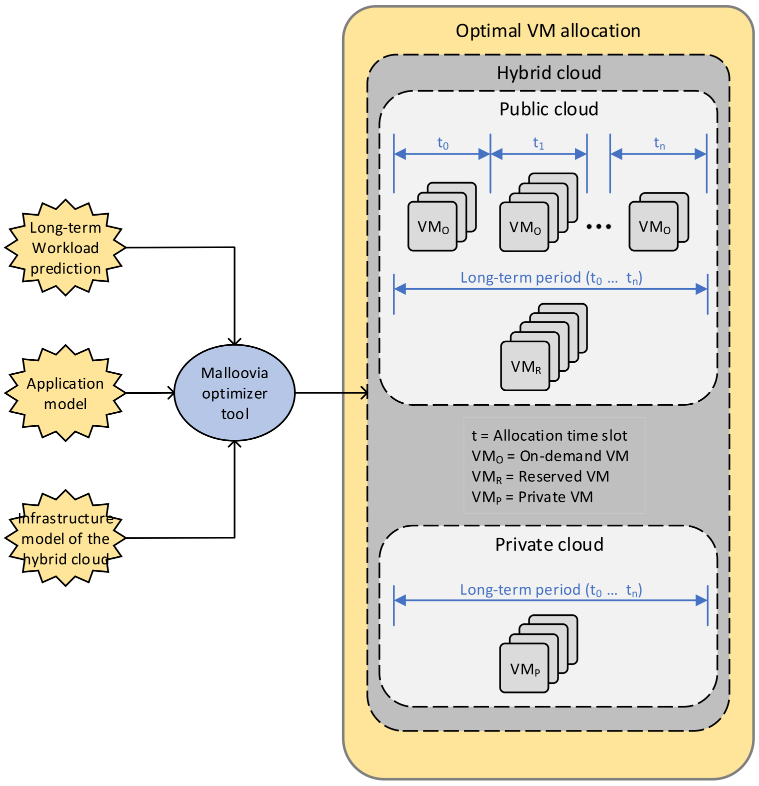


Fig. 1. Malloovia VM allocation strategy in a hybrid cloud scenario.

cases, the cost of the application deployment will not be optimal. In the design process of hybrid clouds, in order to help designers with cost optimization, optimizer and simulation tools can be used.

In previous research, Díaz et al. [3] and Entrialgo et al. [4], developed the Malloovia optimizer tool for the cost optimization of transactional application deployments in public clouds using two categories of Virtual Machines (VMs): on-demand and reserved (such as those offered by major IaaS platforms, like Amazon EC2 [5], Azure Virtual Machines [6] and Google Compute Engine [7]). In this paper, the use of Malloovia for the cost optimization of hybrids clouds is analyzed. To this end, both public and private clouds must be taken into account. The private category of VM is added to the Malloovia model (in addition to the on-demand and reserved categories), and the private clouds are modeled as collections of private VMs, whose characteristics, costs and limits must be determined by the cloud designer.

Malloovia implements a VM allocation strategy, designed for a long-term period (usually one year, to take advantage of reserved VMs), whose use in a hybrid cloud scenario is shown in Fig. 1. The inputs of Malloovia are two models (one for the applications and one for the infrastructure) and a long-term workload prediction, and its output is a sequence of VM allocations. An allocation time slot, represented as t in Fig. 1, is established to determine the periods at which VM allocations must be generated. This allocation time slot is normally one hour. Malloovia finds the optimal combination of private, reserved and on-demand VMs. The numbers of private and reserved VMs are fixed for the whole long-term period. However, the number of on-demand VMs is variable, calculated for each time slot in the period. In this way, the private and reserved VMs support the base workload of the applications, while the on-demand VMs provide computational power to support the spikes of the application workloads. Although not represented in the figure, not only the number of VMs are computed, but also their type. An example of a VM type in the Amazon IaaS Platform (EC2) is m5.xlarge [8].

The allocation strategy implemented by Malloovia is designed to support a request rate for a set of applications. That is, Malloovia calculates the optimal number and types of VMs that can process the required number of requests for each application in each allocation time slot, at minimum cost. For example, if an application has to serve 1000 requests in a determined time slot, the VM allocation calculated by Malloovia guarantees enough computational power to process those 1000 requests in that time slot. However, Malloovia can neither establish any guarantee on the response times for the application requests, nor compute the response time values. Furthermore, it cannot calculate the utilization of computational resources. In order to overcome these shortcomings of Malloovia, a simulation tool, called Simlloovia, has been designed, and is presented in this paper.

Fig. 2 summarizes the use of the Malloovia and the Simlloovia tools in the cost optimization procedure of a hybrid cloud, to support the workloads of a set of applications over a long-term period. The Malloovia optimizer tool calculates an optimal VM allocation to support the workload, minimizing cost. Then, Simlloovia is used to obtain additional performance data. As shown in

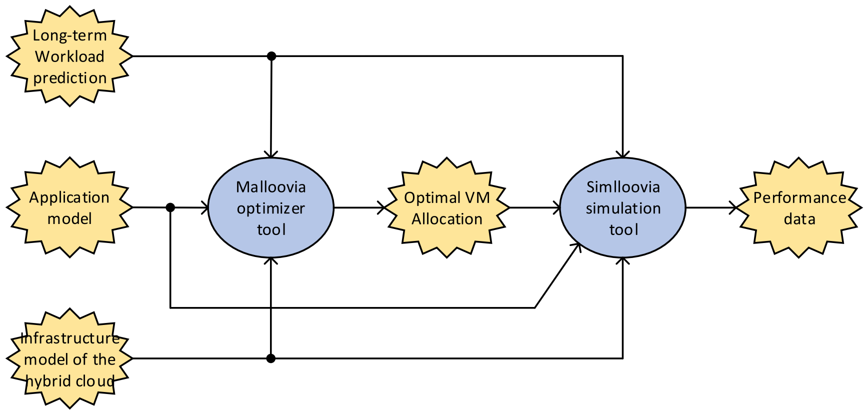

Long-term Workload prediction
Application model
Infrastructure model of the hybrid cloud
Malloovia optimizer tool
Optimal VM Allocation
Simlloovia simulation tool
Performance data

Fig. 2. Use of the Malloovia and the Simlloovia tools in the cost optimization and performance analysis of application deployments in a hybrid cloud.

the figure, as input Simlloovia uses the same application and infrastructure models as Malloovia, as well as the same long-term workload prediction. In addition, Simlloovia is also fed with the sequences of VM allocations generated by Malloovia. With all this information, Simlloovia computes the required performance data, most importantly, the response times for the application requests and the utilization of resources.

The capabilities of the Malloovia and the Simlloovia tools are demonstrated in a motivating scenario, presented in Section 2. The scenario describes an organization that currently uses a private cloud to support its applications and wants to study how to deal with an expected increment in its workload for the next year. The suitability of expanding its private cloud with a hybrid infrastructure to achieve a cost-effective deployment of its applications is analyzed. The analysis of this scenario is presented in the section of Experimental Results. This motivating scenario highlights the great capabilities of Malloovia and Simlloovia to deal with the complex scenarios of cost optimization and performance analysis in hybrid cloud infrastructures.

The main contributions of the research presented in this paper are the following:

- Analysis of how the Malloovia VM cost optimization strategy can be used in hybrid clouds.
- Development of a simulator, based on the Malloovia VM allocation model, designed for the performance analysis of transactional applications deployed in the cloud. To the best of our knowledge, this simulator is the first of its kind specifically designed for this type of application.
- Publication of the simulator software, as well as all the experimental data, including data traces, so that the experimentation carried out in this research can be reproduced in other research works.

The rest of this paper is organized as follows. The motivating scenario is presented in Section 2. Section 3 discusses the related work. The Malloovia mathematical model is summarized in Section 4. In Section 5, the architecture and operation of the Simlloovia simulator, as well as its simulation model, are presented. Section 6 describes a set of experiments, related to the motivating scenario, to show how the combined use of Malloovia and Simlloovia can help cloud designers in the cost optimization and performance analysis of hybrid clouds. Finally, Section 7 provides the main conclusions of the paper.

2. Motivating scenario

As a motivating scenario, let us consider a company that has a private data center where four applications are executed in a private cloud. The company is studying how to handle the workload for the next year. Continuing with the trends seen this year, it is expecting a significant increase in workload and the company wants to obtain a cost-effective combination of private and public infrastructure to handle it.

The private infrastructure of the company is limited as the data center has space restrictions that limit the number of new VMs that can be added. As there is a significant base workload level throughout the year, the company is interested in determining whether one-year reserved instances in public infrastructure would be cost-effective, and how many on-demand instances would be required to handle the peaks in the workload when the capacity of the private and the reserved infrastructure is not sufficient.

The company has decided to use Amazon Web Services (AWS) in the regions of London or Ireland because they are close to its customers. It will only consider instances in the c5 family because they are optimized for computation and the workload is computationally heavy.

Given this situation, the following questions are raised:

- **Q1** What would be the optimal allocation for the next year from the point of view of cost? Obtaining the optimal allocation means computing how many new VMs are to be added to the private data center and how many instances (both reserved and on-demand) are to be hired from AWS.

- **Q2** What would the cost of this allocation be?
- **Q3** What would the utilization of the VMs be? This will give the company insight into the amount of resources wasted and how close the infrastructure is to its limits.
- **Q4** What would the response times be? This will inform the company of the quality of service (QoS) obtained with the allocation.
- **Q5** How much would changing the capacity provided affect the response time and the cost? It is expected that adding more capacity reduces the response time but increases the cost, and vice versa. The company wants to try different configurations in order to assess this trade-off.

Using Malloovia, the corporation can answer questions Q1 and Q2, but in order to answer Q3, Q4 and Q5, a simulation with Simlloovia must be carried out. The results are presented in Section 6.

3. Related work

In recent years, cloud computing has become a fundamentally enabling technology for many IT platforms, with hybrid cloud solutions increasingly adopted [9,10]. The research effort carried out in hybrid cloud computing solutions is extensive. In this work we deal with the IaaS services optimization problem and within this topic we focus on the users' perspective more than the providers'. The review of the literature is divided into four subsections. The first presents a brief look at cost optimization in hybrid clouds. The second covers the cost optimization techniques more directly related with our work. Since the results obtained with our solution are tested by simulation, in the third subsection the cloud simulators available are commented. As the simulator developed is coded in Python, the final subsection is devoted to the simulators specifically developed in this language.

3.1. Optimization problem in hybrid clouds

According to the most recent state of the cloud report [10], 74% of responders used a hybrid combination of public and private clouds. This solution adopts diverse forms and solves different challenges depending on how the cloud is used. Some of the challenges to overcome are: quality of service (QoS), reliability, availability, possible vendor lock-in, and especially, cost. All these characteristics make hybrid cloud optimization a very diverse problem. In this review we focus on only one of these characteristics: cost optimization.

The main distinction among the works in the literature is the kind of workload they deal with as their optimization target.

The most common kind of workload is called “Bag of Tasks” (BoT), some examples of which are image processing, data mining and big data analytics. This kind of workload is represented by applications composed of several tasks that must be finished to complete the application. The tasks are normally taken as independent and can be executed in any order. Some examples of works dealing with the deployment of BoT in hybrid clouds are [11–15] and [16].

In [11], Malawski et al. present a model that assumes multiple heterogeneous providers that offer computational and storage cloud services. Their optimization objective is to reduce the total cost under deadline constraints. The approach presented by Balagoni et al. [12] represents the public cloud as a group of heterogeneous instances with uncertainty in their performances. This work is unique in that it uses spot instances instead of on-demand instances. Abdi et al. [13] analyze a mix of a private cloud and a cloud federation. Taking into account this infrastructure, they optimize the execution of BoT applications under deadline restrictions. In [14], Pelaez et al. propose a scheduling algorithm that minimizes the cost while maximizing the number of deadlines met. The algorithm can work autonomously by using sampled observations of previous processing times. Stavrinides and Karatza [15] present a model and the heuristics for the scheduling of real-time BoTs and cost minimization. This work makes a distinction between BoTs depending on the contents they use, and as a result, some tasks are not allowed to migrate to the public cloud. Finally, Naik et al. [16] describe their solution, called Adaptive Multiobjective Resource Selection Model (AMORSM), based on genetic algorithms. With this method the authors meet multiple conflicting objectives such as minimum execution time, reduction in execution costs and energy consumption.

The second most frequent workload in hybrid clouds is called “Workflows”. They consist of tasks with precedence constraints between them, and are normally represented by a directed acyclic graph (DAG), where the nodes represent the tasks, and the edges, the relationships between them. A few examples of cost optimization works with workflows deployed in hybrid cloud solutions are [17,18] and [19].

In [17], Lin et al. propose an online-scheduling strategy that aims to complete the maximum number of deadline-constrained applications at the lowest possible price. Their solution is composed of several algorithms to manage the time constraints among the tasks and carry out an application partition procedure. A more practical model for cost optimization is presented in Liang et al. [18]. In their model the private cloud is represented by its physical computing resources, which can support only defined VM configurations. Another important characteristic of this work is that it contemplates sustained-use discounts in the billing process. The last example is represented by the work of Stavrinides and Karatza [19], where the workflow is tackled in a fog environment. The fog elements take the role of the private cloud that can send tasks to the public cloud; their strategy is based on a trade-off between performance and cost.

A comprehensive revision of workload scheduling and resource provisioning in hybrid cloud environments can be found in Wang et al. [20].

Table 1
Comparison of our work with the most closely related papers.

	Application types	Multi-Cloud	Multiple VM types	VM limits	Reserved VM	On-demand VM	Private cost	Public VM cost	Other costs	Simulator	Optimization
Li 2011 [21]	○	○	○	○	○	○	○	○	●	○	Trace-driven
Bjorkqvist 2012 [22]	●	○	◐	○	○	●	●	●	○	●	Algo.[a]
Kaviani 2014 [23]	◐	○	○	○	○	●	○	●	●	○	BIP[b]
Qiu 2015 [24]	○	●	○	○	○	○	○	●	●	○	Lyapunov[c]
Li 2015 [25]	○	○	○	◐	●	●	◐	●	●	○	Lyapunov/Algo.
Lu 2015 [26]	○	●	○	●	○	●	●	●	●	○	Lyapunov/Algo.
Niu 2015 [27]	○	○	○	●	○	●	○	●	●	○	Lyapunov/Algo.
Liu 2017 [28]	●	●	○	○	○	●	○	●	○	○	Algo.
Ogawa 2017 [1]	○	○	○	○	○	●	●	●	●	○	Numerical simulation
Wang 2017 [29]	◐	○	●	○	○	●	○	●	○	○	BIP/Lyapunov
Li 2018 [2]	○	●	●	○	○	●	○	●	◐	○	Algo.
Sadeghani 2018 [30]	○	○	◐	○	○	●	○	●	○	○	BIP
This work	●	●	●	●	●	●	◐	●	○	●	ILP[d]

●The characteristic is fully supported
◐The characteristic is partially supported
○The characteristic is not supported.
[a]Algo: Algorithm.
[b]BIP: Binary Integer Programming.
[c]Lyapunov: Lyapunov optimization.
[d]ILP: Integer Linear Programming.

3.2. Cost optimization of one-tier applications in hybrid clouds

In this work we deal with transactional workloads, which in the survey of Wang et al. [20] are also called one-tier applications. They are the third-most frequent type of workload in hybrid clouds. Some examples of this type of workload are web search, online video or business transactions. The most relevant papers in this area are reviewed below.

In [21], Li et al. present a cost optimization model for video on demand (VoD) services, where the cost is represented by network communication and cloud storage. In the same way, Bjorkqvist et al. [22] propose an algorithm to dynamically optimize the allocation of a service provider. This algorithm calculates the cost for the private cloud as the sum of a fixed part and another that is a function of the cloud utilization. The approach followed in Kaviani et al. [23] consists in exploring the asymmetric communication costs in the cloud in order to partition web applications. The partitioning problem covers the distinction between contents that can be migrated and contents that cannot. Qiu et al. [24] design a generic optimization framework for dynamic, cost-minimizing migration of content distribution services into a hybrid cloud, taking into account response time constraints. In [25], Li et al. optimize the operational cost by from a theoretical point of view; they include reserved, on-demand and spot VMs. This algorithm incorporates no a priori information of public cloud renting prices or future probability of user requests.

A different approach is taken in Lu et al. [26], where the authors try to maximize the profit of the owner of the private cloud. The model obtains a revenue for each request completed, which is reduced by the cost of resources required both in the private and public clouds. In [27], Niu et al. apply queuing theory to evaluate the response time while they pursue a trade-off between performance and cost for a given budget. Liu et al. [28] focus on cost minimization and guarantee of performance, proposing the Least Cost per Connection (LCC) algorithm. This algorithm chooses the most cost-effective VMs from the available public clouds. In [1], Ogawa et al. present a cloud bursting approach that includes the total cost and response time constraints. The cost model distinguishes between private, public and management costs. Wang et al. [29] formulate the computing request scheduling problem as an optimization problem. This model identifies the type of request, and depending on their type, they are sent to the public cloud or not. The work of Li et al. [2] proposes a cost-aware scheduling approach based on queuing theory, in which each VM is represented as a queue. Then they use genetic algorithms to share the requests among these queues and a neural network to predict their execution time. Although the model identifies storage, communication and computation costs, in the end the cost is given only by the VM type. Finally, in [30], Sadeghani and Tarokh present a simplified cost minimization model formulated by binary integer programming that differentiates between the sensitivity of the jobs.

This work is based on Díaz et al. [3] and Entrialgo et al. [4]. In [3] the authors developed a methodology, called Lloovia, based on integer linear programming that is used to optimize the deployment cost in the cloud. The model supports different types of VMs and takes into account the constraints imposed by cloud providers on the number of VMs that can run simultaneously. Later, in Entrialgo et al. [4], the framework was extended as Malloovia, to support multiple applications running simultaneously in the cloud, each of them with their own requirements and workload.

Table 1 presents a comparison of all the previously cited works in this subsection with our work, covering several factors. The first assesses whether the workload consists of only one type of application or can adopt different types. The second factor covers the characteristics of the public cloud, such as if the solution can be applied to one or several cloud providers, whether the solution supports different types of VMs, if there is a limit to the number of VMs that can be hired, and the price model of the VMs hired. The third factor summarizes how the cost of the hybrid cloud is calculated, if it includes the cost of the private cloud, and if the public cloud is represented only by the VM cost or incorporates additional costs. Another interesting factor is whether the optimization model is supported by a simulation tool that makes it possible to extend the tests. Finally, the last column indicates the methodology used to obtain the solution.

As can be seen in Table 1, few of the related papers cover aspects such as having different applications or taking into account the heterogeneity of the types of VM and their limitations. It can also be seen that the majority of the works do not develop a simulation tool able to validate and extend their work.

3.3. Cloud simulators

In Table 1 it is noted that our proposed optimization model includes a simulation tool to study other characteristics not directly supported by the model. This subsection is devoted to a review of the available cloud simulators.

Clouds are complex systems that can involve a large number of computers, networks, storage and other electronic devices. The validation of solutions proposed for cloud optimization such as scheduling, allocation or deployment strategies can sometimes be very expensive. The cost may come from the great number of computational resources needed or the time required to test the solution. Furthermore, even if the tests for validation are affordable, the results obtained may differ among them, due to the random nature of the real environment.

The solution to this well-known problem is the use of modeling and simulation techniques. Altevogt et al. [31] emphasize the value of modeling and simulation technologies in the context of cloud computing. In their work, the authors detail how various cloud hardware and software objects can be modeled individually and then put together to form a simulation model of a complete cloud.

There are many cloud simulators in the literature, CloudSim [32] being one of the first and most widely used. CloudSim is an event driven simulator, written in Java and developed in the CLOUDS Laboratory at Melbourne University. It has been used as a toolkit to model and simulate many cloud elements and characteristics such as private, public or hybrid clouds. In spite of its extensive use, CloudSim has several limitations. For example, it cannot represent interactive workloads as noted in the work of Vondra et al. [33]. In order to overcome these limitations, several authors have developed new simulation tools based on CloudSim and also written in Java.

There are also may simulation tools based on CloudSim, two examples being CloudAnalyst [34] and NetworkCloudSim [35]. CloudAnalyst was developed to overcome the limitations of CloudSim to evaluate large-scale distributed cloud applications. This simulator also offers a GUI that simplifies the configuration of the cloud system. NetworkCloudSim improves CloudSim mainly by extending the representation of the internal network of the data-center. In addition, it supports the modeling of advanced application models such as message passing applications. Recently, a new evolution of CloudSim, called CloudSim Plus [36], has been developed. It improves the way in which cloud appliances are represented, simplifying their simulation.

Other examples of tools for cloud modeling and simulation are GreenCloud [37], iCanCloud [38], Simic [39] and CloudSched [40], each with different characteristics. GreenCloud [37], written in C++ and OTcl (Object oriented TCL), is an extension of the well-known network simulator NS-2. GreenCloud is focused mainly on studying the energy consumption of cloud data-centers. iCanCloud [38], written in C++, is a simulator of cloud infrastructures that enables large experiments, permits the integration of brokering policies and offers a GUI for configuring simulations. Simic [39], written in Java, is able to simulate the collaboration of multiple clouds in distributing service requests. Finally, CloudSched [40], written in Java, is oriented towards the study and analysis of solutions involving different resource scheduling algorithms.

The simulation tools cited are only a representative group of all those that have been developed for cloud simulation. A first review can be found in Fakhfakh et al. [41], including both the simulation tools based on CloudSim, and other specific simulation tools. Recently an improved and updated survey paper was carried out by Mansouri et al. [42].

However, as the centralized cloud model has evolved towards the distributed Internet of Things (IoT) model, new paradigms have appeared such as *Fog computing* and *Edge computing*. In Bendechache et al. [43] the authors present a review in which they analyze simulation tools in the continuous cloud to IoT. The novelty of this work is that it presents some Fog simulators, iFogSim [44] being the most important. This tool is an extension of CloudSim to analyze Fog environments. Similar to CloudSim, iFogSim has been used as a base to develop new simulation tools that cover some new functionalities. An example is MobFogSim [45], which is used to study the migration of services in fog computing.

3.4. Simulators in Python

The tools developed in Díaz et al. [3] and Entrialgo et al. [4] (Lloovia and Malloovia) were coded using Python, so the simulator developed in this work, Simlloovia, is also coded in Python for compatibility. The use of Python programming language has increased in recent years and nowadays it is one of the most important in data science and other areas. However, Python is not a common tool for the development of cloud simulators. In the literature there are few papers that use simulation tools coded in Python.

Vondra et al. [46] developed a simulator to evaluate cloud autoscaling alternatives. The simulator is built using a queuing network paradigm and discrete event simulation, and although it is not written in Python, the authors claim its portability. In Boza et al. [47] the authors develop a model, also based on queuing theory, to evaluate three alternatives to pay for computing resources in the cloud. They develop a simple tool, written in Python, to calculate the cheapest cost of the cloud solution. However, it is in the new Fog paradigm that a new generation of simulators written in Python is spreading: Edge-Fog [48], YAFS [49], FogDirMime [50] and more recently 5GPy [51] and FogDirSim [52]. A complete review of Fog simulators can be found in Margariti et al. [53].

We implement our own simulation tool in Python, instead of using one of the simulation packages available. The main reason for this decision was to take advantage of the work carried out in Malloovia. To this end, it is specially adapted to the characteristics of the environment analyzed in the Malloovia model, taking into account the parameters used to describe the system and the workload.

4. Optimizer tool

4.1. Introduction to the model and solver

Malloovia is an optimization tool which uses the workload as a prediction, and generates an allocation that ensures that the workload is handled, at minimum cost. This allocation specifies the number and types of the VM to be instantiated at each time slot. In order to take advantage of the reserved instance prices, the allocation has to take into account a whole year (which is the minimum time for a reservation in most cloud providers).

Malloovia operates in two phases. In the first phase an allocation is generated, using a long-term workload prediction which covers one year. This prediction is assumed to be known. How to obtain the prediction is out of the scope of this paper, but any of the techniques shown in [54] can be used to obtain an approximation, or for simple scenarios the same workload observed in previous years can be used, perhaps scaled by some factor. This phase is run “off-line” (before the actual deployment). The solution of this phase is used to determine the required number of reserved public instances, and the size of the private infrastructure.

Later, a second phase is run “on-line”. In the second phase Malloovia is used again to determine the optimal allocation for the next time slot only. It uses a short-term workload prediction that contains only the workload for the next time slot, making it more accurate than the long-term prediction. This short-term prediction can be derived from the workload of previous time slots. Again, this prediction is assumed as input and it is outside the scope of Malloovia how to obtain it. The second phase uses the private and public reserved infrastructure from the first phase, and decides only about the optimal mix of additional on-demand instance types required to handle the workload for the next time slot.

The complete set of parameters used in the model and the mathematical notation is presented in Section 4.2. The formal definition of the allocations produced as output is presented in Section 4.3. The cost function to be optimized and the restrictions to be enforced are presented in Section 4.4. Finally, some considerations about how to use this tool to model a hybrid scenario are presented in Section 4.5.

4.2. Optimizer inputs

All the data the optimizer tool requires can be classified in two sets of parameters: those that describe the cloud infrastructure in which the applications will be run, and those that describe the applications themselves, their computational requirements and the expected workload.

4.2.1. Infrastructure model

Cloud providers offer different types of VMs with different characteristics and prices, which may vary depending on the geographical region in which they are deployed. In addition, there are VMs that are deployed in the private infrastructure. To model this variety, Malloovia defines the concept of “Instance Class”, which encompasses the VM type, the region and the pricing model. An example of Instance Class could be a machine of type c5.xlarge in the Ireland region, under “on-demand” pricing schema. Another Instance Class would be a particular type of VM in the private data center.

“Limiting sets” are used to model the different restrictions set by the cloud provider, or by the private cloud infrastructure. They can be used to model per-region or per-availability zone limits imposed by public providers, or per data center resource limits in private clouds. Malloovia’s solver must ensure that none of the restrictions imposed by the limiting sets are violated in the solution allocation. The notation for the cloud infrastructure is summarized in Table 2.

Each “Instance Class” (IC_i), specifies a VM type (type_i, e.g. c5.large), the number of cores the VM uses (c_i), a boolean (rsv_i), which denotes whether the VM is reserved or on-demand, another boolean (priv_i) specifying whether the VM runs in private or public infrastructure, the price (p_i) or imputed cost per time slot, the performance $\mathrm{perf}_{i,a}$ which can be achieved for each application (more on this in Section 4.2.2), and the list of limiting sets ($\mathbf{LS}_i$) to which it belongs.

Each limiting set, LS_j, defines several resource limits, such as the maximum number of VMs allowed in that set, $\mathrm{LS}_j^{\mathrm{vms}}$, or the maximum number of cores, $\mathrm{LS}_j^{\mathrm{cores}}$, which can be used in that set at a given time. Each instance class, IC_i, can belong to one or more limiting sets. Finally, S_j denotes the list of instance classes that belong to limiting set LS_j.

Limiting sets are a versatile tool to model different sets of restrictions. For example, in recent years Amazon EC2 changed the kind of limits imposed. Initially, the limits were on the maximum number of VMs (of any type) per region. Later, the limits were imposed on VM families, for example, the maximum number of VMs of the family c5 that could be run in the same region. Currently (as of January 2021) there are limits on the total number of reserved VMs per region (of any type), and on the total number of cores used by families or groups of families of VMs. Azure also imposes limits on the number of cores per family and also on the total number of cores per region. All these kinds of limits can be modeled using Limiting Sets. For example, to model the limit on the reserved instances that can be deployed in a region, a limiting set LS_1 can be created, a value is assigned to its $\mathrm{LS}_1^{\mathrm{vms}}$ property,

Table 2
Notation in the model. Input variables related to the cloud infrastructure (instance classes and limiting sets).

IC_i	Instance class i
$\mathrm{IC}^{\mathrm{res}}$	Set of all reserved (or private) instance classes
$\mathrm{IC}^{\mathrm{dem}}$	Set of all on-demand instance classes
rsv_i	Boolean: is the instance class i reserved?
priv_i	Boolean: is the instance class i private?
p_i	Price or cost per time slot of instance class i
$\mathrm{perf}_{i,a}$	Performance of the instance class i running application a
c_i	Number of cores in instance class i
$\max_i$	Maximum allowed running instances for instance class i
$\mathbf{LS}_i$	List of limiting sets to which the instance class IC_i belongs
LS_j	Limiting set j
$\mathrm{LS}^{\mathrm{vms}}{}_j$	Maximum total number of VMs that can be active at any time in the limiting set j
$\mathrm{LS}^{\mathrm{cores}}{}_j$	Maximum total number of CPU cores that can be active at any time in the limiting set j
S_j	List of instance classes belonging to limiting set j

Table 3
Notation in the model. Input parameters related with the workload.

N_A	Total number of applications
A_a	Application a
T	Length of the reservation period
t	Length of each time slot
t_k	kth time slot
$l_{k,a}$	Expected workload during the time slot t_k for application A_a
$\boldsymbol{l}_k$	Workload vector (tuple of workloads, one per application) for time slot t_k
LTWP	Long-term workload prediction (T/t workload vectors) used in Phase I
STWP	Short-term workload prediction (a single workload vector) used in Phase II

and all Instance Classes representing reserved instances in that region will be assigned to the limiting set LS_1. Any instance class can be assigned to several limiting sets, which allows modeling the kind of limits imposed by Azure. For example, if an Azure region imposes a limit of 30 cores, but also the D series family imposes a limit of 30 cores, this will be modeled by using two limiting sets LS_2 (for the region) and LS_3 (for the D series in that region), with $\mathrm{LS}_2^{\mathrm{cores}} = 30$ and $\mathrm{LS}_3^{\mathrm{cores}} = 30$. Any Instance Class modeling VMs of type D1, D2, …, D5, will belong to both Limiting Sets $\{\mathrm{LS}_2, \mathrm{LS}_3\}$, but Instance Classes modeling VMs of the A series will belong to LS_2 but not to LS_3.

Limiting sets can be used also to model restrictions on the available hardware in the private cloud. For example, all private instance classes in the same data center can be assigned to the same limiting set, e.g., LS_4, which can limit the total number of cores used by these VMs through its property $\mathrm{LS}_4^{\mathrm{cores}}$.

4.2.2. Application and workload model

The kind of workload for which Malloovia optimizes the cost is composed of several independent stateless transactional applications, each one receiving a number of requests per time slot, which must be fulfilled within that time slot. The notation for the application and workload model is summarized in the first lines of Table 3.

We denote the set of applications as $A = \{A_1, \dots, A_{N_A}\}$, and use the subindex a to denote one particular application. Each instance class can run a single application from the set A in any given time slot. Depending on the characteristics of the instance class, the performance of the application being run will vary, so the performance of the system is given as a set of values $\{\mathrm{perf}_{i,a}\}$, each one representing the maximum number of requests that the instance class IC_i running application A_a can serve in a time slot. This number, which can be found by benchmarking or estimated by other means, is an input for Malloovia.

The expected workload for each application in each time slot is supposed to be known, and it is an input parameter for Malloovia. Two different predictions are required, depending on the solving phase:

- In Phase I, an approximation of the expected workload for the whole deployment period is required, but it needs not to be very accurate. We call this prediction "Long Term Workload Prediction" (LTWP). It is represented by a sequence $\{\boldsymbol{l}_1, \dots, \boldsymbol{l}_{T/t}\}$. Each $\boldsymbol{l}_k$ represents the global workload at time slot t_k, and is decomposed in several values $\boldsymbol{l}_k = (l_{k,1}, \dots, l_{k,N_A})$, one per application. Therefore, $l_{k,a}$ represents the expected number of requests for application A_a at time slot t_k. The sequence of $\{\boldsymbol{l}_k\}$ has length T/t, which is the number of time slots in a deployment period. For example, if time slot t is one hour and deployment period T is one year, T/t is 8760, the number of hours in a year, and this would be the length of the LTWP.
- In Phase II the required workload prediction is simply that of the next time slot. We call this prediction "Short Term Workload Prediction" (STWP), and it is composed of a single tuple, $\boldsymbol{l}_k = (l_{k,1}, \dots, l_{k,N_A})$, with the expected number of requests per application for the next time slot t_k.

Table 4
Notation in the model. Output variables (allocations).

X_{aik}	Number of VMs of on-demand instance class IC_i which must be running application A_a in time slot t_k
Y_{ai}	Number of VMs of reserved or private instance class IC_i which must be running application A_a in all time slots

4.3. Optimizer output

The output of the optimizing tool, which is summarized in the bottom part of Table 4, is an allocation that can generally be represented by a set of values, $\{X_{aik}\}$, which represents the number of VMs of the instance class IC_i that must be simultaneously running the application A_a in time slot t_k. The solver ensures that this allocation fulfills all the restrictions imposed by the limiting sets, that the expected workload during each time slot can be completed within the time slot, and that the total cost for the sum of all the time slots is as low as possible.

From this output it is possible to find out how many resources are required in the private cloud (by looking at the X_{aik} values in which i represents instance classes IC_i belonging to the private infrastructure), how many reserved instance classes need to be purchased in the public cloud (by looking at the X_{aik} in which i represents public reserved instance classes), and so on.

For efficiency, we will use the set $\{X_{aik}\}$ to denote only the on-demand instance classes, and use a different symbol $\{Y_{ai}\}$ to denote the reserved instance classes. Y variables do not have a k subindex, because the number of reserved VMs does not depend on the time slot, since they are available for the whole deployment period. This separation of variables into two groups reduces the number of different variables and thus the size of the problem to be solved.

4.4. Optimization problem

4.4.1. Problem statement

Given the input parameters described in Sections 4.2.1 and 4.2.2 , and summarized in Tables 2 and 3, the goal of the solver is to find an allocation for each time slot such that:

- It minimizes the total cost for the whole deployment period.
- It provides enough computational power to serve the expected number of requests per application at each time slot.
- It meets the constraints imposed by the limiting sets to which instance classes belong.

To find this optimal allocation, a set of equations can be written that describe the total cost of the deployment, and the restrictions that the solution must fulfill. These are linear equations that can be solved using Linear Programming techniques. The unknowns of the problem are the values of $\{X_{aik}, Y_{ai}\}$ (refer to Table 4). By solving those unknowns, the desired optimal allocation is obtained.

The cost to minimize can be expressed as:

$$C = \sum_a \sum_i Y_{ai} p_i^{\text{res}} T/t + \sum_a \sum_i \sum_k X_{aik} p_i^{\text{dem}} \tag{1}$$

The restriction which ensures that the expected workload is fulfilled in each time slot can be written as:

$$\sum_i Y_{ai} \text{perf}_{i,a} + \sum_i X_{aik} \text{perf}_{i,a} \geq l_{k,a}$$
$$\forall k \in \{1, \dots, T/t\}; \quad \forall a \in \{1, \dots, N_A\} \tag{2}$$

Other restrictions, such as the limits on the number of VMs of each class active in one time slot, are written as:

$$\sum_a Y_{ai} \leq \max_i^{\text{res}} \qquad \forall i \in \text{IC}^{\text{res}} \tag{3}$$

$$\sum_a X_{aik} \leq \max_i^{\text{dem}} \qquad \forall k \in \{1, \dots, T/t\}; \quad \forall i \in \text{IC}^{\text{dem}} \tag{4}$$

in which $\max_i^{\text{res}}$ is the maximum number of reserved instances of type i allowed by the public cloud provider, or by the private infrastructure, and $\max_i^{\text{dem}}$ is the same but for on-demand instance classes.

Similar equations can be written for the limits per region and zone, imposed by the limiting sets:

$$\sum_a \sum_{i \in S_j^{\text{res}}} Y_{ai} + \sum_a \sum_{i \in S_j^{\text{dem}}} X_{aik} \leq \text{LS}_j^{\text{vms}} \qquad \forall k \in \{1, \dots, T/t\}, j \in \{\text{LS}\} \tag{5}$$

$$\sum_a \sum_{i \in S_j^{\text{res}}} c_i Y_{ai} + \sum_a \sum_{i \in S_j^{\text{dem}}} c_i X_{aik} \leq \text{LS}_j^{\text{cores}} \qquad \forall k \in \{1, \dots, T/t\}, j \in \{\text{LS}\} \tag{6}$$

where $\{\text{LS}\}$ is the set of all limiting sets in the model, LS_j^{vms} is the maximum total number of virtual machines that can be active in any given time slot in the limiting set j, $\text{LS}_j^{\text{cores}}$ is the maximum total number of CPU cores that can be used in any given time slot in the limiting set j, and S_j^{res} (respectively S_j^{dem}) is the set of all reserved (respectively on-demand) instance classes IC_i which belong to the limiting set LS_j.

The same set of equations is valid for Phase I and Phase II. The difference is that in Phase I the number of time slots to consider are those in the whole deployment period (usually one year), and the unknowns are both Y_{ai} and X_{aik}, while in Phase II the number of time slots to consider is only one, the values of Y_{ai} are given (by the result of Phase I), and only X_{aik} are unknown.

For more details about how more restrictions can be formulated, as well as how the number of equations and unknowns for Phase I of this problem can be reduced by using techniques such as "workload histograms" and "workload quantization", the reader is referred to [3,4].

4.4.2. Resolution

Once the problem is mathematically formulated, the equations must be written in a format appropriate for the software used to solve the linear programming problem.

Malloovia is written in Python, but it uses an external solver to find the actual solution. The solver is independent of Malloovia, but in the experiments the CBC solver from the COIN-OR suite [55] is always used.

In order to carry out Phase I, Malloovia reads all the parameters required in the model from YAML files and creates from these a set of Python objects that represent the different entities in the model: instance classes, applications, workload predictions, limiting sets, and so on. Alternatively, this model can be created programmatically. Then, the PuLP library is used to create a Python representation of the linear problem to be solved (the objective function and the restrictions), and to convert all of this into a file that describes the problem for a particular solver, CBC in our case.

The solver is run in a separate process. When the solver finishes, the solution, which has been written by the solver in a file, is read back by PuLP, which converts the data to Python objects. Malloovia converts this raw data into allocation objects. This make it possible to examine the allocation per time slot, aggregate it using different criteria, compute its cost, extract the number of reserved VMs and the new private infrastructure to purchase, etc. This solution is written back into a Malloovia Solution file, in a YAML or pickle format. It is then made available to the analyst, who can further analyze it with other tools such as Jupyter Notebooks, or Simlloovia. This solution file also includes the infrastructure and application models as well as the workload used to find that solution.

Phase II can be run on-line before a new time slot begins. In this second phase, the same tools, models, files and solver are used. The only difference is that the input contains an initial number of machines to be kept active (corresponding to reserved and private instances), and the workload prediction is for the next time slot only. The solution in this case includes the allocation for the next time slot only, and this allocation uses the given values for the reserved and private instances, and solves only for the on-demand instances.

Malloovia is open source[1] and includes a comprehensive set of tests, both unit and integration tests, to check the validity of the solutions found, the correctness of the generated files, and the correct working of all the utility functions. These tests provide a coverage of 98%.

4.5. Hybrid clouds

Malloovia's model was originally designed to be multi-cloud. Thanks to the concepts of instance classes and limiting sets, it can also be used for hybrid clouds. In this section we explore how the private cloud can be modeled.

Virtual machines in the private infrastructure can be modeled as one or more private instance classes, belonging to one or more limiting sets. The performance of these instance classes can be obtained by benchmarking. However, for the price per time slot and the limits imposed through limiting sets, some additional considerations must be taken into account. The distinction made by Malloovia between reserved and on-demand instance classes basically translates into the fact that reserved instances are paid for all time slots, even if they are not required, whereas on-demand instances are paid for only for the time slots in which they are used. From this perspective, since private infrastructure is paid for even if not used (especially the amortization cost of the purchased machines), it can be modeled as reserved instance classes.

Private infrastructure can be seen as composed of two categories of machines: those that are already present in the private cloud, and additional machines which may be required, and thus must be purchased.

The VMs that will run in the "already present" infrastructure can be modeled as having a cost of zero (or nearly zero), because the cost of that infrastructure has already been paid for. The operational cost of this infrastructure may not be zero, but it is nevertheless outside the scope of the optimizer, because it cannot be reduced by using a different allocation. Assigning zero cost to these private VMs will make the solver use them before starting new VMs in the public cloud. To model the limited private resources already present, all machines in this category will be modeled as belonging to the same limiting set LS_j, and the value of $\text{LS}_j^{\text{cores}}$ will reflect the maximum number of cores available in the "already present" infrastructure.

The VMs that will run in the infrastructure to be purchased can be modeled with a price that is calculated by the cloud designer from the amortization cost of the equipment for the deployment period, by making it proportional to the number of cores each VM uses. For example, if the cost of purchasing some equipment with N cores is P, the amortization time is three years and the deployment period is one year ($1/3$ of the amortization period), then the cost per core and per hour will be $(1/3)P/(N \times 8760)$. A

[1] https://github.com/asi-uniovi/malloovia

VM which uses M cores will have a “price” per hour of $(1/3)(P \times M)/(N \times 8760)$. All these “to-be-purchased” machines will be modeled as belonging to one limiting set that will limit the maximum number of cores to be purchased, due, for example, to space restrictions in the company data center.

With this kind of modeling, Malloovia’s solver will prefer to use the already available private infrastructure for the base workload if the limits set on the private infrastructure allow it. If this infrastructure is not enough to serve the expected workload, Malloovia will balance the use of additional “to-be-purchased” private infrastructure, with the use of reserved instance classes in the public cloud, to serve the base workload, and with occasional use of on-demand public instances for the periods with higher workloads. The mix of private and public instance classes found by Malloovia for each time slot guarantees the minimum cost for the whole planning period.

5. Simulation tool

Malloovia obtains allocations with the optimal cost, but it does not give performance metrics such as response times and utilizations. In addition, in order to be able to solve the model, some assumptions and simplifications have been made. For instance, Malloovia assumes that all the requests in a time slot arrive with a uniform distribution within the time slot and that the load balancer distributes the requests between the available VMs perfectly.

In order to obtain performance metrics and to test Malloovia assumptions, a simulation tool, called Simlloovia, has been developed and is introduced in this paper for the first time. The code has been open sourced.[2]

5.1. Simulator inputs and outputs

The simulator is a command line program. As input, it needs the application and infrastructure models, the allocation and the workload to simulate. In addition, it also provides some options to control the simulation. It generates a set of files with performance data as output.

The application and infrastructure models, the allocation and the workload can be obtained from a Malloovia Solution file in either YAML or pickle format. This file is usually generated by Malloovia, but any other tool can be used: the YAML file is a text file and can be created simply by using a text editor. Thus, Simlloovia can be used with other tools to study transactional applications deployed in cloud infrastructure. In addition, a file with a different workload to the LTWP included in the Malloovia Solution file can be fed to the simulator, so that it can test the allocation with a workload different from that predicted. When this additional workload file is used, a parameter is required to indicate the length in seconds of the time slot used in the workload provided. If the workload file is not used, Simlloovia considers, as does Malloovia, that the length of the time slot in the workload is the same as the length of the time slot in the allocation.

In the set of options that are part of the input to Simlloovia, there is one to control how many time slots of the workload are simulated. The other options are used to select which output files are generated. By default, only one output file with a summary of the simulation is generated. This summary includes the number of requests processed, the response time (average, median, and 90 and 95 percentiles), the total cost and the average utilization.

One of the options that can be selected with an input parameter allows generating a request event file. This file lists, in CSV (Comma Separated Values) format, the following information for each request: creation time, processing start time, processing end time, the application to which the request corresponds, the VM that processed the request and the instance class of that VM.

Another option can be enabled to generate an output file that lists, in CSV format, the utilization for each VM.

These files can be processed with any tool. For instance, the figures shown in Section 6, which compare several simulations, were obtained using Jupyter Notebooks. However, in order to make it easier to study the results of a simulation, a dashboard has been created using the Dash Python library,[3] which generates dashboards as web pages. This dashboard (see Fig. 3) shows the results in the summary graphically, a Gantt diagram of the requests, and several plots of response time and utilization using different aggregations, which can be selected interactively.

5.2. System description

This section describes the system to be simulated, the parts of which will become simulation components in the simulator architecture shown in Fig. 4.

As this work focuses on cost optimization from the point of view of a client executing transactional applications in hybrid clouds using IaaS, the VMs executed in private and public clouds are one of the main components.

These VMs support different kinds of applications, each with their own workload, independent of the rest. Thus, a load balancer must distribute the requests addressed for each application between the VMs available for that application. In addition, there must be some mechanism to start and stop the VMs. This can be part of the load balancer, but for the sake of generality, this work assumes it to be independent. The component responsible for this task is called the “VM manager”.

The set of VMs available at each instant, i.e., the allocations, are the output of the cost optimization techniques used before. These allocations are provided as an input to the VM manager by a component called the “allocator”.

[2] https://github.com/asi-uniovi/simlloovia

[3] https://plotly.com/dash

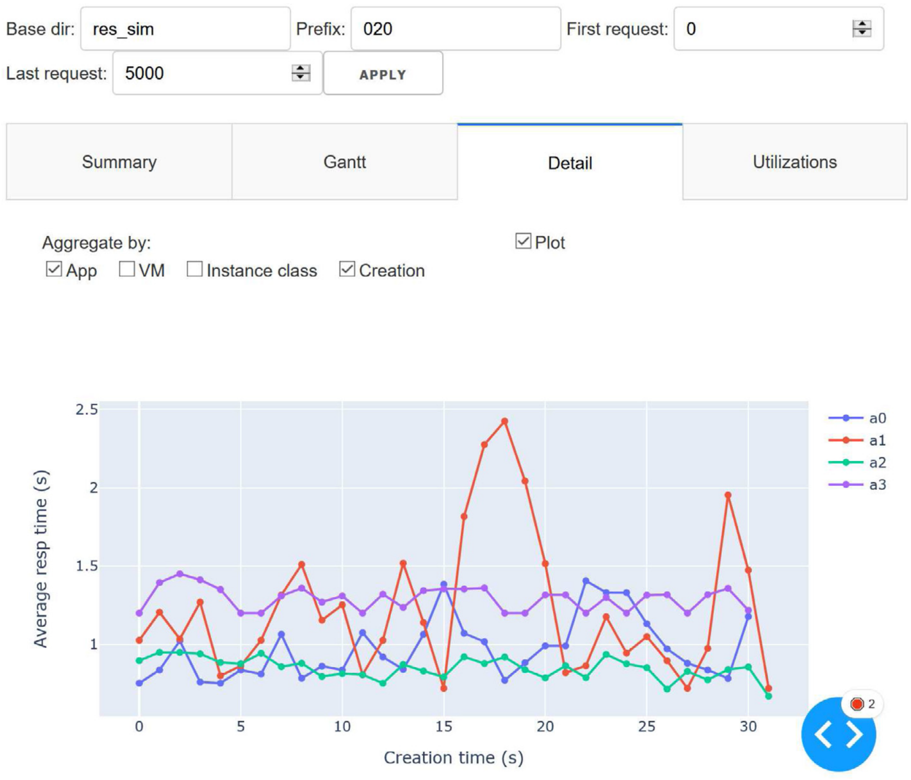

Base dir: res_sim Prefix: 020 First request: 0
Last request: 5000 APPLY

Fig. 3. Example of Simlloovia dashboard showing how the response time of each application changes in time.

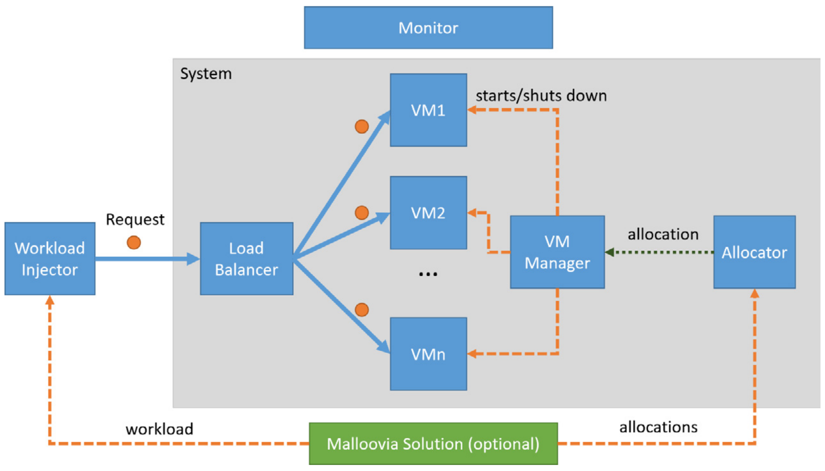


Fig. 4. Architecture of the simulator.

5.3. Simulation model

In order to simulate the system described in the previous subsection, the open queuing model shown in Fig. 5 has been used. The requests are generated by a workload injector that sends them to a queue representing the load balancer, which, in turn, forwards the request to the queues that represent each VM. Finally, the requests are destroyed in a sink.

The VM manager and the allocator are not modeled as queues because the model focuses on obtaining the response time of the requests and the cost associated with handling them.

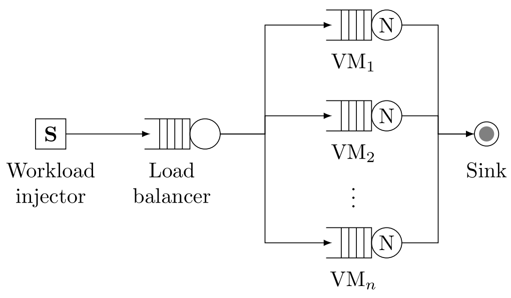


Fig. 5. Queuing simulation model.

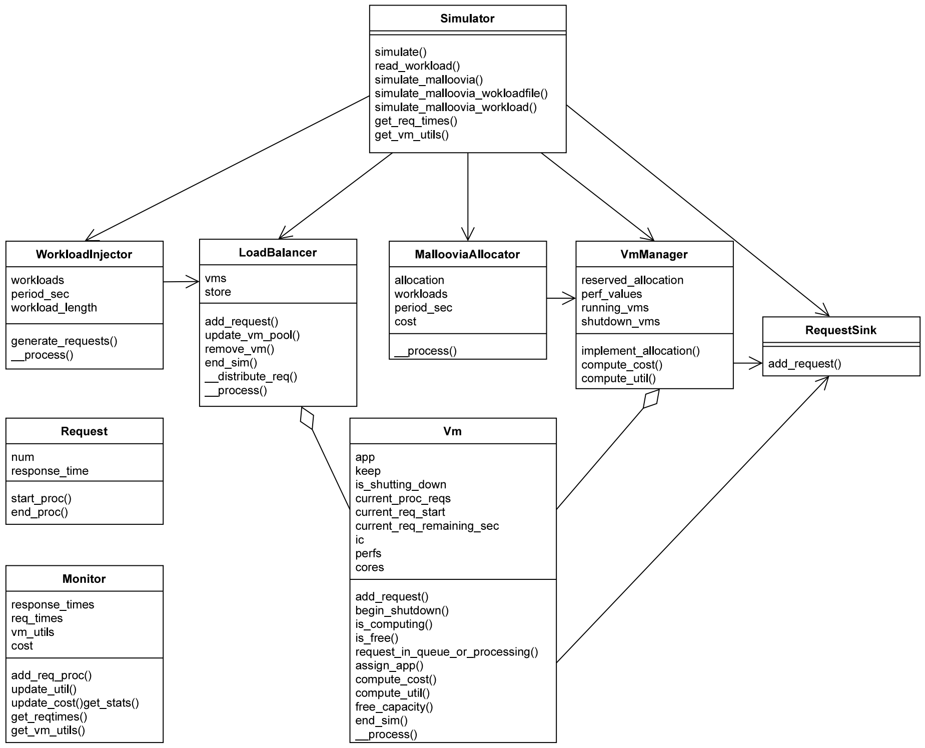


Fig. 6. Simlloovia class diagram.

5.4. Implementation

The model described above is implemented in Python using the Simpy [56] library. Fig. 4 shows the basic architecture of the simulator and Fig. 6 shows the main classes, attributes and methods that make up Simlloovia.

The `Simulator` class provides methods to carry out simulations with different options: using both the allocation and the workload in a Malloovia Solution file (with the method `simulate_malloovia`) or using only the allocation in this file but injecting a different workload that can come from a file (`simulate_malloovia_workloadfile`) or directly from a list generated programmatically (`simulate_malloovia_workload`). All these methods create the corresponding instances of the simulation components and call `simulate` to start the simulation and record the results. In order to take advantage of the code developed in the Malloovia tool, the data structures for describing the infrastructure, application and workload models are implemented with the classes defined in Malloovia.

As can be seen, the architecture components shown in Fig. 4 are classes in Fig. 6. All active simulation components have a private method called `__process`, which executes a simulation loop. The `WorkloadInjector` generates instances of the `Request` class in its loop. These are added to the `LoadBalancer`, which saves them in the `store` queue before distributing them to one of the VMs. Many load balancing techniques could be used here, but in this work a technique is used that sends each request to the VM with fewer requests in the queue or being processed. This value is obtained by the load balancer by iterating over the set of available VMs for an application and calling the corresponding method of the `Vm` class.

In order to simulate the processing of requests, instances of the `Vm` class simulate a multiprocessing system using a round robin scheduling policy with a quantum. Each VM has a queue for the requests and a Simpy `Resource` object to represent each core. The service time, $s_{i,a}$, of a VM of Instance Class i, for requests of app a, is computed by dividing the number of cores by the performance expressed in requests per time slot using this formula:

$$s_{i,a} = c_i / \mathrm{perf}_{i,a} \tag{7}$$

Each time a request is received, the service time is computed and stored in the field `current_req_remaining_sec` of the `Vm` class, where the remaining seconds for each request being executed in the VM are stored. The request waits for a core resource of the VM to be available. When this happens, it runs for a quantum and its remaining time is decremented. It then waits for a core to be available to repeat this process until there is no service time remaining.

VMs are created and shut down by the `VmManager` class to implement the allocation that the `MallooviaAllocator` sets in each allocation time slot. Instances of the `Vm` class register themselves with the load balancer to indicate that they are ready to receive requests. When the VM manager instructs a VM to shutdown, it deregisters itself.

The system includes a monitor component, implemented by the `Monitor` class, that receives information from all the other simulation components and generates the metrics that constitute the simulation tool output. In addition, the `Vm` class has methods to obtain the cost and the utilization of a VM. The cost of a VM is obtained by multiplying the number of seconds the VM has been running by the cost per second of its instance class. The total cost of the deployment is computed by adding the cost of all the VMs used. The utilization of a VM is computed by dividing the number of seconds it has been executing requests by the number of seconds it has been running and by the number of cores.

All these classes are used by a command line interpreter (CLI) module implemented using the Click[4] Python library. This module reads the arguments from the command line and calls the corresponding methods of the `Simulate` class to run the simulation and to obtain the request response times and VM utilizations so that it can save them in the output files selected with the input options.

In order to extend the simulator, alternative implementations of the classes shown in Fig. 6 can be provided. For instance, in order to implement a different load balancing policy, a new implementation of the `LoadBalancer` class can be provided, either by writing it from scratch or by subclassing the one provided and implementing only the method `__distribute_req()` differently. Similarly, to use an allocator not based on Malloovia, a new allocator class can be written.

Simlloovia was validated by simulating different small scale scenarios and comparing their results with computations carried out manually. These scenarios were included as integration tests in the test suite that can be found in the source code for Simlloovia.

In addition, larger scenarios were solved by Malloovia and then simulated. The cost obtained by Malloovia and Simlloovia was compared in order to validate the computations of cost. These scenarios were also included as integration tests.

Finally, unit tests were added where appropriate. The coverage of all tests is 95%.

6. Experiments and results

Section 2 introduced a motivating scenario with five questions about the dimensioning of the hybrid cloud for a company. This section shows the process to answer these questions using Malloovia and Simlloovia. All the code to generate and analyze the results has been open sourced.[5]

6.1. Inputs

The first step to use these tools is to establish the set of instance classes, both for the private and public clouds, that may be used to support the company's applications. For each instance class, the performance for each application, the cost per time unit and the limiting sets must be determined.

To quantify the performance of the instance classes of the previously existing private infrastructure, "priv-prev" for short, the company must run benchmarks in its infrastructure. In this example, it is assumed that there is only one configuration of existing VMs, but both Malloovia and Simlloovia can work with several configurations at the same time. The performance is expressed as the number of requests per second that a VM can serve before the response time starts to increase exponentially and can be measured using JMeter[6] or any other similar tool. Each application must be measured in isolation and the average number of requests served per hour must be recorded.

The company must determine the cost of its existing infrastructure. In this example, the existing infrastructure is assumed to have been already amortized so it will be modeled as having zero cost.

For the limiting sets of priv-prev, the maximum number of VMs that can be run in the existing infrastructure must be established. In this example, the maximum number of VMs is 10.

Regarding the new private infrastructure, "priv-new" for short, the company must determine the characteristics of the new servers to be purchased. In this example, it is established that only one type of server will be used. From the results of standard

[4] https://palletsprojects.com/p/click/
[5] https://github.com/asi-uniovi/hybrid_exp
[6] https://jmeter.apache.org/

Table 5
Performance and price of different VM types in the public and private cloud used in the motivating scenario.

Region	VM type	On demand	Reserved	Performance (rph)			
		($/h)	($/h)	A_0	A_1	A_2	A_3
Ireland	c5.large	0.10	0.06	10 000	5000	20 000	3000
	c5.xlarge	0.19	0.11	20 000	10 000	40 000	6000
	c5.2xlarge	0.38	0.23	39 000	19 500	78 000	11 700
	c5.4xlarge	0.77	0.45	73 000	36 500	146 000	21 900
London	c5.large	0.10	0.06	10 000	5000	20 000	3000
	c5.xlarge	0.20	0.12	20 000	10 000	40 000	6000
	c5.2xlarge	0.40	0.24	39 000	19 500	78 000	11 700
	c5.4xlarge	0.81	0.48	73 000	36 500	146 000	21 900
Private	priv-prev	–	0.00	10 000	5000	20 000	3000
	priv-new	–	0.01	20 000	10 000	40 000	6000

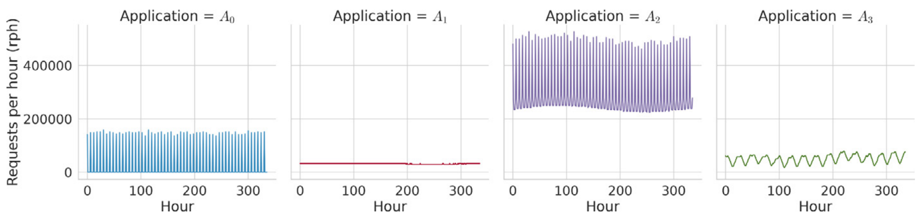


Fig. 7. Workload for each application during the first two weeks.

benchmarks, such as those published by SPEC,[7] the performance of the instance classes executed in these new servers can be forecast by comparing benchmark results of currently available servers with the results of the servers to be purchased.

In addition, the company must estimate the amortization cost per VM of deploying each server of the new private infrastructure, to use it as the price for each priv-new instance class. In this example, it is set at 0.01 $/h.

The company must also determine the maximum number of VMs that its data center can hold. This value will be used to define a limiting set for priv-new instances. In this example, the maximum number of VMs is 15.

The performance of instance classes in the public infrastructure can be obtained with the same measurement procedure used in the private infrastructure. In this example, it is assumed that this measurement has been carried out in a c5.large instance on AWS and its performance has been scaled to other types of instances using the relative ECUs (an AWS performance metric).

The cost of VMs in public clouds can be obtained from their public pricing information. The limits for different kinds of instances can also be obtained from this information. In this example, a limit of 20 instances per instance type is assumed.

Table 5 summarizes the performance and cost of all the VM types used in this example.

The company must also select an allocation period, i.e., the length of each time slot during which the allocation is not changed. A common approach is to use one hour, since until recently it was the billing period used by major cloud providers, i.e., VMs were charged for one hour even if they were only used for one second in that hour. Currently, AWS uses per-second billing, with a minimum of one minute, but using very short allocation periods is not practical, because of the time it takes to start VMs, so it is assumed that the company continues to use one hour as the allocation period.

As the company is interested in assessing whether using one-year reserved instances in AWS can generate savings, it needs a workload prediction for a whole year, which can be obtained by using workload prediction techniques [54] from the requests received the previous year. In this example, this long-term workload prediction (LTWP) has been generated synthetically and is composed of four applications with different characteristics (see Fig. 7). Application A_0 receives periodic requests every 6 h. Application A_1 has sustained activity, with daily and seasonal cycles. The workload of application A_2 is proportional to a mix of the previous two but with a higher base workload. Application A_3 has sustained activity with daily and seasonal cycles different from those of A_1.

This workload has been selected to represent the kind of workloads that cloud systems usually have. The workload of applications A_0, A_1 and A_2 is based on the workload presented in [4], while the workload of A_3 is based on the Wikipedia workload of 2014 [57].

The LTWP is expressed in requests per second (rps), but the allocation period is one hour, so the workload for Malloovia must be expressed in requests per hour (rph). In order to aggregate the rps into rph, the company can choose any percentile of the requests per second and multiply it by 3600 (the number of seconds in one hour). In this way Malloovia ensures that there are enough VMs to serve this number of requests in any given second. For instance, the 90-percentile for application A_0 in the first hour is 39. This means that in 90% of the seconds the workload is 39 requests or less. In order to serve all the requests in those seconds, the combined

[7] https://www.spec.org/benchmarks.html

Table 6
Summary of allocations.

Percentile	priv-new	priv-prev	pub-dem	pub-res	cost/year ($)
90	15	10	42	4	10305.4
95	15	10	42	8	12479.2
99	15	10	50	13	16181.4
100	15	10	63	26	23879.8

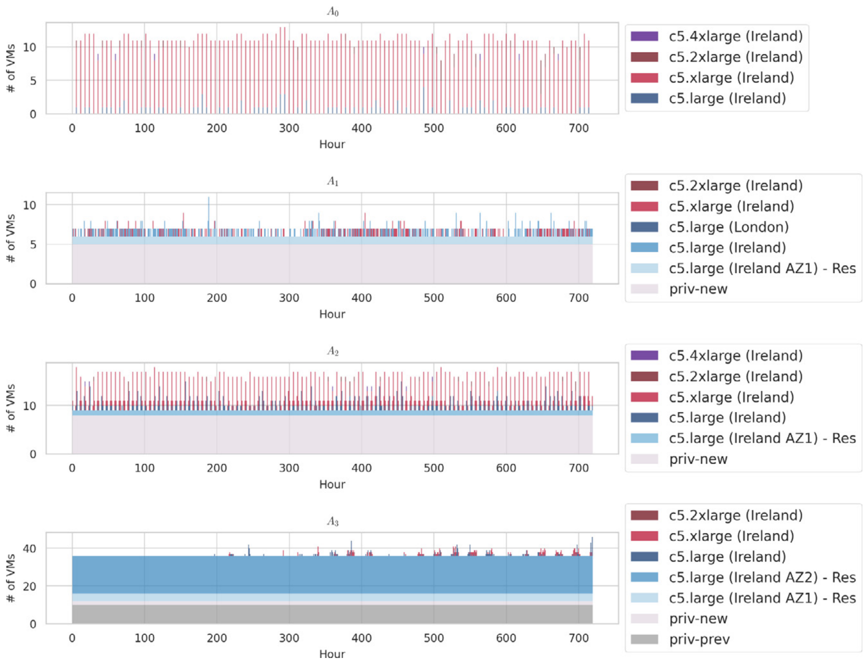


Fig. 8. Allocation for the first month of the configuration for the 100 percentile LTWP.

performance in rps of the VMs for that application must be equal to or higher than 39 and, thus, the combined performance in rph must be $39 \times 3600 = 140\,400$.

The percentile used changes the LTWP used by Malloovia, and thus the resulting allocation. This has an impact on the cost, the utilization and the response time. In order to handle a higher percentile, more infrastructure is required. This results in a higher cost but lower utilization because of the excess of capacity for seconds in which the workload is smaller. On the other hand, if lower percentiles are selected, the cost is lower, but the response times are higher. Thus, the company is considering configurations supporting 90, 95, 99 and 100 percentiles, obtaining different LTWP in rph.

6.2. Results obtained with Malloovia

With all this information, the company can create a program in Python using Malloovia to obtain the optimal allocation (answering Q1 from Section 2) and the cost (answering Q2) for each LTWP. As the company is interested in dimensioning its infrastructure for the whole year, only Phase I of Malloovia is used.

The results obtained are summarized in Table 6. Column "priv-new" is the number of VMs to be purchased for the new private infrastructure. Column "priv-prev" is the number of VMs in the previous existing private infrastructure. Column "pub-dem" is the number of on-demand VMs in the public cloud corresponding to the time slot in which the greatest number of VMs is used. Column "pub-res" is the number of reserved VMs in the public cloud. Finally, column "Cost" is the total cost of the allocation obtained by Malloovia. In addition, Fig. 8 shows how the allocation obtained by Malloovia for the 100-percentile case changes over time for each hour in the first month. There is one plot for each application with the number of VMs of each type for each hour.

These results, obtained by Malloovia, show that each configuration requires a very different allocation, from using 4 reserved instances in the public cloud for the 90 percentile to using 26 instances for the 100 percentile, with a large increase in on-demand instances. In addition, the apparently small increase from the 95 percentile to the 100 percentile means almost doubling the cost.

Table 7
Summary of simulation results for one week.

LTWP	Response time (s)				Cost ($)	VM
percentile	Min	Avg	Median	Max		utilization
90	0.18	1.00	0.72	33.00	147.68	0.71
95	0.18	0.82	0.72	10.16	197.76	0.66
99	0.18	0.78	0.66	2.60	259.53	0.60
100	0.18	0.74	0.56	2.40	408.81	0.47

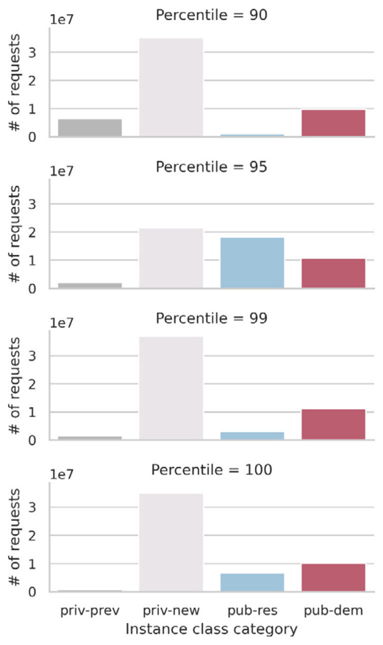


Fig. 9. Requests in the first week per category of instance class.

6.3. Results obtained with Simlloovia

Although these results obtained by Malloovia provide very valuable information for the company, there is no data about response time and utilizations, so the trade-off between cost and QoS cannot be fully explored only with Malloovia. For this, simulations using Simlloovia must be carried out.

The company can run simulations by feeding Simlloovia with solution files generated by Malloovia and the LTWP in rps. Table 7 summarizes the results obtained by simulating for the first week of each LTWP.

From these results obtained with Simlloovia, the company can answer Q3, about the VM utilization in each configuration. As can be seen, the configuration for the 90 percentile has a high level of utilization, which is reduced when higher percentiles of the LTWP are selected. One very interesting result shown in the last two rows of Table 7 is that moving from percentile 99 to percentile 100 results in a much larger difference in utilization than moving from percentile 95 to percentile 99. Fig. 9 shows how the requests are distributed among the different categories of VMs. As can be seen, the higher the percentile, the higher the number of requests that are served by the public cloud. This helps to explain why the utilization is lower for higher percentiles, as private infrastructure is always present even when it is not used.

The results obtained by Simlloovia also answer Q4 about the response times obtained with the different allocations and, thus, the QoS perceived by the user. As can be seen in Table 7, there is a very significant difference in the maximum response time between the configuration using the 90 percentile of the LTWP and the rest. These differences can be studied in more detail with the request event file generated by Simlloovia, which can be used to plot the CDF (Cumulative Distribution Function) of the response times for each application, as shown in Fig. 10. This figure shows response times in logarithmic scale in the X axis and the accumulated probability of that response time in the Y axis. It can be seen that there are significant differences in the response times of the different applications and how increasing the percentile reduces these response times.

Finally, all these comparisons can be used to answer Q5 about how changing the capacity provided affects both the response time and the cost. Accordingly, the company can choose the configuration that best fits their goals or, if it decides that it wants to explore more options, it can change some of the parameters and use Malloovia and Simlloovia again to study their effects. For instance, it could explore using a different kind of new VMs in the private infrastructure, using more regions in the public cloud or choosing a different size for the allocation time slot, e.g., half an hour.

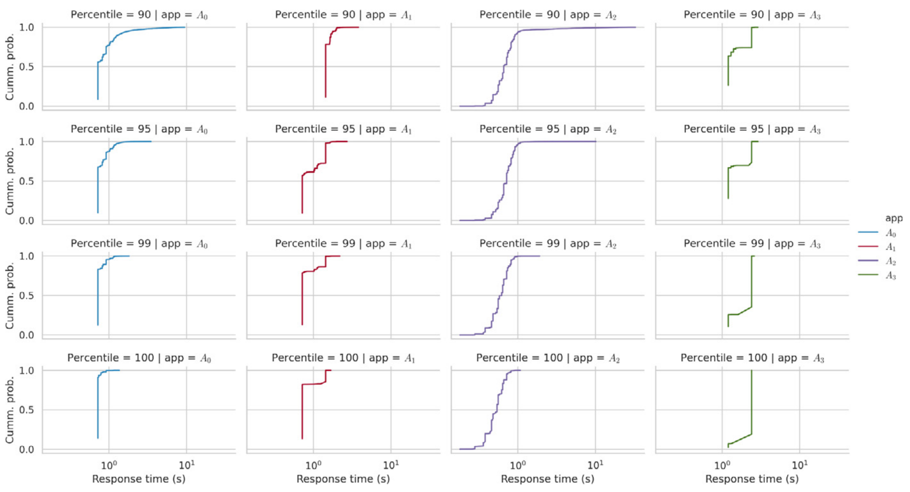


Fig. 10. CDF of the response times for each application.

The scenario described in this section has shown how the combination of Malloovia and Simlloovia can help companies in dimensioning their hybrid cloud infrastructure and how each of these tools provides different kinds of results.

7. Conclusions

Hybrid clouds are a common target for deploying transactional applications. As different kinds of VMs can be used in private and public clouds, VM deployments are complex and require correct dimensioning. This paper presents an approach for cost optimization and performance analysis of VM deployments in hybrid clouds based on two tools: Malloovia and Simlloovia.

Malloovia is a state-of-the-art technique for cost optimization in multi-cloud environments and this paper presents an extension so that it can be applied to hybrid clouds. The main advantage of Malloovia over other techniques is that it can take into account a wide array of features, such as using several applications, selecting between different types of VMs, leveraging reserved instance discounts, and respecting the limits imposed by the private and public clouds.

Simlloovia is a discrete event simulator for transactional applications running on cloud computing infrastructure. It is introduced for the first time in this paper. It can be used to analyze the performance characteristics of VM allocations in hybrid clouds, like those generated with Malloovia. These VM allocations can be simulated with different workloads in order to analyze properties such as response time and how requests are distributed between VMs.

Several experiments have been carried out to demonstrate how the combination of modeling and simulation provided by these two tools can help a cloud designer in selecting the best mix of different types of VMs in a hybrid cloud, including pre-existing and new infrastructure in the private cloud and on-demand and reserved instances in the public cloud.

The tools are open source, and the code to run all the experiments is made available in order to make the research reproducible.

Future work includes extending the simulator with more detail, so that it is closer to the behavior of real systems. Examples of extensions that can be added include taking into account the time required to start a new VM or the delays in network communications. Another aspect that will be explored is the use of containerized infrastructures to support multi-service applications and applying these ideas to non-transactional applications, such as machine learning applications, where the performance must be characterized differently.

References

[1] Y. Ogawa, G. Hasegawa, M. Murata, Cloud bursting approach based on predicting requests for business-critical web systems, in: 2017 International Conference on Computing, Networking and Communications, ICNC, IEEE, Silicon Valley, CA, USA, 2017, pp. 437–441, http://dx.doi.org/10.1109/ICCNC.2017.7876168.

[2] C. Li, J. Tang, Y. Luo, Towards operational cost minimization for cloud bursting with deadline constraints in hybrid clouds, Cluster Comput. 21 (4) (2018) 2013–2029, http://dx.doi.org/10.1007/s10586-018-2841-4.

[3] J.L. Díaz, J. Entrialgo, M. García, J. García, D.F. García, Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing, Future Gener. Comput. Syst. 71 (2017) 129–144, http://dx.doi.org/10.1016/j.future.2017.02.004.

[4] J. Entrialgo, J.L. Díaz, J. García, M. García, D.F. García, Cost minimization of virtual machine allocation in public clouds considering multiple applications, in: C. Pham, J. Altmann, J.A. Bañares (Eds.), Economics of Grids, Clouds, Systems, and Services, 10537, Springer International Publishing, Cham, 2017, pp. 147–161, http://dx.doi.org/10.1007/978-3-319-68066-8_12.

[5] Amazon, Amazon EC2, 2020, https://aws.amazon.com/ec2/. (Accessed 03 Apr. 2020).

[6] Microsoft, Microsoft azure virtual machines, 2020, https://azure.microsoft.com/en-us/services/virtual-machines/. (Accessed 03 Apr. 2020).

[7] Google, Google compute engine, 2020, https://cloud.google.com/compute/. (Accessed 03 Apr. 2020).

[8] Amazon, Amazon EC2 instance types, 2020, https://aws.amazon.com/ec2/instance-types/. (Accessed 03 Apr. 2020).

[9] RightScale, Rightscale 2019 state of the cloud report, RightScale, 2019.

[10] Flexera, Flexera 2020 State of the Cloud Report, Tech. Rep., Flexera, 2020.

[11] M. Malawski, K. Figiela, J. Nabrzyski, Cost minimization for computational applications on hybrid cloud infrastructures, Future Gener. Comput. Syst. 29 (7) (2013) 1786–1794, http://dx.doi.org/10.1016/j.future.2013.01.004.

[12] Y. Balagoni, R.R. Rao, A cost-effective SLA-aware scheduling for hybrid cloud environment, in: 2016 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC, IEEE, Chennai, 2016, pp. 1–7, http://dx.doi.org/10.1109/ICCIC.2016.7919621.

[13] S. Abdi, L. PourKarimi, M. Ahmadi, F. Zargari, Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds, Future Gener. Comput. Syst. 71 (2017) 113–128, http://dx.doi.org/10.1016/j.future.2017.01.036.

[14] V. Peláez, A. Campos, D.F. García, J. Entrialgo, Online scheduling of deadline-constrained bag-of-task workloads on hybrid clouds, Concurr. Comput.: Pract. Exper. 30 (19) (2018) e4639, http://dx.doi.org/10.1002/cpe.4639.

[15] G.L. Stavrinides, H.D. Karatza, Dynamic scheduling of bags-of-tasks with sensitive input data and end-to-end deadlines in a hybrid cloud, Multimedia Tools Appl. (2020) (2020) http://dx.doi.org/10.1007/s11042-020-08974-8.

[16] K.B. Naik, G.M. Gandhi, S.H. Patil, Pareto-based adaptive resources selection model in hybrid cloud environment, IETE J. Res. (2018) 1–13, http://dx.doi.org/10.1080/03772063.2018.1535919.

[17] B. Lin, W. Guo, X. Lin, Online optimization scheduling for scientific workflows with deadline constraint on hybrid clouds, Concurr. Comput.: Pract. Exper. 28 (11) (2016) 3079–3095, http://dx.doi.org/10.1002/cpe.3582.

[18] H. Liang, Y. Du, E. Gao, J. Sun, Cost-driven scheduling of service processes in hybrid cloud with VM deployment and interval-based charging, Future Gener. Comput. Syst. 107 (2020) 351–367, http://dx.doi.org/10.1016/j.future.2020.01.035.

[19] G.L. Stavrinides, H.D. Karatza, Cost-aware cloud bursting in a fog-cloud environment with real-time workflow applications, Concurr. Comput.: Pract. Exper. (2020) (2020) http://dx.doi.org/10.1002/cpe.5850.

[20] B. Wang, C. Wang, Y. Song, J. Cao, X. Cui, L. Zhang, A survey and taxonomy on workload scheduling and resource provisioning in hybrid clouds, Cluster Comput. (2020) (2020) http://dx.doi.org/10.1007/s10586-020-03048-8.

[21] H. Li, L. Zhong, J. Liu, B. Li, K. Xu, Cost-effective partial migration of vod services to content clouds, in: 2011 IEEE 4th International Conference on Cloud Computing, IEEE, Washington, DC, USA, 2011, pp. 203–210, http://dx.doi.org/10.1109/CLOUD.2011.41.

[22] M. Bjorkqvist, L.Y. Chen, W. Binder, Cost-driven service provisioning in hybrid clouds, in: 2012 Fifth IEEE International Conference on Service-Oriented Computing and Applications, SOCA, IEEE, Taipei, Taiwan, 2012, pp. 1–8, http://dx.doi.org/10.1109/SOCA.2012.6449447.

[23] N. Kaviani, E. Wohlstadter, R. Lea, Partitioning of web applications for hybrid cloud deployment, J. Internet Serv. Appl. 5 (1) (2014) 14, http://dx.doi.org/10.1186/s13174-014-0014-0.

[24] X. Qiu, H. Li, C. Wu, Z. Li, F.C. Lau, Cost-minimizing dynamic migration of content distribution services into hybrid clouds, IEEE Trans. Parallel Distrib. Syst. 26 (12) (2015) 3330–3345, http://dx.doi.org/10.1109/TPDS.2014.2371831.

[25] S. Li, Y. Zhou, L. Jiao, X. Yan, X. Wang, M.R.-T. Lyu, Towards operational cost minimization in hybrid clouds for dynamic resource provisioning with delay-aware optimization, IEEE Trans. Serv. Comput. 8 (3) (2015) 398–409, http://dx.doi.org/10.1109/TSC.2015.2390413.

[26] P. Lu, Q. Sun, K. Wu, Z. Zhu, Distributed online hybrid cloud management for profit-driven multimedia cloud computing, IEEE Trans. Multimed. 17 (8) (2015) 1297–1308, http://dx.doi.org/10.1109/TMM.2015.2441004.

[27] Y. Niu, B. Luo, F. Liu, J. Liu, B. Li, When hybrid cloud meets flash crowd: Towards cost-effective service provisioning, in: 2015 IEEE Conference on Computer Communications, INFOCOM, IEEE, Kowloon, Hong Kong, 2015, pp. 1044–1052, http://dx.doi.org/10.1109/INFOCOM.2015.7218477.

[28] F. Liu, B. Luo, Y. Niu, Cost-effective service provisioning for hybrid cloud applications, Mob. Netw. Appl. 22 (2) (2017) 153–160, http://dx.doi.org/10.1007/s11036-016-0738-0.

[29] Y. Wang, G. Xue, S. Qian, M. Li, An online cost-efficient scheduler for requests with deadline constraint in hybrid clouds, in: 2017 International Conference on Progress in Informatics and Computing, PIC, IEEE, Nanjing, 2017, pp. 318–322, http://dx.doi.org/10.1109/PIC.2017.8359564.

[30] S. Sadeghani, M.J. Tarokh, A cost model for hybrid cloud, Int. J. Comp. Inf. Technol. 5 (1) (2018) 01–06.

[31] P. Altevogt, W. Denzel, T. Kiss, Cloud modeling and simulation, in: S. Murugesan, I. Bojanova (Eds.), Encyclopedia of Cloud Computing, John Wiley & Sons, Ltd, Chichester, UK, 2016, pp. 315–326, http://dx.doi.org/10.1002/9781118821930.ch26.

[32] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. - Pract. Exp. 41 (1) (2011) 23–50, http://dx.doi.org/10.1002/spe.995.

[33] T. Vondra, J. Šedivý, J.M. Castro, Modifying CloudSim to accurately simulate interactive services for cloud autoscaling: Modifying CloudSim to accurately simulate interactive services for cloud autoscaling, Concurr. Comput.: Pract. Exper. 29 (10) (2017) e3983, http://dx.doi.org/10.1002/cpe.3983.

[34] B. Wickremasinghe, R.N. Calheiros, R. Buyya, 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, Australia, 2010, pp. 446–452, http://dx.doi.org/10.1109/AINA.2010.32.

[35] S.K. Garg, R. Buyya, NetworkCloudSim: Modelling parallel applications in cloud simulations, in: 2011 Fourth IEEE International Conference on Utility and Cloud Computing, IEEE, Victoria, NSW, 2011, pp. 105–113, http://dx.doi.org/10.1109/UCC.2011.24.

[36] M.C. Silva Filho, R.L. Oliveira, C.C. Monteiro, P.R. Inacio, M.M. Freire, CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness, in: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management, IM, IEEE, Lisbon, 2017, pp. 400–406, http://dx.doi.org/10.23919/INM.2017.7987304.

[37] D. Kliazovich, P. Bouvry, Y. Audzevich, S.U. Khan, GreenCloud: A packet-level simulator of energy-aware cloud computing data centers, in: 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, IEEE, Miami, FL, USA, 2010, pp. 1–5, http://dx.doi.org/10.1109/GLOCOM.2010.5683561.

[38] A. Núñez, J.L. Vázquez-Poletti, A.C. Caminero, G.G. Castañé, J. Carretero, I.M. Llorente, ICanCloud: A flexible and scalable cloud infrastructure simulator, J. Grid Comput. 10 (1) (2012) 185–209, http://dx.doi.org/10.1007/s10723-012-9208-5.

[39] S. Sotiriadis, N. Bessis, N. Antonopoulos, A. Anjum, SimIC: Designing a new inter-cloud simulation platform for integrating large-scale resource management, in: 2013 IEEE 27th International Conference on Advanced Information Networking and Applications, AINA, IEEE, Barcelona, 2013, pp. 90–97, http://dx.doi.org/10.1109/AINA.2013.123.

[40] W. Tian, Y. Zhao, M. Xu, Y. Zhong, X. Sun, A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center, IEEE Trans. Autom. Sci. Eng. 12 (1) (2015) 153–161, http://dx.doi.org/10.1109/TASE.2013.2266338.

[41] F. Fakhfakh, H.H. Kacem, A.H. Kacem, Simulation tools for cloud computing: A survey and comparative study, in: 2017 IEEE/ACIS 16th International Conference on Computer and Information Science, ICIS, IEEE, Wuhan, China, 2017, pp. 221–226, http://dx.doi.org/10.1109/ICIS.2017.7959997.

[42] N. Mansouri, R. Ghafari, B.M.H. Zade, Cloud computing simulators: A comprehensive review, Simul. Model. Pract. Theory 104 (2020) 102144, http://dx.doi.org/10.1016/j.simpat.2020.102144.

[43] M. Bendechache, S. Svorobej, P. Takako Endo, T. Lynn, Simulating resource management across the cloud-to-thing continuum: A survey and future directions, Future Internet 12 (6) (2020) 95, http://dx.doi.org/10.3390/fi12060095.

[44] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, Softw. - Pract. Exp. 47 (9) (2017) 1275–1296, http://dx.doi.org/10.1002/spe.2509.

[45] C. Puliafito, D.M. Gonçalves, M.M. Lopes, L.L. Martins, E. Madeira, E. Mingozzi, O. Rana, L.F. Bittencourt, MobFogSim: Simulation of mobility and migration for fog computing, Simul. Model. Pract. Theory 101 (2020) 102062, http://dx.doi.org/10.1016/j.simpat.2019.102062.

[46] T. Vondra, J. Šedivý, Cloud autoscaling simulation based on queueing network model, Simul. Model. Pract. Theory 70 (2017) 83–100, http://dx.doi.org/10.1016/j.simpat.2016.10.005.

[47] E.F. Boza, C.L. Abad, M. Villavicencio, S. Quimba, J.A. Plaza, Reserved, on demand or serverless: Model-based simulations for cloud budget planning, in: 2017 IEEE Second Ecuador Technical Chapters Meeting, ETCM, IEEE, Salinas, 2017, pp. 1–6, http://dx.doi.org/10.1109/ETCM.2017.8247460.

[48] N. Mohan, J. Kangasharju, Edge-Fog cloud: A distributed cloud for Internet of Things computations, in: 2016 Cloudification of the Internet of Things, CIoT, IEEE, Paris, France, 2016, pp. 1–6, http://dx.doi.org/10.1109/CIOT.2016.7872914.

[49] I. Lera, C. Guerrero, C. Juiz, YAFS: A simulator for IoT scenarios in fog computing, IEEE Access 7 (2019) 91745–91758, http://dx.doi.org/10.1109/ACCESS.2019.2927895.

[50] S. Forti, A. Ibrahim, A. Brogi, Mimicking FogDirector application management, SICS Softw.-Intensive Cyber-Phys. Syst. 34 (2–3) (2019) 151–161, http://dx.doi.org/10.1007/s00450-019-00403-y.

[51] R.I. Tinini, M.R.P.d. Santos, G.B. Figueiredo, D.M. Batista, 5GPy: A simPy-based simulator for performance evaluations in 5G hybrid Cloud-Fog RAN architectures, Simul. Model. Pract. Theory 101 (2020) 102030, http://dx.doi.org/10.1016/j.simpat.2019.102030.

[52] S. Forti, A. Pagiaro, A. Brogi, Simulating FogDirector application management, Simul. Model. Pract. Theory 101 (2020) 102021, http://dx.doi.org/10.1016/j.simpat.2019.102021.

[53] S.V. Margariti, V.V. Dimakopoulos, G. Tsoumanis, Modeling and simulation tools for fog computing: A comprehensive survey from a cost perspective, Future Internet 12 (5) (2020) 89, http://dx.doi.org/10.3390/fi12050089.

[54] M. Masdari, A. Khoshnevis, A survey and classification of the workload forecasting methods in cloud computing, Cluster Comput. 23 (4) (2020) 2399–2424, http://dx.doi.org/10.1007/s10586-019-03010-3.

[55] johnjforrest, S. Vigerske, H.G. Santos, T. Ralphs, L. Hafer, B. Kristjansson, jpfasano, EdwinStraver, M. Lubin, rlougee, jpgoncal1, h-i-gassmann, M. Saltzman, coin-or/Cbc: Version 2.10.5, 2020, http://dx.doi.org/10.5281/zenodo.3700700.

[56] A. Meurer, C.P. Smith, M. Paprocki, O. Čertík, S.B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J.K. Moore, S. Singh, T. Rathnayake, S. Vig, B.E. Granger, R.P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M.J. Curry, A.R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, A. Scopatz, SymPy: symbolic computing in Python, PeerJ Comput. Sci. 3 (2017) e103, http://dx.doi.org/10.7717/peerj-cs.103.

[57] Wikitech, Analytics/archive/data/pagecounts-raw — Wikitech,, 2019, [Online]. URL https://wikitech.wikimedia.org/w/index.php?title=Analytics/Archive/Data/Pagecounts-raw&oldid=1821019. (Accessed 10 January 2021).