Cost Minimization of Virtual Machine Allocation in Public Clouds Considering Multiple Applications

Joaquín Entrialgo, José Luis Díaz, Javier García, Manuel García, and Daniel F. García

University of Oviedo, Asturias, 33204 Gijón, Spain,
{joaquin, jldiaz, javier, mgarcia, dfgarcia}@uniovi.es

Abstract. This paper presents a virtual machine (VM) allocation strategy to optimize the cost of VM deployments in public clouds. It can simultaneously deal with multiple applications and it is formulated as an optimization problem that takes the level of performance to be reached by a set of applications as inputs. It considers real characteristics of infrastructure providers such as VM types, limits on the number VMs that can be deployed, and pricing schemes. As output, it generates a VM allocation to support the performance requirements of all the applications. The strategy combines short-term and long-term allocation phases in order to take advantage of VMs belonging to two different pricing categories: on-demand and reserved. A quantization technique is introduced to reduce the size of the allocation problem and, thus, significantly decrease the computational complexity. The experiments show that the strategy can optimize costs for problems that could not be solved with previous approaches.

Keywords: Cloud Computing · Cost Optimization · Virtual Machine Allocation · Multi-application

1 Introduction

Cloud computing has evolved quickly in recent years, becoming a useful and attractive alternative for deploying new applications. Cloud computing offers almost unlimited computing capacity (scalability), which can be immediately increased or decreased following the users' demand (elasticity). All of these characteristics are available through a "pay-as-you-go" model.

However, users who want to deploy their applications on the cloud have to answer an important question: how much cloud computing power should they hire? Hiring too much means a waste of money; on the other hand, hiring too little may reduce profits or, even worse, incur economic penalties if SLAs are not met. Therefore, there is a broad research in order to answer this question and to find the most cost-effective allocation, such as [1], [4,5,6], [9,10], [14,15] or [18].

Among the services provided by cloud computing, here we consider Infrastructure as a Service (IaaS), one of the fastest growing fields. Cloud providers offer different Virtual Machine (VM) types (for example, a VM type provided by Amazon EC2 is m4.large, which offers 2 virtual CPUs and 8 GiB of RAM [3]). Moreover, with regard to pricing, VMs can belong to two different categories: on-demand and reserved [2]. Thus, the cost optimization problem is complex. Firstly users have to choose the cloud provider and the datacenters, each of them with different costs, where to carry out their deployments. Then, users should choose the most appropriate VM types to execute their applications. Finally, if the applications will run for a long time, users should consider to take advantage of the lower price of reserved instances.

Therefore, minimizing deployment costs, while guaranteeing the fulfillment of a determined level of performance, is a usual objective of cloud computing users. To achieve this goal, a significant number of VM allocation strategies have been developed.

An allocation strategy produces a VM allocation that represents the number, types and pricing categories of the VMs to be deployed in a given time period. In order to determine the instants in which an allocation strategy is applied, the time is usually divided in regular time slots. A typical length for these slots is one hour, coinciding with the billing period of some important providers, such as Amazon EC2.

VM allocation strategies can be focused in the short or in the long term. A short-term strategy generates an allocation for the next time slot. In contrast, a long-term strategy usually operates with a yearly period, and it generates a VM allocation for each time slot of a year. Short-term strategies rely on on-demand VMs, because these are oriented to be started or stopped as required, following the instant variations of the workload. In contrast, long-term strategies can take advantage of reserved VMs in addition to on-demand VMs. Reserved VMs support the base workload of the applications, and on-demand VMs supplement the computing capacity provided by reserved VMs as required by the application demands.

Long-term strategies have to deal with severe difficulties. The allocation problems to be solved are huge (for example, considering a reservation period of one year and a time slot of one hour, 8760 allocations must be calculated and they can not be solved independently). Moreover, a significant number of VM types may have to be taken into account, so the solution space to be explored for each allocation may be vast, and the limits imposed by providers (that is, the maximum number of VMs that can be deployed in a region or in an availability zone) also hinder the computation of a solution.

A previous VM allocation approach, referred to as LLOOVIA (Load Level based OptimizatiOn for VIrtual machine Allocation) [6], combined a long-term and a short-term strategy, organized in two phases, to take advantage of both reserved and on demand VMs. LLOOVIA is designed to minimize allocation costs, while guaranteeing the fulfillment of a determined level of performance. However, LLOOVIA lacks the ability to deal with multiple applications: it is designed for managing only one application. To overcome this shortcoming, an improved version of LLOOVIA, named MALLOOVIA (Multi-Application Load

Level based OpimizatiOn for Virtual Machine Allocation), has been designed and is presented in this paper.

MALLOOVIA can deal with multiple applications, each one of them considered perfectly scalable by horizontal replication. MALLOOVIA is formulated as an optimization problem that takes the levels of performance to be reached by a set of applications as input, and generates a VM allocation to support the performance requirements of all the applications as output. The generated allocation minimizes the deployment cost of the applications, guaranteeing the required performance for each one of them. The applications managed by MALLOOVIA can be totally independent or can be part of a service. An example of the latter case is a multi-tier service, which can be considered as composed of different applications, corresponding each one of them to a different tier.

The management of multiple applications increases the size of the problems to be solved extraordinarily, and usually problems become intractable. To overcome this shortcoming, MALLOOVIA employs a quantization method that has the ability of reducing the number of performance levels to be dealt with for each application very significantly. This method has proven to be very effective in the experimental cases analyzed with MALLOOVIA.

2 Related Work

In this research, we focus on allocation cost minimization, while guaranteeing the fulfilment of performance requirements. We consider the problem of the deployment of several applications, in a multi-cloud environment. In this multi-cloud environment several types of VM, each of them with different prices and performance, can be chosen. Furthermore, cloud providers impose limits to the number of VMs that can be hired by the user. Finally two pricing models are considered, on-demand and reserved instances.

In the literature, there are several papers that approach the cost optimization problem of VM allocation. However, none of them cover all the previously mentioned aspects simultaneously. There are two main approaches to study this problem. They differ on the way they represent the system workload.

There is a first group of papers ([12], [5], [18], [8], [11] and [16]) where the workload is represented as the number of VMs required in each period, instead of as the arrival rate of requests to the system, which is more frequently used in transactional applications. The approach presented in these works solves only part of the problem, namely, to find the optimal allocation of VMs to providers, but it does not address the issue of determining the appropriate type and number of VMs to support a given arrival rate. This issue is not solved in those papers, because it is assumed to be known in advance.

In [12] the authors develop a heuristic approach to minimize the cost using a single cloud provider. They also consider reserved and on-demand VMs, but they do not consider any VM type distinction or limit. Their model is based on a prediction over historical data, and in a first stage, using this prediction, the number of reserved VMs to hire are estimated. In a second stage, using the real demand, the number of on-demand VMs needed to fulfill the requirements are obtained.

In [5] the authors propose an optimal cloud resource provisioning algorithm whose aim is to minimize the total cost for provisioning resources in a certain time period. The authors consider the cost resulting from both reserved and on-demand resources from multiple clouds. This model is able to manage different types of applications, but each application must be supported by the same VM type, that is, different types of VM cannot be mixed in the same application. In this model, the VMs are specified as a set of resources: computational power, storage, network bandwidth and electricity power, and in the same way, each cloud provider is represented as a pool of these resources. However, nowadays cloud providers offer VMs as a discrete set of configurations called VM types. The algorithm proceeds in two steps: in the first step a prediction of the VM demand is calculated, in the second step the number of reserved VMs to hire is obtained. In this paper, it is not clear how the on-demand VMs needed to cover the real demand are chosen.

The authors of [18] follow the same approach that [5], but with the difference that they rely on a combination of heuristic methods to find the optimal allocation in a reasonable time.

A more limited study can be found in [8], where the authors develop a heuristic to calculate only the number of reserved VM required for a given prediction. The heuristic provides a sub-optimal solution when it supports different reserved VM contracts. This heuristic is limited to only one application, not considering VM types or VM limits.

In [11] the authors apply a stochastic model based on Inventory Theory to find the optimal combination of reserved and on-demand VMs which minimizes the cost. Applying this model, the authors find an equation to calculate the number of reserved VMs to be leased. From this expression, they apply a heuristic process to find a purchase plan. The main drawback of this model is that it is limited to only one application, one VM type and one cloud provider, and the model does not consider any limit in the number of VMs that can be hired.

The last paper in which workload is given as number of required VMs is [16]. The authors propose a global broker which receives the users' demands and allocates them among a set of cloud providers working cooperatively. The proposed method works in two phases. In the first phase the number of reserved VMs needed is obtained, in a similar way to [8]. In the second phase, a heuristic algorithm is executed to minimize the users' usage cost of provisioning on-demand VMs. The main limitations of this work are that it is limited to only one application and it does not support VM types or any kind of limit.

There is a second group of papers ([10], [7], [4] and [17]) where the optimization problem is analyzed considering the workload as an arrival rate of requests that must be served using the allocated VMs. This represents a more common problem of cloud resource optimization.

In [10] the authors study the VM allocation problem for multimedia application providers. The providers aim to minimize the resource cost while meeting the round trip time requirements. They propose two optimal schemes for VM allocation for both single-site and multi-site clouds. In this work, both reserved and on-demand pricing schemes are considered, but the number of reserved VMs is known and fixed at the beginning of the algorithm. The algorithm only decides how many of them are used. This is not a valid approach because reserved VMs imply an initial cost, whether the VMs are used or not.

In [7] the authors investigate the time-cost optimization problem of tasks with deadline taking advantage of reserved VMs. They present two solutions: the cost optimization problem, where they look for the cheapest allocation, and the time optimization problem, where for a given budget, they find the allocation with the best processing time. This work considers only an application and it is guided by the VM leasing time, more than VM types or limits.

In [4] the authors present a model that optimizes the cost of a deployment of a multi-site application in a multi-cloud environment. The model considers both reserved and on-demand VMs and their pricing schemes. However, it only uses one application and one VM type in the analysis.

In [17] the authors propose a cloud brokerage service that aggregates the cloud user demands to take advantage of cheaper prices of reserved VMs. The service handles the cloud user demands with a pool of VMs that are either reserved or launched on-demand. The aim is to minimize the cost using as few on-demand VMs as possible. This work is limited to one VM type and only one cloud provider.

Finally, the most related work is [6]. This work covers all the characteristics considered here, except that it only deals with a single application. The objective of our paper is to extend [6] to solve the cost optimization problem when several applications have to be allocated on the same cloud resources.

In the literature there are only two similar works ([9] and [15]) that consider cost optimization of several applications on a cloud computing environment. However, they are limited by the type of resources they support and they only perform an static (off-line) analysis. Thus, in [9] the authors propose an algorithm that finds the most cost-effective allocation which meets the QoS requirements with the lowest cost. Its main drawbacks are that it is limited to on-demand VMs and it does not take advantage of reserved VMs. Finally, in [15] the authors solve the problem of cost optimization of concurrent services when they are executed on a multi-cloud environment considering different VM types, but as in the previous work it does not take advantage of reserved VMs.

Our paper approaches the cost optimization problem in a more complete way than previous works: it considers several application simultaneously; it can be used in multi-cloud environments; it takes into account how cloud providers support different VM types and their restrictions; and it considers both reserved and on-demand price schemes.

3 System Model and Resolution

3.1 Overview

The model presented in this paper is very similar to the one in [6], but extended to allow multiple applications to be deployed in the same (shared) cloud infrastructure. Most of the concepts and notation in [6] are still relevant, being the main differences: a) the workload is not longer a single number per timeslot, but a set of numbers, one per application, and b) the performance of any given instance class (defined later) is not a single number, but also a set of numbers.

The problem to solve is to find, for each timeslot, the number and types of VMs which should be acquired (both reserved and on-demand VMs), to run each of the applications. The number and types of reserved VMs will be fixed after the purchase, and will be the same for all timeslots, while the number and types of on-demand instances will vary. The problem is solved in two phases. Phase I tries to find the optimal number of reserved VMs of each type. This phase requires a long-term prediction of the workload, for the whole reservation period and for each application. This phase is carried out off-line, before the deployment. The result of this phase is used to buy reserved VMs for a whole reservation period. Phase II starts assuming that those reserved VMs are available, and uses a short-term workload prediction for each application, which consists of the expected workload per application, for the next timeslot. During phase II, at each timeslot, the expected workload is supported by a mix of the available reserved VMs, plus some extra on-demand VMs whose number and type is to be found in this second phase.

The problem is complicated by the fact that cloud providers can impose a limit on the maximum number of allocated VMs of each type, or a maximum total number of VMs allocated per region, or a maximum total number of CPU cores allocated per region, or a mix of several of these limits. Since these limits are on the infrastructure, and are not set per application, they make impossible to decompose the problem in several independent problems, one per application.

3.2 Infrastructure Model

To unify the different kinds of limits imposed by different cloud providers, we use the concept of **Limiting Set**, denoted by LS_j, which are sets in which VMs are deployed and which impose some kind of global limit on the VMs in that set. Each LS_j defines two limits: $\mathrm{LS}_j^{\mathrm{vms}}$, which is the maximum number of virtual machines which can run simultaneously in LS_j, and $\mathrm{LS}_j^{\mathrm{cores}}$, which is the maximum number of CPU cores which can run simultaneously in LS_j.

To unify the different VM types offered by different cloud providers in their different availability zones, and under different pricing schemas, we use the concept of **Instance Class**, denoted by IC. For example, an on-demand c4.large on Amazon EC2 on region us-east-2, is a different instance class than a reserved c4.large on Amazon EC2 on availability zone us-east-1b. For each IC_i the following attributes are defined:

- p_i is the price per time slot of the class. For reserved VMs this price should include the upfront payment prorated over the duration of the reservation period, and the per-hour cost.
- perf_{ai} is the performance of that class when it is used to run the application A_a, under the considered kind of load for that application, and expressed in the same units as the load. These values can be obtained via benchmarking or monitoring.
- rsv_i is a boolean denoting whether this instance class is reserved or not.
- c_i is the number of CPU cores provided by this class.
- ls_i is an integer, j, which is the index of the LS_j to which this instance class belongs.
- max_i is the maximum number of VMs of this class which can be instantiated in its limiting set. Some cloud providers also impose this kind of restriction, especially for high performance VM types.

Without loss of generality we can divide the set of all IC_i into two disjoint subsets, depending on the value of rsv_i. We will use the superindex $^{\text{res}}$ to refer to attributes of reserved instance classes and $^{\text{dem}}$ for on-demand ones. For example, p_i^{res}, $\text{perf}_{ia}^{\text{dem}}$, etc.

3.3 Applications and Workload Model

An application is the software that will be run in the instance classes. It can be thought as the disk image used to boot the VM. For example, one application can be a database and a second application can be a web server. The set of possible applications $A = \{A_1, A_2, ..., A_{N_A}\}$ is fixed. Each application has a different performance for each possible instance class, and this is captured by the attribute perf_{ia} previously seen, which is assumed to be known for all instance classes and applications.

We assume time divided into *slots* of length t (e.g., 1 hour), and denote each of these time slots by t_k. At any timeslot t_k, the workload is a vector $\boldsymbol{l}_k = \{l_{k,1}, l_{k,2}, \ldots, l_{k,N_A}\}$. The component $l_{k,a}$ is the expected workload for application A_a during timeslot t_k.

Note that for Phase I the sequence of $\boldsymbol{l}_k$ for all timeslots t_k is required in advance. This is a prediction that we will denote by LTWP (Long Term Workload Prediction). For Phase II, however, only the workload for the next timeslot is required, and we will denote it by STWP (Short Term Workload Prediction).

To reduce the problem size for Phase I we choose to represent the LTWP as a histogram. Given an arbitrary workload vector $\boldsymbol{L} = \{L_1, \ldots, L_{N_A}\}$, the histogram $H(\boldsymbol{L})$ is the number repetitions of that workload in the LTWP. More formally $H(\boldsymbol{L}) = \sum_{k=1}^{T/t} \text{eq}(\boldsymbol{L}, \boldsymbol{l}_k)$, being $\text{eq}(\boldsymbol{x}, \boldsymbol{y}) = 1$ if $\boldsymbol{x} = \boldsymbol{y}$, and 0 otherwise.

We define the effective workload, and denote it by $\mathbb{L}$, as the set of all vector loads which appear at least once in the LTWP, or, more formally $\mathbb{L} = \{\boldsymbol{L} : H(\boldsymbol{L}) > 0\}$. Note that, if some $\boldsymbol{L}$ appears twice or more times in the LTWP, then the size of $\mathbb{L}$ will be smaller than the size of LTWP, so this representation saves space, being equal in the worst case (when no workload vector ever repeats).

3.4 Optimization Problem for Phase I

The optimization problem can be formulated as an integer linear programming problem, with the unknown integer variables Y_{ai}, which is the number of reserved VMs of class $\mathrm{IC}_i^{\mathrm{res}}$ to be purchased at the beginning of the reservation period T to run application A_a, and $X_{ai\boldsymbol{L}}$ which is the number of on-demand VMs of class $\mathrm{IC}_i^{\mathrm{dem}}$ to run application A_a to be purchased at any time slot for which the predicted vector load is $\boldsymbol{L}$. Since reserved instances are paid even if not used, the analysis assumes those machines to be always available.

The function to optimize is the cost for the whole reservation period, which can be calculated as:

$$C = \sum_{a=1}^{N_A} \sum_{i=1}^{N^{\mathrm{res}}} Y_{ai} p_i^{\mathrm{res}} T/t + \sum_{a=1}^{N_A} \sum_{i=1}^{N^{\mathrm{dem}}} \sum_{\boldsymbol{L} \in \mathbb{L}} X_{ai\boldsymbol{L}} p_i^{\mathrm{dem}} H(\boldsymbol{L}) \tag{1}$$

This cost is minimized subject to restrictions:

$$\sum_{i=1}^{N^{\mathrm{res}}} \mathrm{perf}_{ai}^{\mathrm{res}} Y_{ai} + \sum_{i=1}^{N^{\mathrm{dem}}} \mathrm{perf}_{ai}^{\mathrm{dem}} X_{ai\boldsymbol{L}} \geq L_a \qquad \forall \boldsymbol{L} \in \mathbb{L}, \forall a = 1, \ldots, N_A \tag{2}$$

$$\sum_{a=1}^{N_A} Y_{ai} \leq \max_i^{\mathrm{res}} \qquad \forall i = 1, \ldots, N^{\mathrm{res}} \tag{3}$$

$$\sum_{a=1}^{N_A} X_{ai\boldsymbol{L}} \leq \max_i^{\mathrm{dem}} \qquad \forall \boldsymbol{L} \in \mathbb{L}, i = 1, \ldots, N^{\mathrm{dem}} \tag{4}$$

$$\sum_{a=1}^{N_A} \sum_{i \in S_j^{\mathrm{res}}} Y_{ai} + \sum_{a=1}^{N_A} \sum_{i \in S_j^{\mathrm{dem}}} X_{ai\boldsymbol{L}} \leq \mathrm{LS}_j^{\mathrm{vms}} \qquad \forall \boldsymbol{L} \in \mathbb{L}, j = 1, \ldots, N^{\mathrm{LS}} \tag{5}$$

$$\sum_{a=1}^{N_A} \sum_{i \in S_j^{\mathrm{res}}} c_i Y_{ai} + \sum_{a=1}^{N_A} \sum_{i \in S_j^{\mathrm{dem}}} c_i X_{ai\boldsymbol{L}} \leq \mathrm{LS}_j^{\mathrm{cores}} \qquad \forall \boldsymbol{L} \in \mathbb{L}, j = 1, \ldots, N^{\mathrm{LS}} \tag{6}$$

Restriction (2) states that, for each application, the performance given by the solution should be at least equal to the workload for that application, for all predicted workload vectors. Restrictions (3) to (6) represent the limits imposed by cloud providers on the total number of VMs of ech type, the total number of VMs per region and the total number of CPU cores per region, respectively. In the last two restrictions, the symbol S_j represents the set of instance classes which share the same limiting set LS_j, i.e: $S_j = \{i : \mathrm{ls}_i = j\}$.

3.5 Optimization Problem for Phase II

Phase II is very similar to Phase I, but much simpler. The set of equations to solve are the same already seen in Phase I, but now L is a set with a single element: the vector load for the next time slot (STWP).

During Phase II only on-demand instances can be hired, so it is necessary to include new restrictions which fix the number of reserved instances to the values found by Phase I. However, we can allow Phase II to reuse reserved instances for a different application than the one given by the allocation generated in Phase I. This way we can accommodate discrepancies between the long term prediction and the short term prediction (which will be in general more accurate).

To formalize this idea, lets call Y'_{ia} the solution found by Phase I. Then, in Phase II the following restriction is added:

$$\sum_{a=1}^{N_A} Y_{ia} = \sum_{a=1}^{N_A} Y'_{ia} \quad \forall i = 1, \ldots, N^{\text{res}} \tag{7}$$

3.6 Solving Strategies and Approximations

The size of the problem in Phase I is usually huge, especially when no workload vector repeats in the LTWP, and thus the size of $\mathbb{L}$ is large. This can be alleviated if the LTWP is approximated by a quantized version.

Formally, given a set of quantization steps $\{Q_a\}$, one per application, the quantized long-term workload prediction, QLTWP, is defined as the sequence of vector loads $\bar{\boldsymbol{l}}_k$, for $k = 1, \ldots, T/t$, being:

$$\bar{\boldsymbol{l}}_k = \{\bar{l}_{ka}\} = \left\{ \left\lceil \frac{l_{ka}}{Q_a} \right\rceil Q_a \right\} \quad a = 1, \ldots, N_A \tag{8}$$

Note that, by taking the ceiling operator, QLTWP is a pessimistic approximation of LTWP, assuming workloads greater than or equal to the ones predicted. This is to ensure that the performance restriction in (2) is still fulfilled.

Since the quantization reduces the number of possible values that the workload can take, it increases the chance of observing repetitions of the same vector load $\boldsymbol{L}$, and thus the histogram of QLTWP, $\bar{H}(\boldsymbol{L})$, will have a smaller number of non-zero points than $H(\boldsymbol{L})$, i.e: the size of the effective quantized workload $\bar{\mathbb{L}}$ will be smaller (or equal in the worst case) than the size of the original effective workload $\mathbb{L}$.

Using QLTWP instead of LTWP the size of the problem can be thus reduced. The quantization steps $\{Q_a\}$ gives us control over the size of the problem, at the cost of possibly introducing suboptimality in the solution.

There is one choice of $\{Q_a\}$ which is particularly intesting because it does not introduce suboptimality in the solution. This is the case in which each Q_a is the greatest common divisor of the performances for that application among all instance classes, i.e:

$$Q_a = \gcd_i \text{perf}_{ia} \tag{9}$$

Using Q_a chosen as in eq. (9) the solution of Phase I gives the same values for Y_{ia} than in the case without quantization, but the quantized version is a smaller problem, easier to solve, as shown in the experimental results section.

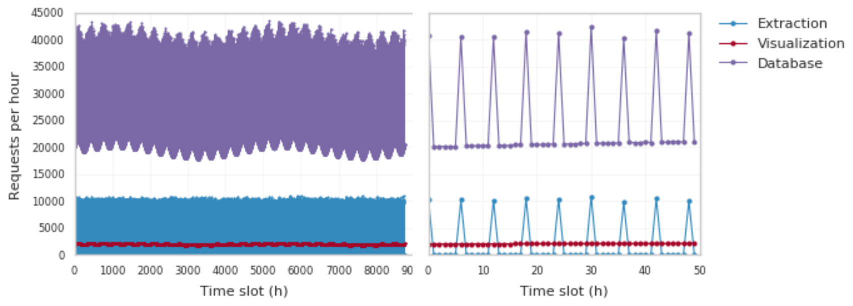

Fig. 1. Workload of the case study, for a year (left) and for the first 50 hours (right)

4 Experimental Results

In order to show how the technique proposed in this paper can solve problems that previous state-of-the-art techniques are not able to address, a synthetic case study is presented in this section. In this case study, a hypothetical analytics company uses three applications: a data extraction application that every six hours fetches the data from different external sources, an analysis application that the customers use and a database that is used by the extraction application to save the data and by the analysis application to carry out the analysis.

These three applications are executed in Amazon's EC2 cloud. The analytics company has statistics about the number of expected requests for the next year and wants to obtain the allocation with the minimum cost that fulfills the performance requirements. Fig. 1 shows the synthetic workload that has been generated to simulate the statistics from the company. As can be seen the three applications have different request patterns. In particular, the extraction application only executes every six hours; the analysis application exhibits periodic behaviour with daily, weekly and yearly cycles; finally, the database application workload is compounded from the other two application workloads, using different visit ratios to the database.

In order to have real prices and limits, we are going to assume that the applications have to be deployed in region US West (N. California) of Amazon's EC2 cloud, where there are three availability zones. There is a limit of 20 reserved VMs in each zone. In addition, there is a limit of 20 VMs for each type of on-demand instance. In order to provide a variety of options, VM types m3.medium, m3.large, c3.large and c3.xlarge have been selected as possible types to execute the application. Table 1 shows the performances and price for each VM type. The values of the performances have been generated synthetically, but the relation between them follows the relation between ECU (the performance metric used by Amazon) for each type. In addition, compute optimized VM types (c3.large and c3.xlarge) have been given more performance for the analysis application to provide a more interesting case study where different applications behave differently in different VM types.

Table 1. Performance and price of different VM types

	Application performance (rph)			Price ($/h)	
VM type	Extraction	Analysis	Database	On-demand	Reserved
c3.large	5750	30	18900	0.12	0.0766
c3.xlarge	10350	50	34020	0.239	0.154
m3.large	4600	20	15120	0.154	0.105
m3.medium	2300	10	7560	0.077	0.0532

Table 2. Quantization amounts used in phase I

Quantization amount	GCD multiplier	Number of variables	Quantization step Q_a (rph)		
			Extraction	Analysis	Database
0	N/A	91632	None	None	None
1	1	8544	1150	10	3780
2	3	2256	3450	30	11340
3	5	1200	5750	50	18900
4	7	672	8050	70	26460
5	10	720	11500	100	37800
6	15	384	17250	150	56700

As the goal of the experimentation is showing how this new technique compares to previous works, the case study has been analyzed with MALLOOVIA and LLOOVIA. However, since LLOOVIA can only handle one application, the three applications have been analyzed independently, generating three solutions, one per application, and the total cost has been computed as the sum of the cost of the three solutions. Notice that, since LLOOVIA considers the limits independently in each application, there is a risk that total number of VMs for the combined solution exceeds the limits, so the solution obtained with LLOOVIA would be unfeasible.

To study how quantization affects the results, the problem has been solved without quantization and with several amounts of quantization using different quantization steps, and to isolate the error introduced by quantization from the error in the workload predictions the same workload has been used as LTWP and STWP. In the first case, the quantization step for each application is the greatest common divisor (GCD) of the performance of each VM type. This quantization step does not introduce error. In the rest of the cases, the quantization step is multiplied by 3, 5, 7, 10 and 15, giving the values shown in Table 2. As can be seen, increasing the quantization steps reduces the number of variables in the optimization problem for phase I. Using the GCD, the number of variables decreases from 91632 to 8544.

Fig. 2 shows that in MALLOOVIA, the cost increases very little when the quantization step is increased (notice that the y axis starts in 45 000). As expected, when the quantization step is the greatest common divisor, the cost is the same as with no quantization. In the worst case, the cost is incremented in

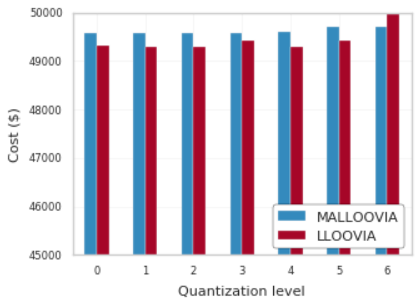

Fig. 2. Comparison of allocation costs in MALLOOVIA and LLOOVIA

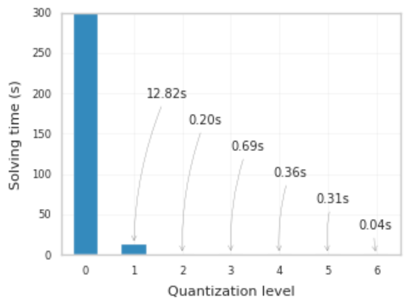

Fig. 3. Time required for solving phase I of MALLOOVIA

less than 0.3%. On the other hand, Fig. 3 shows that the solving time for phase I of MALLOOVIA is greatly reduced using quantization. Using the GCD, the solving time is one order of magnitude smaller than without quantization, but the solution obtained is also optimal.

Fig. 2 shows a result that may be unexpected at first: except when the quantization amount is maximum, costs are higher in MALLOOVIA than in LLOOVIA. As it was mentioned before, there is a reason that explains why, in fact, MALLOVIA is better: as LLOOVIA is not prepared for working with several applications, the allocation obtained by combining the three independently generated allocations does not respect the limits imposed by the providers and, thus, is not feasible.

This can be seen in Fig. 4, which shows the number and types of VMs allocated for each application in MALLOVIA and LLOOVIA for the 50 first hours. Reserved VMs are represented with different colors (depending on the type and zone), while on-demand VMs are depicted with levels of grey. It can be seen that LLOOVIA allocates 60 reserved VMs for the analysis application. That is the maximum number of reserved VMs that can be allocated with the 20 limit per region and the three regions available, and it is a valid solution when only that application is taken into account. However, when solving for the database application, LLOOVIA also allocates two reserved VMs, making the combined allocation not compliant with the limits imposed by the provider; thus, the combined solution is unfeasible. On the other hand, MALLOOVIA allocates 60 VMs between the three applications, obtaining a feasible solution.

In addition, MALLOVIA has another advantage: if for any time-slots the STWP for an application were smaller than the LTWP, some reserved instances for that application would not be needed and could be reused for another application, avoiding to hire new on-demand VMs and, thus, reducing the cost.

This case study has demonstrated that MALLOOVIA can solve problems that strategies not prepared for multi-applications solve incorrectly. In addition, it has shown that using quantization the solving time can be significantly reduced without increasing the cost significantly.

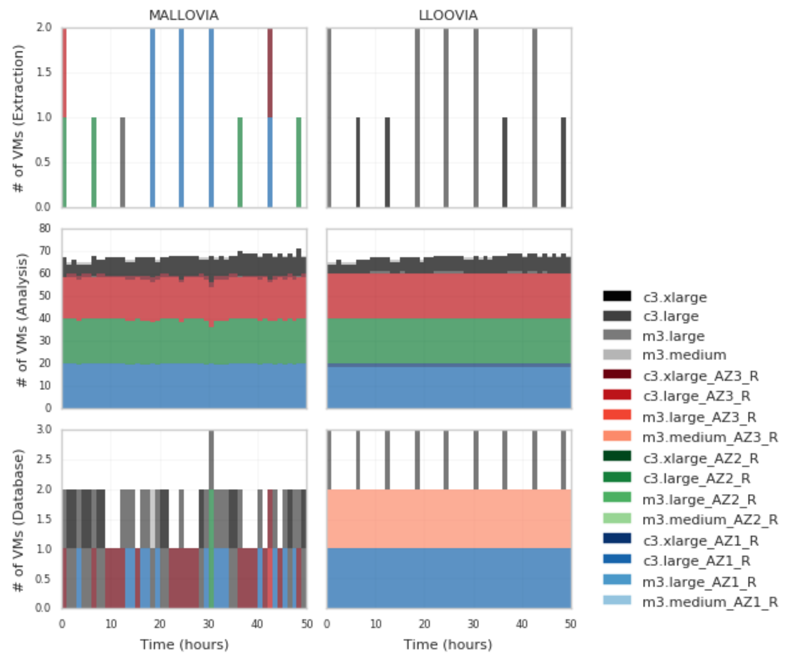

Fig. 4. Comparison of number of VMs allocated in MALLOOVIA and LLOOVIA. In the legend, the suffix "_R" indicates that the VM is reserved and "_AZn" indicates that it is deployed in availability zone n.

5 Conclusions and future Work

In this paper we have presented the MALLOOVIA allocation strategy. It enables the cloud user to find the most economical allocation that meets performance when the user wants to deploy several applications running simultaneously and for a long time period.

MALLOOVIA succeeds in representing the real characteristics found in the cloud market: different cloud providers, several types of VMs, constraints imposed by the providers to the number of VMs that can be simultaneously hired and pricing schemes. In addition, it supports multi-application deployments. The optimization problem was formulated using integer linear programming and solved in two phases: phase I obtains the number of reserved VMs to be hired at the beginning of the reservation period, and phase II allocates applications to the hired reserved VMs and obtains the number of extra on-demand VMs needed at each time slot.

MALLOOVIA uses an approximation method based on quantization, which reduces significantly the size and resolution time of the problem with practically no cost deviation for small quantization levels. In the experimental results, this strategy has shown that it is able to solve the allocation cost-minimization problem without violating the imposed restrictions, unlike previous strategies.

Future work focuses on improving the realism of the cloud model, for example taking into account extra constraints, such as the amount of memory or the number of cores required for an application, and studying how the VM performance variability within the same instance class impacts the results [13]. Another future work direction is to investigate new approximations to reduce the problem size without lost of accuracy.

Acknowledgments. This work was supported by the Spanish National Plan for Research, Development and Innovation [Project MINECO-15-TIN2014-56047-P].

References

1. Álvarez, P., Hernández, S., Fabra, J., Ezpeleta, J.: Cost Estimation for the Provisioning of Computing Resources to Execute Bag-of-Tasks Applications in the Amazon Cloud. In: Altmann, J., Silaghi, G.C., Rana, O.F. (eds.) Economics of Grids, Clouds, Systems, and Services, vol. 9512, pp. 65–77. Springer International Publishing, Cham (2016)
2. Amazon: Amazon EC2 pricing (2016), `https://aws.amazon.com/ec2/pricing/`
3. Amazon: Amazon EC2 - instance types (2017), `https://aws.amazon.com/ec2/instance-types/`
4. Bellur, U., Malani, A., Narendra, N.C.: Cost optimization in multi-site multi-cloud environments with multiple pricing schemes. In: 2014 IEEE 7th International Conference on Cloud Computing. pp. 689–696. IEEE (Jun 2014)
5. Chaisiri, S., Lee, B.S., Niyato, D.: Optimization of resource provisioning cost in cloud computing. IEEE Transactions on Services Computing 5(2), 164–177 (Apr 2012)

6. Díaz, J.L., Entrialgo, J., García, M., García, J., García, D.F.: Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing. Future Generation Computer Systems 71, 129 – 144 (2017)
7. Hu, M., Luo, J., Veeravalli, B.: Optimal provisioning for scheduling divisible loads with reserved cloud resources. pp. 204–209. IEEE (Dec 2012)
8. Khatua, S., Sur, P.K., Das, R.K., Mukherjee, N.: Heuristic-based optimal resource provisioning in application-centric cloud. CoRR abs/1403.2508 (2014)
9. Mireslami, S., Rakai, L., Wang, M., Far, B.H.: Minimizing Deployment Cost of Cloud-Based Web Application with Guaranteed QoS. pp. 1–6. IEEE (Dec 2015)
10. Nan, X., He, Y., Guan, L.: Optimal allocation of virtual machines for cloud-based multimedia applications. In: Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on. pp. 175–180. IEEE (Sep 2012)
11. Nodari, A., Nurminen, J.K., Frühwirth, C.: Inventory theory applied to cost optimization in cloud computing. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. pp. 470–473. ACM Press (2016)
12. Orbegozo, I.S.A., Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Cloud capacity reservation for optimal service deployment. In: CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization. pp. 52–59. IARIA (Sep 2011)
13. O'Loughlin, J., Gillam, L.: Performance Evaluation for Cost-Efficient Public Infrastructure Cloud Use. In: Altmann, J., Vanmechelen, K., Rana, O.F. (eds.) Economics of Grids, Clouds, Systems, and Services, vol. 8914, pp. 133–145. Springer International Publishing, Cham (2014)
14. Pietri, I., Sakellariou, R.: Cost-Efficient CPU Provisioning for Scientific Workflows on Clouds. In: Altmann, J., Silaghi, G.C., Rana, O.F. (eds.) Economics of Grids, Clouds, Systems, and Services, vol. 9512, pp. 49–64. Springer International Publishing, Cham (2016)
15. Ran, Y., Yang, B., Cai, W., Xi, H., Yang, J.: Cost-Efficient Provisioning Strategy for Multiple Services in Distributed Clouds. pp. 1–8. IEEE (May 2016)
16. Reddy, K.H.K., Mudali, G., Sinha Roy, D.: A novel coordinated resource provisioning approach for cooperative cloud market. Journal of Cloud Computing 6(1), 8 (2017)
17. Wang, W., Niu, D., Liang, B., Li, B.: Dynamic cloud instance acquisition via IaaS cloud brokerage. IEEE Transactions on Parallel and Distributed Systems 26(6), 1580–1593 (Jun 2015)
18. Yousefyan, S., Dastjerdi, A.V., Salehnamadi, M.R.: Cost effective cloud resource provisioning with imperialist competitive algorithm optimization. In: 2013 5th Conference on Information and Knowledge Technology (IKT). pp. 55–60. IEEE, IEEE (May 2013)