Analysis of the influence of per-second billing on virtual machine allocation costs in public clouds

José Luis Díaz, Joaquín Entrialgo, Javier García, Manuel García, Daniel F. García, *Member, IEEE*

Abstract—For a long time, the common billing time slot used by cloud providers was the hour, but recently they have changed it to one second with a minimum charge of one minute. In this paper, the impact of this change on virtual machine allocation strategies is analyzed. With the minimum one-minute charge, the state-of-the-art allocation strategies are no longer optimal. This paper proposes a new analytic model to obtain the optimal allocation that minimizes the total cost considering this new billing scheme. However, the optimization problem using this model is unmanageable due to its computational complexity. Therefore, other sub-optimal allocations strategies are proposed. The performance of these strategies (in terms of total allocation cost and computational effort) is analyzed in order to assess the influence of the time slot length. Allocation time slots of one hour, one minute and one second are compared. The experimental work is carried out using synthetic workloads based on two real public traces. The study concludes that the time slot of one minute offers the best trade-off between allocation cost and computational effort. The experimentation shows that the proposed strategy can save up to 14% over using the time slot of one hour.

Index Terms—Cloud computing, cost optimization, virtual machine allocation

1 Introduction

Cloud computing has become a mainstream technology of the IT industry. Among the different service models offered by cloud computing providers, in this paper we consider Infrastructure as a Service (IaaS), a model in which providers basically offer Virtual Machines (VM).

A transactional service in the cloud (such as a web service) requires the allocation of a set of VMs to support its workload. Transactional services represent one of the most important cloud workloads [1]. These workloads are given as requests per time unit. Recent studies [2] show a significant waste of resources in service deployments, highlighting the importance of cost optimization in VM allocations. Therefore significant research efforts have been carried out in this field.

An allocation strategy produces successive VM allocations to support a varying workload. A VM allocation is a set of VMs, indicating their types, the number of each type, and the pricing categories (on-demand or reserved) to which the VMs belong. In order to determine the instants at which an allocation strategy is applied, the time is usually divided in regular time slots. There are allocation strategies focused on the short and on the long term. The former produces VM allocations for the next time slot, while the latter generates allocations for all the time slots within a long period. The aim of long-term strategies is to take advantage of reserved VMs. Although in the past reserved VMs were only offered by Amazon [3], nowadays other major public providers (e.g., Microsoft, Google and Alibaba Cloud) also provide this category of VM [4], [5], [6]. Except for Alibaba Cloud, which offers a reservation period of one month, all these providers offer a minimum reservation period of one year.

The length of the time slots managed by allocation strategies is usually matched with the billing time slot. In the past, the billing time slot used by major cloud providers was the hour. However, on 2 October 2017, Amazon announced a major change in its billing policy: the new billing time slot became the second, with a minimum charge of one minute [7]. This means that once a VM is put in a running state, a minimum charge of one minute is applied, even when the VM remains in this state for less than one minute. Once the first minute has elapsed, the total number of seconds in the running state is charged. Microsoft Azure has recently changed from one hour to one minute (seconds are discarded), and Google and Alibaba Cloud also started billing by seconds in their IaaS service [8], [9].

Per-second billing with a minimum charge of one minute raises three important issues: 1) the impact on the size and complexity of the VM allocation problem, 2) the interdependence among successive time slots in relation with the calculation of the optimal allocation for each time slot, and 3) the necessity of workload traces with a resolution of one second and a length of one year (in order to deal with reserved VMs properly), for testing allocation algorithms.

With regard to issue 1), the reduction of the billing time slot to one second, in principle, also implies the reduction of the time slot used by the allocation strategy to one second. When reserved VMs are considered and the period of analysis is extended to one year, the number of time slots to be managed by the allocation strategy is extremely large. Thus the feasibility and the convenience of calculating a VM allocation for each second of a year must be analysed.

In relation to issue 2), the interdependence among successive time slots is caused by the minimum charge of one minute for each VM launched. To take this minimum charge into account, the allocation strategy should keep

• *Authors are with the Department of Computer Science, Oviedo University, Asturias, 33204 Gijón, Spain.*
E-mail: {jldiaz, joaquin, javier, mgarcia, dfgarcia}@uniovi.es

Manuscript received April 19, 2005; revised August 26, 2015.

track of allocation decisions in previous time slots, as well as the workload in future time slots. This interdependence among time slots may make the VM allocation problem extraordinarily complex, so it must be studied carefully.

Regarding issue 3), workload traces are a very useful tool in the research of VM allocation strategies. To this end, the availability of public traces is very important. However, the requirement of a per-second resolution with a length of one year raises important questions: are there public traces corresponding to transactional services with a per-second resolution and a length of one year? In the case of a trace with a resolution lower than one second, can it be transformed into a trace with a per-second resolution? What techniques can be used to this end? These questions must be analyzed.

In the research presented here, a new analytic model for an allocation strategy that considers per-second billing with a minimum charge of one minute is proposed. This model uses the same input parameters as a previous model, called Malloovia [10], but introduces a new problem formulation to take into account the restrictions imposed by the new per-second billing procedure. The analysis of the VM allocation problem using the proposed formulation reveals that the interdependence among successive time slots generates a huge number of variables, making the VM allocation problem unmanageable. Therefore, several approximated solutions using variations of the standard Malloovia algorithm are proposed and analyzed.

The research presented here provides the following main contributions:

1) A model of a VM allocation strategy with a time slot of one second and a minimum charge of one minute, called TWOS (Time-Window based Optimal Solver). The analysis of this model shows that the VM allocation problem with the minimum charge of one minute is unmanageable.
2) A comparative analysis of the calculation of VM allocations employing three different time slots: one hour, one minute and one second. The comparative analysis shows that the best choice is using a time slot of one minute, because it offers the best trade-off between the allocation cost and the computational effort required to find the allocation. In addition, the analysis reveals significant differences in cost among the VM allocations produced with the three time slot alternatives, depending on the shape of the workload to be managed. This comparative analysis was carried out using Malloovia, as well as a modified version of Malloovia called Malloovia Guided, but the insights obtained are general.
3) A set of traces with a resolution of one second and a length of one year. These traces were obtained applying diverse processing techniques to traces of real publicly available workloads. The traces obtained have been made available in a public repository, thus they can be used in future research works.

To the best of our knowledge, the research presented in this paper is the first to analyze the influence of the billing time slot length on the cost of allocation strategies, comparing the traditional per-hour billing with the new per-second billing.

The rest of this paper is organized as follows. Section 2 discusses the related work. In section 3, an analytic model to obtain the optimal cost, as well as other alternative approximations, are developed. Section 4 presents a set of experimental case studies that compare the performance of the different allocation strategies. Section 5 discusses several relevant results obtained in this research, as well as the limitations of the techniques developed. Finally, Section 6 provides the main conclusions of the paper.

2 Related Work

The latest reports from RightScale about the state of the cloud [2] emphasize the significant amount of wasted cloud spend, estimated between 27% and 35%. This has made users focus on cost and how it can be reduced.

In the literature, several papers study the improvement of VM allocation on IaaS, and the way to reduce the allocation cost. They can be divided in two groups according to how the workload of the cloud system is represented. A classification of the related work can be seen in Table 1. In this table, the main characteristics of each work in relation with our work are highlighted.

In the first group of papers [11], [12], [13], [14], [15], [16], [17], [18] and [19] the workload is represented as the number of VMs requested in each period. They obtain an optimal allocation, but they only solve part of the problem because the issue of determining the appropriate type and number of VMs required to serve the workload is not solved. This last aspect is itself another optimization problem. In order

TABLE 1
Comparison of this research with the related papers. The black circle means that the characteristic is supported, a white circle that it is not supported and a half-filled circle that it is partially supported.

	Multi-Cloud	Different app. types	Multiple VM types	VM Limits	Uses reserved VM	Real workload	Workload length	Workload resolution	Kind of workload
Chaisiri 2009 [11]	●	●	◐	◐	●	◐	N/A	N/A	VMs
Mark 2011 [12]	●	●	◐	◐	●	◐	5 mos.	N/A	VMs
Orbegozo 2011 [13]	○	○	○	○	●	●	3 yrs.	N/A	VMs
Chaisiri 2012 [14]	●	●	◐	◐	●	◐	N/A	N/A	VMs
Yousefyan 2013 [15]	●	●	◐	◐	●	○	—	—	VMs
Hwang 2014 [16]	○	○	●	○	●	●	2 mos.	5 min	VMs
Khatua 2014 [17]	●	○	○	○	●	●	6 mos.	sec	VMs
Nodari 2016 [18]	○	○	◐	○	●	◐	1 mo.	hour	VMs
Reddy 2017 [19]	●	○	○	○	●	○	—	—	VMs
Nan 2012 [20]	●	●	●	○	●	○	—	—	Req
Hu 2012 [21]	○	○	◐	○	●	○	—	—	Req
Bellur 2014 [22]	●	○	○	○	●	●	1 yr.	hour	Req
Wang 2015 [23]	○	○	○	○	●	●	1 mo.	μsec	Jobs
Li 2015 [24]	◐	○	○	●	●	●	1 mo.	sec	Req
Cao 2016 [25]	○	◐	●	○	●	○	—	—	Req
Delimitrou 2016 [26]	○	●	●	○	●	○	2 hrs.	sec	Req
Lloovia 2017 [27]	●	○	●	●	●	●	1 yr.	hour	Req
Malloovia 2017 [10]	●	●	●	●	●	○	1 yr.	hour	Req
This work	●	●	●	●	●	●	1 yr.	sec	Req

to obtain the minimum cost it is necessary to obtain the right type, minimum number of VMs and the best allocation simultaneously. If both optimization problems are solved separately, the final solution might not be globally optimal; therefore the papers in this group are not described in detail.

The second group of papers [20], [21], [22], [23], [24], [25], [26], [27] and [10] analyzes the cost optimization problem considering the workload as an arrival rate of requests that must be served using the allocated VMs. These papers will be discussed following three aspects, which correspond to the three groups of columns in Table 1.

The first aspect analyzed is whether the papers consider one or multiple clouds. Only [20], [22], [27] and [10] consider the allocation of VMs over several cloud providers, which is a more general solution. In [24] the authors take into account a solution based on a hybrid cloud.

Secondly, attending to the number of the applications to deploy and the properties of VMs in the problem, the analyzed papers follow different approaches:

- Number of applications supported. Most of the works consider a single application which is deployed on the VMs. This makes the analysis simpler. However in [20], [26] and [10] the authors consider multiple applications deployed on the set of VMs, providing a more complete resolution methodology. An intermediate case is [25] where three applications are considered.
- Types of VM and cloud provider constraints. Several works consider only one type of VM and no restrictions on the number of VM that can be leased, making the problem easier to solve. On the other hand, in [20], [25], [26], [27] and [10] the authors take into account different types of VM, making the study closer to real conditions. An intermediate case is studied in [21], where the authors consider a small set of VM types. Real cloud providers limit the number and types of VM that can be leased simultaneously. Only [24], [27] and [10] consider this constraint, although in the case of [24] it is only applied to one type of VM.

Finally, considering the characteristics of the test workload, less than half of the analyzed works are tested with real workloads: [22], [23], [24] and [27]. Among them, only two, [22] and [27], consider a workload length of one year and neither of these uses a resolution of seconds.

All the previously cited papers use a billing period of one hour in order to obtain the final cost. That is, if an on-demand VM is used for less than an hour, the user will be charged for the price of one hour. This paper studies how to obtain an optimal allocation working with actual conditions and the new one-second billing period.

3 System Model and Resolution

3.1 Introduction

The goal of the techniques presented here is to solve an optimization problem that takes as input the level of performance to be reached at each time slot by a set of applications, and generates as output a VM allocation to support the performance requirements of all the applications. The generated allocation minimizes the deployment costs of the applications, guaranteeing the required performance for each one of them. This is a problem similar to the one solved by Malloovia [10], which is an evolution of Lloovia [27].

The Malloovia model is independent of the length of the time slot. As long as the price, workload and performance (in requests per time slot) use the same time units, the model and implementation will find the optimal allocation for each time slot. This suggests that using one second as the time slot length (instead of one hour as in [10]) the same algorithm should provide the optimal allocation per second. However, the minimum charge of 60 seconds (applied by the providers using the per-second billing) invalidates the resolution approach of Malloovia. To overcome this limitation, a new model must be developed.

The purpose of this section is to present a VM allocation model for the case of one-second time slots with a minimum charge of 60 seconds. We refer to this model as TWOS (Time-Window based Optimal Solver). The complexity of the VM allocation problem dealt by TWOS is also investigated. In addition, other approximations for the VM allocation problem based on modifications of the standard Malloovia algorithm are analyzed and discussed.

3.2 Problem description

The inputs used by the TWOS model are the same as the ones used in Malloovia, which are detailed in [10]. We provide here a short summary of the notation, in which, to simplify the exposition and without loss of generality, we assume that a single application is being considered, so we omit the application sub-index. Table 2 summarizes this notation.

The set of all possible VMs is modelled using the concept of "instance class" (IC_i), which represents a family of VMs of the same price (p_i) per time unit, performance (perf_i), and features, which are deployed in the same "limiting set" (LS_i). The limiting set is a generalization of regions and availability zones and imposes a limit on the total number of running instances (of any instance class) in that set. All VMs of the same instance class are considered reserved or on-demand depending on the Boolean rsv_i. An example of instance class in Amazon EC2 is an on-demand m4.xlarge instance in region us-east-1. The total number of instances of the same instance class which can run simultaneously is limited to max_i, regardless of the limiting set to which they belong. There are a total of M different instance classes. Note that this model requires that the price per time unit of the instance class does not depend on the number of time slots the instance is used; therefore it is not applicable to providers that apply discounts based on the time the instance is running, such as Google Cloud Engine [5].

The planning period (usually a year) is divided in N time slots t_k, and the expected workload for each time slot (l_k) is assumed to be known in advance. The problem to solve is, given a prediction for the workload, to find a mix of VMs of different types, for each time slot, that can provide enough performance to satisfy the workload for that time slot, while minimising the total cost in the planning period.

The allocation problem is solved in two phases. Phase I requires a "Long Term Workload Prediction" (LTWP) which

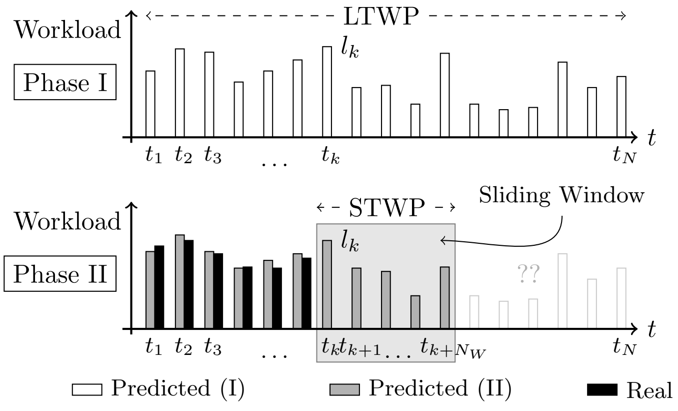


Fig. 1. Phases and workload prediction in the model

contains the expected workload for each time slot in the planning period (see upper graph of Fig. 1). The solution of Phase I provides the optimal allocation for each time slot, which includes the optimal number of on-demand and reserved instances to be active in each time slot (for the reserved instances, the number will be the same for all time slots, since reserved instances are kept running at all times, because they are paid for, whether or not they are used). If the LTWP were a perfect prediction, the complete planning for the whole period would result from Phase I. However it is unrealistic to assume a perfect prediction for such a long period, so a second phase is required. In Phase II, the number of reserved instances from Phase I is used, but the number of on-demand instances is recomputed at each time slot for a sliding window of N_W time slots (see bottom graph of Fig. 1), using a "Short Term Workload Prediction" (STWP), which is the expected workload for the next N_W time slots. The STWP is assumed to be much shorter and more accurate than the LTWP, since it is focused on the short term.

Despite the independence from the time slot size, Malloovia assumes that on-demand instances are not charged for the time slots in which the instance is not active. However, currently all providers which use seconds as a billing unit impose a minimum charge of 60 seconds, which invalidates the assumption of Malloovia.

TABLE 2
Notation for the model inputs

	Cloud infrastructure
M	Number of different instance classes
IC_i	Instance class i
rsv_i	Boolean: is the instance class reserved?
p_i	Price of the instance class i
perf_i	Performance of the instance class i
LS_i	Limiting set for instance class i
$\max_i$	Maximum allowed running instances for instance class i
	Time and workload
N	Number of time slots in the planning period of Phase I
N_W	Number of time slots in the planning window of Phase II
t_k	k-th time slot
l_k	Expected workload for time slot t_k
LTWP	Long-term workload prediction (N values, for Phase I)
STWP	Short-term workload prediction (N_W values, for Phase II)

3.3 Time-Window based Optimal Solver (TWOS)

To find the allocation with the minimum cost, it is important to realize that shutting down one VM when it is not being used does not always decrease the cost of the allocation. If the machine is required again in a few seconds, it may be cheaper to leave it running and pay for the extra seconds than to shut it down and pay for a minimum of 60 seconds when it is activated again. This means that the formulation of the problem must take into account the extra cost of running a machine for less than 60 seconds. The strategy used here is to include restrictions in the model which ensure that each VM is running for at least 60 seconds.

To ease the notation and the formulation, we initially make the following simplifications:

1) No reserved instances are used. All VM types are on-demand.
2) No global limits per region or availability zone are enforced (so no limiting sets are considered). Only the limits per instance class, $\max_i$, are considered.
3) No multi-application is considered. A single application is run, the same in all VMs.

We will see that, even with this extremely simplified model, the size of the associated integer problem is very large, rendering it unsolvable in practice.

3.3.1 TWOS problem model

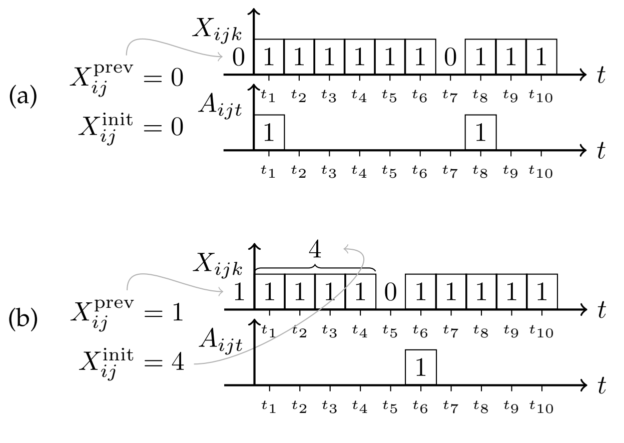
Table 3 provides a summary of the notation used in this section. Each instance that could be potentially activated will have a binary variable associated to it for each time slot t_k. We denote this variable by X_{ijk}, with $i = 1, \dots, M$, $j = 1, \dots, \max_i$, $k = 1, \dots, N$, and it represents the state of the machine at that time slot ($X_{ijk} = 1$ means that instance j of instance class IC_i is active in time slot t_k, while $X_{ijk} = 0$ means it is not active).

In order to take into account the cost of the first minute in a general way, the parameter $T_{\min}$ is introduced. It represents the number of time slots which are paid for as a minimum once the machine starts. In current practice $T_{\min} = 60$ when the time slot is one second.

The special consideration of a minimum price for machines which run fewer than $T_{\min}$ time slots could be introduced in the function that computes the deployment cost, but this would make the function non-linear. It is

TABLE 3
Notation in the TWOS formulation

X_{ijk}	Boolean variable: is the j-th instance of instance class i active during time slot t_k?
A_{ijk}	Boolean variable: was the j-th instance of instance class i started up at time slot t_k?
$T_{\min}$	Integer value: minimum number of time slots which are charged once an instance is activated.
X_{ij}^{prev}	Boolean value: state (active or inactive) of instance j of instance class i at the time slot previous to the planning period
X_{ij}^{init}	Integer value: number of time slots to keep active instance j of instance class i at the beginning of the planning period



(a)
$X_{ij}^{\text{prev}} = 0$
$X_{ij}^{\text{init}} = 0$
X_{ijk}
0 1 1 1 1 1 1 0 1 1 1
A_{ijt}
1 1
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10}
t
(b)
$X_{ij}^{\text{prev}} = 1$
$X_{ij}^{\text{init}} = 4$
4
1 1 1 1 1 0 1 1 1 1 1
1

Fig. 2. Illustration of the meaning of some problem variables. (a) Scenario in which machine X_{ij} was not active in previous window. (b) Scenario in which machine X_{ij} was active in previous window and still has 4 time units remaining. Both scenarios use $T_{\min} = 6$ for simplicity.

preferable to express this case as additional restrictions for the problem, to be explained later. This simplifies the cost function, which can be expressed linearly as:

$$\text{Cost} = \sum_{i=1}^{M} \sum_{j=1}^{\max_i} \sum_{k=1}^{N} X_{ijk} \cdot p_i \tag{1}$$

Eq. (1) is the objective function to be minimized, with the following restrictions:

Performance. In each time slot t_k, the total performance given by allocation $\{X_{ijk}\}$ must be at least equal to the expected workload, l_k, for that time slot.

$$\sum_{i=1}^{M} \sum_{j=1}^{\max_i} X_{ijk} \cdot \text{perf}_i \geq l_k \qquad k = 1 \dots N \tag{2}$$

Minimum execution time of $T_{\min}$ time slots. Once a VM instance is started, it must remain active for the following $T_{\min}$ time slots (because it will be charged anyway).

To capture this in the formulation it is necessary to introduce new variables in the problem. A_{ijk} is a binary variable which will be 1 if instance j of instance class IC$_i$ was started up at t_k, i.e., when $(X_{ijk} = 1) \wedge (X_{ij,k-1} = 0)$ (or, equivalently, when $X_{ijk} - X_{ij,k-1} = 1$) and 0 otherwise. This behaviour for A_{ijk} is enforced by adding the following restrictions to the problem:

$$X_{ijk} - X_{ij,k-1} < 1 + A_{ijk} \qquad \begin{array}{l} k = 1 \dots N, \\ i = 1 \dots M \\ j = 1 \dots \max_i \end{array} \tag{3}$$

$$X_{ijk} - X_{ij,k-1} > 2(A_{ijk} - 1) \qquad \begin{array}{l} k = 1 \dots N, \\ i = 1 \dots M \\ j = 1 \dots \max_i \end{array} \tag{4}$$

A_{ijk} is therefore a Boolean which detects the t_k at which instance j of instance class IC$_i$ is started. If $A_{ijk} = 1$, then from this t_k on, and for the next $T_{\min}$ time slots, the machine should remain active (i.e., $X_{ijk} = 1$). This implication can be modelled with the following set of restrictions:

$$X_{ijk'} + 1 - A_{ijk} \geq 1 \qquad \begin{array}{l} k' \in [k, k + T_{\min}] \\ k = 1 \dots N \\ i = 1 \dots M \\ j = 1 \dots \max_i \end{array} \tag{5}$$

The rationale behind this equation is that, when $A_{ijk} = 1$, the inequality $X_{ijk'} + 1 - 1 \geq 1$ can only be true if $X_{ijk'} = 1$, so the value of this variable is forced to 1 by the solver (and thus the machine is forced to be active) for the different values of k' which represent the next $T_{\min}$ slots.

Previous state. The problem must include information about the state of each VM in the time slot previous to the first one, because restrictions (3) and (4) require the value for $X_{ij,k-1}$. We assume that the previous state of all machines is given in the input values $\{X_{ij}^{\text{prev}}\}$. This allows us to set the values for all X_{ijk} for $k = 0$ (note that k starts at 1 for the remaining formulae).

$$X_{ij,0} = X_{ij}^{\text{prev}} \qquad \begin{array}{l} i = 1 \dots M \\ j = 1 \dots \max_i \end{array} \tag{6}$$

As an example, Fig. 2 shows two scenarios, one (a) in which the machine X_{ij} was not active at the end of the previous period, and another one (b) in which it was. Therefore, the value of A_{ij} at the first time slot is different in each scenario. To simplify the figure, a value of $T_{\min} = 6$ instead of 60 was used.

Forced state. Finally, the problem can also include information to set some instances as active at the beginning of the period to solve. For example, referring to the second scenario in Fig. 2, which used $T_{\min} = 6$, suppose that the instance was started 2 time slots before the end of the previous planning period. Since this machine is charged for 6 time slots, we can keep it active for the first 4 time slots of the current period, to leverage its performance which will be paid for regardless. The input values $\{X_{ij}^{\text{init}}\}$ contain, for each instance j of each instance class IC$_i$, the number of time slots in which this instance is kept active from the beginning of the current period. For the given example, $X_{ij}^{\text{init}} = 4$.

Given this information as input, the state of the appropriate machines is set in the following restriction:

$$X_{ijk} = 1 \qquad \begin{array}{l} k = 1 \dots X_{ij}^{\text{init}} \\ j = 1 \dots M \\ i = 1 \dots \max_i \end{array} \tag{7}$$

In summary, the allocation problem can be formulated as an integer programming problem, whose variables (X_{ijk}, A_{ijk}) are restricted to $[0, 1]$, with (1) as the objective function to be minimized, and restrictions (2), (3), (4), (5), (6) and (7).

3.3.2 Phase I (TWOS1)

The above formulation can provide the optimal allocation for the whole year, but the size of the problem makes it intractable for practical purposes. For example, using one-second time slots, Phase I would use $N = 60 \times 60 \times 24 \times 365 = 31\,536\,000$ time slots. Each time slot k uses $2Q$ variables (Q for each X_{ijk} and another Q for each A_{ijk}),

Q being the total number of possible running instances, which can be computed as $Q = \sum_{i=1}^{M} \max_i$. For example, considering $M = 6$ instance classes and a limit of $\max_i = 20$ for all i, this would mean $Q = 6 \times 20 = 120$, and therefore the number of variables per time slot would be $2Q = 240$. The total number of variables in this example would be in the order of 7.5×10^9.

Techniques to reduce the size of the problem, such as the histogram grouping used in Malloovia, cannot be used in this new formulation because the $T_{\min}$ restriction introduces dependencies between time slots, as well as the need to track the state of each instance in each instance class separately. For this reason, the TWOS formulation cannot be used in practice to solve Phase I, even for the simplified model that does not include reserved instances nor global limits per region and considers a single application.

3.3.3 *Phase II (TWOS2)*

For Phase II, it seems that the size of the problem is reduced, because the number of time slots to consider is only N_W (not the whole year). In addition, the use of $\{X_{ij}^{\text{prev}}\}$ and $\{X_{ij}^{\text{ini}}\}$ as initial conditions for the problem allows us to take into account the solutions (allocations) of previous time slots as additional restrictions for the current time slot, hence accounting for the $T_{\min}$ restriction. However, an important question remains: how many time slots (N_W) must be considered in the STWP in order to find an optimal allocation for the next time slot?

Malloovia uses a single time slot STWP ($N_W = 1$), but this is clearly not optimal when considering $T_{\min}$ billing. Consider for example that the workload for the next time slot decreases, and that the performance requirements could be met using fewer instances than in the previous time slot. In this case, the solver of the integer problem will "decide" to deactivate some of the running instances for the next time slot. However, if a few time slots later the workload increases, the deactivated instances could be necessary again. If they were still in their first minute, reactivating these instances would incur a greater cost than leaving them running. Without knowing the workload prediction for some future time slots, it is not possible to decide the optimal solution for the current time slot, that is, a *sliding window* of some size $N_W > 1$ must be used as STWP.

Determining the appropriate size of this window, N_W, is not a trivial question. A value of $N_W = T_{\min}$ may seem reasonable at first, but the decision to stop a machine or leave it running in the next time slot could still depend on the workload after the end of that window. In fact, it can be shown that for any value of N_W, the optimal allocation for the next time slot t_k could depend on the workload in time slots after t_{k+N_W}.

This result means that no optimal allocation can be found in Phase II unless the STWP is as long as the LTWP (i.e., $N_W = N$). In practice, Phase II can still be carried out using a small window (for example $N_W = 2T_{\min}$), but the allocation found by the solver would generally be sub-optimal. Depending on the workload characteristics, the difference with the optimal cost could be negligible.

Nevertheless, even for small windows such as $N_W = 2T_{\min}$, this formulation still incurs in a prohibitive number of variables, $2QN_W$. For example, when $Q = 120$ as in the example for Phase I, and using a window size of $2T_{\min}$ with $T_{\min} = 60$, the number of variables of the integer problem is $2 \times 120 \times 2 \times 60 = 28800$. This is still a very large problem, which must be solved in under a second, every second, to produce the allocation for the next time slot.

For the same reasons explained for Phase I, the number of variables cannot be reduced using Malloovia techniques. In practice, TWOS formulation cannot be used for Phase II either, except for problems with a very small number of instance classes and low limits, $\max_i$.

Since the optimal solution cannot be found in practice, the following sections explore some approximations which use Malloovia techniques to reduce the size of the problem, making it affordable at the expense of non-optimality (i.e., higher costs than the theoretical TWOS solution).

3.4 Malloovia-per-second approximation (MaPs)

The first idea is to use Malloovia with one second as the time slot, ignoring the minimum charge for the first minute. By removing the minimum charge, the dependencies among different time slots are also removed. This greatly reduces the number of variables of the problem in two ways:

1) Machines of the same instance class, IC_i, are grouped in a single problem variable, which is not boolean, but integer. At each time slot t_k, variable X_{ik} represents the total number of machines of instance class IC_i to be active in that time slot. Using this technique, rather than having $\max_i$ variables per instance class and time slot, a single variable per instance class and time slot is used (which takes values between 0 and $\max_i$). This can be reduced even further in the case of reserved instance classes, because the number of reserved instances of the same instance class is constant for all time slots, so a single variable per instance class Y_i is enough to contain the number of reserved instances of instance class IC_i for any time slot.
2) Time slots with the same workload will have the same optimal allocation, so a histogram can be built to store the number of times each load level appears in the LTWP, and it can be used in the equation of the global cost. Using this technique, the problem has an alternative formulation, which in most cases requires a smaller number of variables, while providing the same optimal allocation. In this new formulation, one variable, X_{iL}, can be used to represent the number of active instances of instance class IC_i in any time slot in which the expected workload is L. This reduces the number of variables, since only the different workload levels have to be considered, instead of the different time slots.

Using this formulation, the total number of variables in the problem is $M \times |L|$, M being the number of different instance classes and $|L|$ the number of different predicted workload levels. Consider the same example given in Section 3.3.2, which required approximately 7.5×10^9 variables using TWOS1 formulation. In this example, $M = 6$. Considering, for example, that the workload per-second varies between 0 and 223, the histogram would have 223

TABLE 4
Notation in the Mallovia formulation

X_{ik}	Integer variable: number of instances of on-demand instance class i which are active at time slot t_k
X_{iL}	Integer variable: number of instances of on-demand instance class i which are active at any time slot whose workload is L
Y_i	Integer variable: number of instances of reserved instance class i which are active at any time slot

values at maximum, and the problem would require only $6 \times 223 = 1138$ variables using Malloovia (Phase I) formulation. Table 4 provides a short summary of the notation used in this subsection.

3.4.1 Phase I (MaPs1)

Thanks to the above techniques, Phase I of the problem can have a manageable size and can be solved in a reasonable time. However, the cost function used in this formulation is not correct, since it does not consider the minimum charge of one minute. The solution found by the linear programming solver could include activations and deactivations of instances within the same minute, instead of keeping them active for at least one minute. This means that the optimal cost found by the solver is not realistic, so we will refer to it as $C^*(\text{MaPs1})$, to distinguish it from the actual cost, $C(\text{MaPs1})$, that the provider would charge for deploying such an allocation. Both costs would be the same only when the cloud providers do not charge for the one minute minimum. Nevertheless C^* can be used as a lower bound on the optimal cost, which could theoretically be found by the TWOS formulation, but cannot be known in practice.

The actual cost of the solution found by Malloovia can be computed by simulating the activations and deactivations of the instances, and performing the correct computation of the one-minute charge each time they are activated. This actual cost is generally much higher than the optimum. The optimal cost, denoted by $C(\text{TWOS1})$, will be bounded by these costs: $C^*(\text{MaPs1}) \leq C(\text{TWOS1}) \leq C(\text{MaPs1})$.

Note that even if the MaPs1 solution is not optimal, part of the solution can still be useful, in particular the values of Y_i, which represent the number of reserved instances of each instance class to be purchased.

3.4.2 Phase II (MaPs2 and MaPs2Guided)

Malloovia uses $N_W = 1$, i.e., the STWP contains the expected workload for the next time slot only. As was seen in Section 3.3.3, this inability to know the future makes it impossible to decide whether to keep the running instances active when they are no longer necessary in the next time slot but might be needed again soon.

However, the Malloovia solver for Phase II (MaPs2) can be "guided" to some extent by an external algorithm that tries to minimize the number of unnecessary reactivations (see Algorithm 1). We will refer to this approach as MaPs2Guided, or simply "Malloovia Guided".

The goal of this algorithm is to ensure that once a new instance is started, it remains active for at least the next $T_{\min}$ time slots. The algorithm keeps a set of $active\_vms$, which stores the number of instances of each instance class

Algorithm 1 MaPs2Guided algorithm

```
1: $active\_vms \leftarrow \varnothing$
2: for all timeslots do
3:     $prealloc \leftarrow$ get_prealloc($active\_vms$)
4:     $l \leftarrow$ get_next_workload_prediction()
5:     $alloc \leftarrow$ malloovia_solve_timeslot($l$, $prealloc$)
6:     $active\_vms \leftarrow$ update_active($active\_vms$, $alloc$)
```

which was activated in the previous $T_{\min}$ time slots, and the uptime of each one (i.e., the number of time slots each instance has been running since it was started up for the last time). Using this information, the function get_prealloc() builds the preallocation, $prealloc$, as the minimum number of instances of each instance class that must be kept active in the next time slot (because their uptime is still $\leq T_{\min}$). This preallocation is used by malloovia_solve_timeslot(), which is the implementation of the solver for the integer problem in the Malloovia model, modified to include additional restrictions to obey the preallocation. For example, to ensure that at least 3 instances of the class IC_2 are active in the next time slot k, the restriction $X_{2,k} \geq 3$ can be used as part of the restrictions for that time slot. Once the solver returns the optimal allocation for the next time slot ($alloc$), the method update_active() examines this allocation and compares it with the $active\_vms$ set. If new instances are started, new entries with uptime $= 0$ are added to the set $active\_vms$. The uptime of all elements in the set $active\_vms$ is increased, and the elements with an uptime greater than $T_{\min}$ not used as part of the optimal $alloc$ are removed.

MaPs2Guided ensures that the solver will leverage the performance given for the pre-allocated instances as part of the solution for the next $T_{\min}$ time slots. This reduces the number of unnecessary re-activations in $T_{\min}$. However, since N_W is still 1, it cannot see the future; therefore, if an instance has aged $T_{\min}$ slots and is not required for the next time slot, it will not be part of $alloc$, and thus it will be removed from the $active\_vms$ set and shut down, although it could still be cheaper to keep it running to avoid the cost of an upcoming reactivation. Nevertheless, in general MaPs2Guided would improve the MaPs1 allocation: even if the LTWP were a perfect prediction, the MaPs1 solution could include the shut down and reactivation of instances in their first $T_{\min}$ slots, which MaPs2Guided would avoid. Hence $C^*(\text{MaPs1}) \leq C(\text{TWOS1}) \leq C(\text{MaPs2Guided}) \leq C(\text{MaPs1})$

3.5 Malloovia-per-minute approximation (MaPm)

When a one-minute time slot is used in Malloovia, all instances in the solution run for at least one minute, or not at all. Thus, there is no need to take the $T_{\min}$ restriction into account, and two main advantages are gained:

1) In Phase I (MaPm1), there is no need to introduce artificial constraints to ensure that all instances run for at least one minute.
2) In Phase II (MaPm2), no "guiding" algorithm is needed.

This approach is therefore very simple to implement, since no modifications to the Malloovia implementation are required. In addition, it is more realistic from a practical point of view, because a new allocation is computed and deployed every minute instead of every second.

The only drawback to this approach is that the cost of the solution may not be optimal. For example, if an instance is required only 70 seconds a year, TWOS1 would keep it active for exactly 70 seconds, while MaPm1 would keep it active for 120 seconds. Therefore, $C(\text{TWOS1}) \leq C(\text{MaPm1})$. In general, how $C(\text{MaPm2})$ compares to $C(\text{MaPs2Guided})$ depends on how the workload varies within the minute, as will be explored in the experimentation section.

4 Experimentation

4.1 Methodology

In this section, we test how the allocation strategies proposed in the previous section work in terms of the cost of the best allocation obtained and the time required to find it. The goals of the experimentation are:

1) Assessing the practical feasibility of dealing with very large workloads when using one-second resolution traces.
2) Comparing the cost and the computation time of the different allocation strategies for these workloads.
3) Analyzing how the shape of the workload affects the performance of the allocation strategies.

To achieve these goals, realistic workloads with data of one year with one-second resolution are needed for Phase I. One year is required in order to take into account the periodic behaviour that real workloads usually have, including daily, weekly and seasonal cycles. One-second resolution is required to take into account the influence of the one-minute minimum charge and the workload size challenge.

Our literature review has not revealed any workload trace which fulfills both requirements for transactional systems. Typically, when workloads have one-second resolution, they are short (only hours, months at most). On the other hand, when they are one year long, they have less resolution. Thus, we have followed this methodology:

1) Analyze real workloads.
2) Synthesize traces that fulfill the requirements above from the real workloads analyzed.
3) Test the performance of the allocation strategies with the synthetic traces obtained.

Section 4.2 describes how the workload traces were synthesized, Section 4.3 details the experimental setup, Section 4.4 presents the experimental design used and Section 4.5 describes the results of the experiments.

4.2 Workload traces

The most interesting publicly available traces for the experimentation in this paper are:

1) Wikipedia [28]: it gives the total number of requests per hour to the Wikipedia for several years. This allows us to study intra-year cyclic behaviour.
2) FIFA World Cup [29]: it gives a log with each access to the website of the World Cup 98 for three months. From this, the number of requests per second can be obtained. This allows us to study how variations of requests per second, per minute and per hour affect the different allocation strategies.

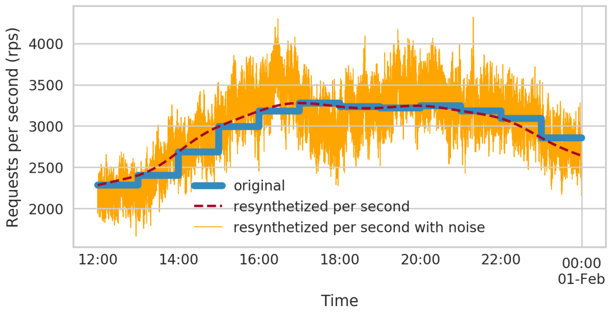


Fig. 3. Twelve hours of the original and the resynthesized Wikipedia trace with and without noise

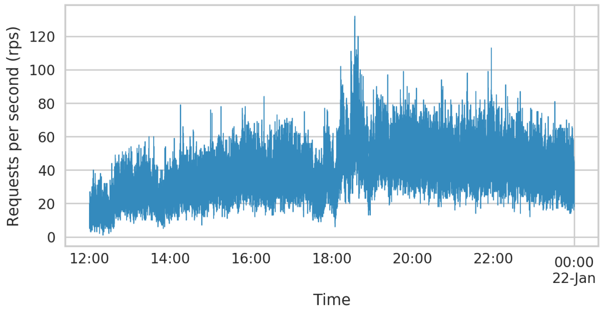


Fig. 4. Twelve hours of the FIFA World Cup trace

In order to have one-year long traces with one-second resolution, the following methodology was used:

- Wikipedia: a Fourier analysis for the data of 2014 was carried out. This provided the frequential components that represent the cyclic behaviour captured by the trace. From this Fourier series, a new trace with one-second resolution was generated, expanding it with zeros and carrying out an inverse Fourier transform. Fig. 3 shows 12 hours of the original trace with a thick blue line and the resynthesized trace with a dashed red line (the thin orange line, which includes synthetic noise, will be considered later). As can be seen, the number of requests per second within each minute is considered constant in the original trace, but changes smoothly in the resynthesized one.
- FIFA World Cup: as the trace was only three months long, a one-year trace was created by repeating the original trace four times. Fig. 4 shows a fragment of twelve hours of the trace. It can be seen that there is significant variance per second and per minute.

In order to study how the level of workload variation within the second and within the minute influences the results of the allocation strategies, pink noise [30] with different sizes and frequencies was added to the synthesized Wikipedia trace. Each element x in the trace is modified following this expression:

$$y = (1 + K_m P_m/3 + K_s P_s/3)x \tag{8}$$

where y is the new value, x is the original value, P_m is pink noise sampled per minute and resynthesized by second, K_m

TABLE 5
Allocation strategies tested. The first character, M, indicates use of Malloovia variations. The next two characters indicate the units used for Phase I and the remaining characters indicate the strategy for Phase II

Name	Phase I	Phase II
Mh1h2	Hours (Malloovia)	Hours (Malloovia)
Mh1m2	Hours (Malloovia)	Minutes (MaPm2)
Mh1s2g	Hours (Malloovia)	Seconds (MaPs2Guided)
Mm1m2	Minutes (MaPm1)	Minutes (MaPm2)
Mm1s2g	Minutes (MaPm1)	Seconds (MaPs2Guided)
Ms1s2	Seconds (MaPs1)	Seconds (MaPs2)
Ms1s2g	Seconds (MaPs1)	Seconds (MaPs2Guided)

TABLE 6
Performance and price of different VM types

VM type	on demand ($/h)	reserved ($/h)	perf (rph)
c4.large	0.124	0.079338	32800
c4.xlarge	0.249	0.159703	65600
m4.large	0.117	0.079338	26650
m4.xlarge	0.234	0.158790	53300

is a coefficient for the noise amplitude per minute, P_s is pink noise sampled per second and K_s is a coefficient for the noise amplitude per second. The standard pink noise generator produces values roughly between -3 and 3; thus, it is divided by 3 so that the peaks are adjusted to be between -1 and 1. The thin orange line in Fig. 3 shows the resynthesized Wikipedia trace with noise when $K_m = 0.01$ and $K_s = 0.10$. As can be seen, it is similar to the one shown in Fig. 4, which comes from a real trace.

4.3 Experimental setup

A set of experiments was carried out to evaluate the different allocation strategies summarized in Table 5. The strategy that gives the optimal solution following the TWOS model presented in Section 3.3.1 was not tested because even with short traces of two minutes it did not finish in a reasonable time (two days), so it cannot be used in practice with one year traces.

A mix of VM types in one Amazon region with three availability zones was selected as an example to test the allocation strategies. The performances for each instance class are shown in Table 6. They were generated taking into account the relative ECUs (Amazon's performance metric [31]) of each instance class. The traces had to be scaled down so that there were feasible allocations, taking into account the limits in this region, i.e., 20 reserved instances in each availability zone and 20 on-demand instances for the region.

All the experiments were run on an Intel Core i7-4790 with 32 GBytes of RAM running Ubuntu Server 16.04 with Python 3.6.3, CBC 2.8.12 and Malloovia 1.1.0 [32], which is a new version including modifications to allow the Malloovia Guided strategy.

4.4 Experimental design

The cost obtained with strategies taking into account different time units is influenced by the shape of the trace, i.e., the average workload level and the workload variability. For instance, if the number of requests per second were constant during each hour, it would make no difference computing an allocation per hour, per minute or per second.

TABLE 7
Experimental design for the Wikipedia trace

Factor	Values
K_m	0.01, 0.10
K_s	0.01, 0.10
Load level	0.01, 0.05, 0.1

In order to test how the attributes of the trace influence the performance of each allocation strategy, an experimental design for the Wikipedia trace with the parameters shown in Table 7 was carried out. The noise coefficients were used to resynthesize the Wikipedia trace four times with different noise levels (0.10 as a value similar to the one observed in the real trace of the FIFA World Cup and 0.01 as an order of magnitude smaller value). Then, 12 traces were obtained by multiplying each of these four resynthesized traces by the three scaling factors for the load level indicated in the table: 0.1 is used as the highest value because it takes the workload close to the maximum performance that can be achieved by the selected instances taking into account the region limits; 0.01 is selected as an order of magnitude smaller value; and an intermediate value of 0.05 is added because the preliminary experiments showed that the results depended significantly on the workload level.

The twelve combinations of factors in Table 7 were used to generate resynthesized traces per second. In order to obtain traces per hour and per minute for the strategies that need them, the per-second traces were resampled taking the maximum number of requests per second within each hour or minute, respectively. The maximum was selected instead of the average to guarantee that the solution meets the per-second performance requirements.

For the FIFA World Cup trace, as the noise is not synthetically generated, the only factor studied was the workload level, using scaling factors of 0.02, 0.10 and 0.20, selected using the same rationale as in the Wikipedia trace.

These traces have been made public[1] so that the research community can use them in future endeavours.

For Phase I, a whole-year prediction is required (LTWP). Phase II only requires the prediction for one time slot (STWP). However, in order to compare the annual cost of different strategies in Phase II, the values of the STWP for each time slot of the whole year are also required. This is simulated in the experiments by using the same trace for Phase II as the one used for Phase I. Note that this would be the same as having a perfect prediction as LTWP. In [27] the implications of a non-perfect prediction are studied.

4.5 Experimental results

4.5.1 Wikipedia trace

To compare the performance of the different strategies with regard to cost, the allocation for the experiments in Table 7

1. https://github.com/asi-uniovi/malloovia-data-seconds

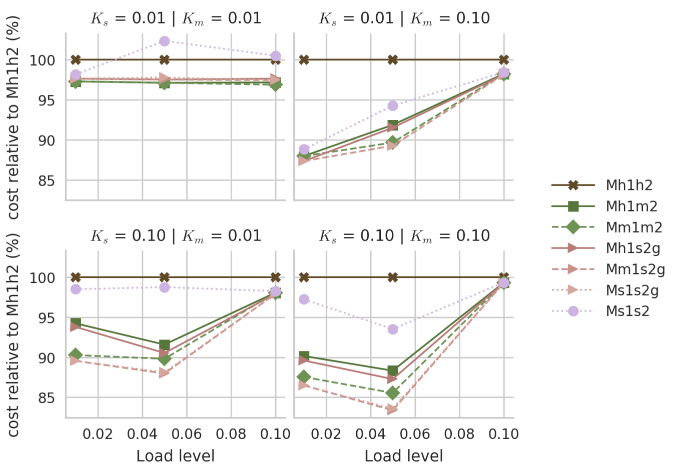


Fig. 5. For the Wikipedia trace, percentage of cost with respect to the cost obtained by Mh1h2

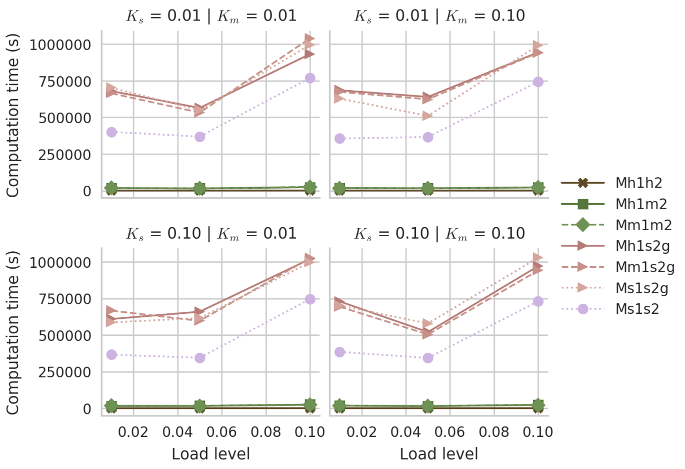


Fig. 6. For the Wikipedia trace, time to compute Phase II for all the time slots in the whole period (one year) using different allocation strategies

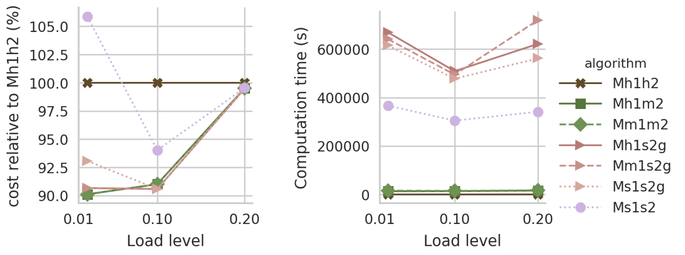


Fig. 7. For the FIFA World Cup trace, percentage of cost with respect to the cost obtained by Mh1h2 (left) and time to compute Phase II for all the time slots in the whole period (right) using different allocation strategies

was obtained for each strategy in Table 5. These allocations are not shown here for the sake of brevity.

The cost of each allocation as a percentage of the cost for the allocation obtained with Mh1h2 was computed. This makes it possible to assess the savings that can be obtained by each strategy with respect to previous state-of-the-art strategies that use hours as time slots in both phases.

Fig. 5 shows that, in most cases, all proposed strategies are better than Mh1h2. This shows the importance of using strategies that take advantage of smaller time slots. However, the next worst strategy after Mh1h2 in most cases is Ms1s2, which uses seconds as time slots for both phases, but does not take into account the one-minute minimum charge during the optimization process. When this minimum is taken into account to compute the actual cost, this is higher than the cost obtained by other strategies. This shows the importance of taking the minimum one-minute charge into account during the optimization.

The best strategies in terms of cost are the ones that use minutes or seconds for Phase I and use seconds with Malloovia guided in Phase II, i.e., Mm1s2g and Ms1s2g. They can provide up to 17% savings over Mh1h2. The next best strategy is Mm1m2, which uses minutes in Phases I and II and can obtain up to 14% saving over Mh1h2. However, as shown in Fig. 6, strategies Mm1s2g and Ms1s2g have a much higher computational cost than Mm1m2 because they have to compute the allocation for many more time slots. In the best case, this increase in computation time only results in approximately 3% extra savings over the allocation obtained by Mm1m2.

The previous analysis has shown that all the strategies using the new version of Malloovia can deal with one-year long traces, although the ones that use seconds in Phase II take a significantly longer time.

Regarding the influence of the parameters of the trace, Fig. 5 shows that there is a very significant influence of the workload scaling factor, but only at the maximum level, i.e., when it is 0.10. In this case, all strategies obtain similar costs because all allocation strategies have to select a large number of reserved instances in order to handle the maximum workload without going over the limits of on-demand instances. These reserved instances are also able to handle the noise without requiring on-demand instances when the workload is low. Thus, using on-demand instances is rarely needed and all strategies obtain a similar cost.

When the performance required is not close to the limits, i.e., for scaling factors 0.01 and 0.05, Fig. 5 shows that when both the variability per second and per minute is low (top left graph), the difference between all the strategies is small. The other graphs show that when the variability per minute or per second increases, the savings obtained with strategies that use time slots smaller than the hour can be increased.

4.5.2 *FIFA World Cup trace*

In order to study the strategies with real noise instead of synthesized noise, they were tested with the the FIFA World Cup trace, repeated four times to extend to one year and scaled with factors of 0.02, 0.10 and 0.20. Fig. 7 (left) shows the costs obtained with the different strategies. As with the Wikipedia trace, all strategies except for Ms1s2 always obtain savings over Mh1h2. Except for the highest scaling factor, the savings are around 10%. Finally, as before, the closer the workload to the limits of the provider, the closer the results of all strategies.

Fig. 7 (right) shows the computation time for solving the whole period with the FIFA World Cup trace. The results are similar to those obtained with the Wikipedia trace: the strategies using seconds in Phase II have an enormously higher computation time. These results reinforce the conclusions obtained with the Wikipedia trace.

5 Discussion and Limitations

The previous section shows important results about the impact on costs of using different time slots to compute the allocation. They do not depend on the particular solving technique used (such as Malloovia Guided), but are general insights into the allocation problem.

These results show, first, that the savings when considering time slots smaller than the hour are significant. In addition, they demonstrate that strategies that do not take the minimum billing time into account incur in higher costs. Finally, they indicate that, from a practical point of view, cloud customers with a workload close to the limits of the regions they are using should negotiate an increase in the limits with the provider or should consider distributing their workload over more regions, so that they can use more on-demand instances and take advantage of the reduction in cost that per-second billing can provide.

The techniques presented in this paper have some limitations. First, a forecast of the workload is required. Although this concern is out of the scope of this work, there are many works that propose techniques for forecasting the workload in cloud computing ([33], [34], [35]). Moreover, [27] studied the influence of errors on the LTWP, by using naive predictors, and found that costs were not much higher than when using a perfect predictor, and were much smaller than not using a predictor and serving the workload using only on-demand instances. Using two phases allows the scheduler to reduce the extra costs when the prediction for Phase I is not perfect.

Another problem is that the performance of an instance type, which the models consider constant, may vary depending on the load of the underlying physical machine. This variability is much more significant for IO-bound than for CPU-bound applications [36]. Our allocation techniques are more suitable for the latter, and therefore, less affected by performance variability.

Another apparent limitation is that the techniques assume that VMs can be started up instantaneously, which is not realistic. However, this could be taken into account simply by computing the scheduling ahead of time.

The techniques introduced in this work have not been tested in a real environment. This is part of the future work.

6 Conclusions and Future Work

In this paper, we have analyzed several allocation strategies considering per-second billing with a one-minute minimum charge. To the best of our knowledge, this is the first scientific work addressing this problem.

We have developed and implemented the formulation for the optimal allocation (TWOS) algorithm. Since this algorithm is impractical due to its high computational complexity, we have proposed alternative strategies based on a state-of-the-art technique, Malloovia, with different time slots. We have also proposed a new way of using Malloovia, guiding the optimization process so that the costs generated by the one-minute minimum charge are reduced.

There is a lack of year-long traces with per-second resolution, so we have synthesized and released a set of traces for the research community to use. We have tested seven allocation strategies with these traces and found that the guided strategies using seconds for Phase II could save up to 17% over allocations that use hours, as previous state-of-the-art strategies did. However, as the computation time of guided strategies is very large and it is not currently practical to deploy a new allocation each second, we propose using one-minute time slots (Mm1m2), which obtains similar savings with much less computation time. Furthermore, this allocation strategy is more in line with the time currently required to deploy a VM.

The influence of the workload variability and average level on the performance of each allocation strategy was also analyzed. We have found that the cost is highly influenced by the relationship between the average workload and the available capacity due to the limits of the provider.

As future work we plan to test the proposed techniques in a real environment. In addition, the model will be extended to consider the cost of network traffic and storage.

Acknowledgments

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

References

[1] M. C. Calzarossa, M. L. Della Vedova, L. Massari, D. Petcu, M. I. M. Tabash, and D. Tessera, "Workloads in the Clouds," in *Principles of Performance and Reliability Modeling and Evaluation*, L. Fiondella and A. Puliafito, Eds. Cham: Springer International Publishing, 2016, pp. 525–550.
[2] RightScale, "Rightscale 2019 state of the cloud report," RightScale, Tech. Rep., 02 2019.
[3] Amazon, "Amazon EC2 pricing," 2018. [Online]. Available: https://aws.amazon.com/ec2/pricing/
[4] Microsoft, "Azure - linux virtual machines pricing," 2018. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/
[5] Google, "Committed use discounts," 2018. [Online]. Available: https://cloud.google.com/compute/docs/instances/signing-up-committed-use-discounts
[6] Alibaba, "Alibaba cloud pricing," 2019. [Online]. Available: https://www.alibabacloud.com/pricing
[7] Amazon, "Announcing amazon EC2 per second billing," 2017. [Online]. Available: https://aws.amazon.com/about-aws/whats-new/2017/10/announcing-amazon-ec2-per-second-billing/
[8] Google, "Extending per second billing in google cloud," 2017. [Online]. Available: https://cloudplatform.googleblog.com/2017/09/extending-per-second-billing-in-google.html
[9] A. Cloud, "Pay-as-you-go ecs instances support per-second billing," 2017. [Online]. Available: https://www.alibabacloud.com/notice/ECS-0928
[10] J. Entrialgo, J. L. Díaz, J. García, M. García, and D. F. García, "Cost Minimization of Virtual Machine Allocation in Public Clouds Considering Multiple Applications," in *Economics of Grids, Clouds, Systems, and Services*, C. Pham, J. Altmann, and J. A. Bañares, Eds. Cham: Springer International Publishing, 2017, vol. 10537, pp. 147–161.
[11] S. Chaisiri, Bu-Sung Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Services Computing Conference*. IEEE, Dec. 2009, pp. 103–110.
[12] C. C. T. Mark, D. Niyato, and T. Chen-Khong, "Evolutionary optimal virtual machine placement and demand forecaster for cloud computing," in *International Conference on Advanced Information Networking and Applications*. IEEE, Mar. 2011, pp. 348–355.
[13] I. S. A. Orbegozo, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Cloud capacity reservation for optimal service deployment," in *CLOUD COMPUTING 2011, The Second International Conference on Cloud Computing, GRIDs, and Virtualization*. IARIA, Sep. 2011, pp. 52–59.

[14] S. Chaisiri, B. S. Lee, and D. Niyato, “Optimization of resource provisioning cost in cloud computing,” *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 164–177, Apr. 2012.
[15] S. Yousefyan, A. V. Dastjerdi, and M. R. Salehnamadi, “Cost effective cloud resource provisioning with imperialist competitive algorithm optimization,” in *2013 5th Conference on Information and Knowledge Technology (IKT)*, IEEE. IEEE, May 2013, pp. 55–60.
[16] R.-H. Hwang, C.-N. Lee, Y.-R. Chen, and D.-J. Zhang-Jian, “Cost Optimization of Elasticity Cloud Resource Subscription Policy,” *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 561–574, Oct. 2014.
[17] S. Khatua, P. K. Sur, R. K. Das, and N. Mukherjee, “Heuristic-based optimal resource provisioning in application-centric cloud,” *CoRR*, vol. abs/1403.2508, 2014.
[18] A. Nodari, J. K. Nurminen, and C. Frühwirth, “Inventory theory applied to cost optimization in cloud computing,” in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM Press, 2016, pp. 470–473.
[19] K. H. K. Reddy, G. Mudali, and D. Sinha Roy, “A novel coordinated resource provisioning approach for cooperative cloud market,” *Journal of Cloud Computing*, vol. 6, no. 1, p. 8, 2017.
[20] X. Nan, Y. He, and L. Guan, “Optimal allocation of virtual machines for cloud-based multimedia applications,” in *Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on*. IEEE, Sep. 2012, pp. 175–180.
[21] M. Hu, J. Luo, and B. Veeravalli, “Optimal provisioning for scheduling divisible loads with reserved cloud resources,” in *2012 18th IEEE International Conference on Networks (ICON)*. IEEE, Dec. 2012, pp. 204–209.
[22] U. Bellur, A. Malani, and N. C. Narendra, “Cost optimization in multi-site multi-cloud environments with multiple pricing schemes,” in *2014 IEEE 7th International Conference on Cloud Computing*. IEEE, Jun. 2014, pp. 689–696.
[23] W. Wang, D. Niu, B. Liang, and B. Li, “Dynamic cloud instance acquisition via IaaS cloud brokerage,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1580–1593, Jun. 2015.
[24] S. Li, Y. Zhou, L. Jiao, X. Yan, X. Wang, and M. R.-T. Lyu, “Towards Operational Cost Minimization in Hybrid Clouds for Dynamic Resource Provisioning with Delay-Aware Optimization,” *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 398–409, May 2015.
[25] Z. Cao, J. Lin, C. Wan, Y. Song, Y. Zhang, and X. Wang, “Optimal Cloud Computing Resource Allocation for Demand Side Management,” *IEEE Transactions on Smart Grid*, pp. 1–13, 2016.
[26] C. Delimitrou and C. Kozyrakis, “HCloud: Resource-Efficient Provisioning in Shared Cloud Systems,” in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM Press, 2016, pp. 473–488.
[27] J. L. Díaz, J. Entrialgo, M. García, J. García, and D. F. García, “Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing,” *Future Generation Computer Systems*, vol. 71, pp. 129 – 144, 2017.
[28] Wikitech, “Analytics/data/pagecounts-raw — Wikitech,” 2015. [Online]. Available: https://wikitech.wikimedia.org/w/index.php?title=Analytics/Data/Pagecounts-raw
[29] Ita, “The internet traffic archives: Worldcup98,” 1998. [Online]. Available: http://ita.ee.lbl.gov/html/contrib/WorldCup.html
[30] F. Miller, A. Vandome, and M. John, *Federal Standard 1037C*. VDM Publishing, 2010.
[31] Amazon, “What is an ec2 compute unit,” 2018. [Online]. Available: https://aws.amazon.com/ec2/faqs/
[32] J. L. Díaz, J. Entrialgo, M. García, J. García, and D. F. García, “Malloovia,” 2018. [Online]. Available: https://github.com/asi-uniovi/malloovia/releases/tag/v1.0.0
[33] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, “Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications’ QoS,” *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449–458, Oct. 2015.
[34] G. Kecskemeti, Z. Nemeth, A. Kertesz, and R. Ranjan, “Cloud workload prediction based on workflow execution time discrepancies,” *Cluster Computing*, Oct. 2018.
[35] W. Iqbal, A. Erradi, and A. Mahmood, “Dynamic workload patterns prediction for proactive auto-scaling of web applications,” *Journal of Network and Computer Applications*, vol. 124, pp. 94–107, Dec. 2018.
[36] P. Leitner and J. Cito, “A Study of Performance Variation and Predictability in Public IaaS Clouds,” *ACM Transactions on Internet Technology*, vol. 16, no. 3, pp. 15:1–15:23, Aug. 2016.

José Luis Díaz received his B.Sc. and Ph.D. degrees in Electrical Engineering from the University of Oviedo, Gijón, Spain, in 1990 and 2003, respectively. He is an Associate Professor in the Department of Informatics at the University of Oviedo. His main research up to 2009 was in the field of real-time systems, with contributions to the probabilistic analysis of these systems. His current research interests are in real-time systems and cloud computing.

Joaquín Entrialgo received the Ph.D. in computer science from the University of Oviedo, Spain, in 2006. He is an Associate Professor with the Informatics Department of the University of Oviedo, where he teaches computer architecture, datacenter infrastructure and network security. His main research interests are green computing and cloud computing.

Javier García received the PhD degree in electrical engineering from the University of Oviedo, Spain, in 1999. He is an associate professor with the Informatics Department of the University of Oviedo. From 2008 to 2016, he was the head of the IT Area in the Office of the Vice-Rector for IT and Infrastructures of the University of Oviedo. From 2016, he coordinates the management of the computing infrastructure of the schools and departments in the University. His current research interest is in cloud computing.

Manuel García received his PhD degree in Electrical Engineering in 2003. He is currently an Associate Professor in the Department of Computer Science at the University of Oviedo. He has taken part in research projects on real-time inspection systems based on vision techniques, and performance analysis of HFC networks. His current research interests are computer performance evaluation and cloud computing.

Daniel García is a full professor in the Department of Computer Science and Engineering at the University of Oviedo, Spain, where he leads the area of computer engineering. His current research interest is in the area of self-adaptive computer systems, including cloud, multimedia and computer vision systems. During the last 20 years, he has lead many research projects and coauthored 70 articles in journals and more than 145 papers in conferences and workshops. He is a member of the IEEE Computer Society.