Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing

José Luis Díaz*, Joaquín Entrialgo, Manuel García, Javier García, Daniel Fernando García

Dept. of Informatics, University of Oviedo, Campus de Viesques, 33204 Gijón (Spain)

Abstract

In the Cloud Computing market, a significant number of cloud providers offer Infrastructure as a Service (IaaS), including the capability of deploying virtual machines of many different types. The deployment of a service in a public provider generates a cost derived from the rental of the allocated virtual machines. In this paper we present LLOOVIA (Load Level based OpimizatiOn for VIrtual machine Allocation), an optimization technique designed for the optimal allocation of the virtual machines required by a service, in order to minimize its cost, while guaranteeing the required level of performance. LLOOVIA considers virtual machine types, different kinds of limits imposed by providers, and two price schemas for virtual machines: reserved and on-demand. LLOOVIA, which can be used with multi-cloud environments, provides two types of solutions: (1) the optimal solution and (2) the approximated solution based on a novel approach that uses binning applied on histograms of load levels. An extensive set of experiments has shown that when the size of the problem is huge, the approximated solution is calculated in a much shorter time and is very close to the optimal one. The technique presented has been applied to a set of case studies, based on the Wikipedia workload. These cases demonstrate that LLOOVIA can handle problems in which hundreds of virtual machines of many different types, multiple providers, and different kinds of limits are used.

Keywords: cloud computing, virtual machine allocation, multi-cloud, reserved instances, cost optimization

1. Introduction

Cloud Computing is a well established industry oriented to providing computing and storage resources to support information technology services and applications. Especially in the case of on-line services, which usually experience significant variations in their workload patterns, Cloud Computing can provide the required flexibility in resource provisioning to deploy fast and cost effective solutions.

The Cloud Computing market is currently operated by a significant number of companies, which are known as public providers, because they offer their services to the public in general. Notable examples of public providers are Amazon Web Services (AWS) [1], Microsoft Azure [2] and Google Cloud Platform [3].

The deployment of a service in a public provider is always supported by a group of virtual machines (VMs), which host the required software for the service, and are deployed on the virtualization infrastructure of the provider. The offer of VMs by a provider is usually known as Infrastructure as a Service (IaaS). VMs can be deployed using different combinations of computational resources (such as virtual cores, memory, etc.), which are frequently referred to as VM types. For example, Microsoft Azure offers VM types such as the Standard_D1 (providing 1 vCore and 3.5 GB of memory) and the Standard_D4 (with 8 vCores and 28 GB of memory). Each VM type has an established price per hour and can reach a determined level of performance for a given application.

With regard to pricing, two categories of VMs can be considered: on-demand VMs and reserved VMs. The on-demand category, which is offered by all public providers, implies that the user is only charged for the time the VM is running. The reserved category is only supplied by some public providers, such as Amazon EC2. Reserved VMs establish a commitment between the user and the provider. The user agrees to pay for the VM for a fixed term (a year, for example), regardless of whether the VM is used or not. As a compensation, the user obtains a significant discount from the provider. The impact of using reserved VMs in the cost of a service may be significant, so they should be included as a possible alternative in the design of new services.

A crucial aspect of a service design is the definition of an allocation strategy for the VMs required to support the service workload. To introduce the VM allocation prob-

*Corresponding author

Email addresses: jldiaz@uniovi.es (José Luis Díaz), joaquin@uniovi.es (Joaquín Entrialgo), mgarcia@uniovi.es (Manuel García), javier@uniovi.es (Javier García), dfgarcia@uniovi.es (Daniel Fernando García)

Paper with DOI https://doi.org/10.1016/j.future.2017.02.004

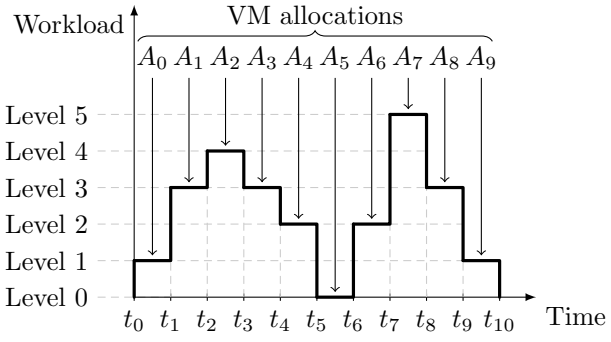

Figure 1: Example of workload evolution of a service and VM allocations to support the workload

lem, a simple example, shown in Fig. 1, is used. The thick line depicted in the figure represents the workload evolution of a service, measured, for instance, in requests per second. As can be seen in the figure, the workload reaches six different levels. In order to provide the required support for this workload, the service designer must define a suitable allocation strategy for the VMs used in the implementation of the service. As shown in Fig. 1, in each time slot t_i, the allocation strategy must provide an allocation A_i suitable for the workload level in that time slot. The allocation A_i represents the number and types of the VMs to be deployed at the corresponding time slot, as well as the providers in which the VMs are deployed. A usual objective for the allocation strategy is to provide the required computational power for the service in each time slot, minimizing the cost.

In order to produce the allocation depicted in Fig. 1, a possible solution would be to calculate each allocation A_i on-line, by applying the allocation strategy at regular time slots. In this way, suitable allocations would be generated periodically during the operation of the service. This type of allocation strategy is only appropriate when only on-demand VMs are used, because machines can be started and stopped as required, and the user is only charged when the machines are in execution. However, this strategy lacks the economic benefits that might be provided by reserved VMs.

In order to take advantage of reserved VMs, other considerations must be taken into account. Reserved VMs are made available to users for long periods: one or three years in the case of Amazon EC2 [4]. From an economic perspective, reserved VMs are profitable when they are in execution doing useful work for a major part of their reservation period. Analyzing Fig. 1, several workload levels can be observed. Considering the interval between t_0 and t_{10} as the reservation period, Level 1 is present for 90% of the time of this period (only in the interval between instants t_5 and t_6 this level is not present), Level 2 is present for 70% of the time, Level 3, for 50%, and so on. Intuitively it is easy to see that reserved VMs would definitively be profitable when they are used to support Level 1 of the workload, and with high probability in the case of Level 2. However, in the case of the workload levels that occupy less time in the reservation period (Levels 3, 4, and so on), the possibility of obtaining profitability with the use of reserved VMs is lower, because of the cost of overprovisioning. These workload levels would probably be better supported by on-demand VMs. From a general point of view, for a long-running on-line service, a suitable combination of on-demand and reserved VMs will generate the most profitable VM allocation.

In order to use reserved VMs, a major problem to be solved by an allocation strategy is the management of a workload prediction for the whole reservation period, as well as the calculation of an allocation for each time slot of this period. If we consider a reservation period of one year and a time slot of one hour, 8760 allocations must be calculated, which implies solving a huge allocation problem.

Two other factors complicate the allocation strategy even more: (1) the enormous number of VM types offered in a multi-cloud environment, and (2) the limits imposed by providers on the VMs running within a site (region or availability zone). With regard to (1), it must be pointed out that each public provider supplies dozens of VM types, each one of them with a different price and level of performance, and belonging to a different category (reserved or on-demand); so the solution space to be explored by the allocation strategy may be enormous. In relation to (2), limits impose constrains on the allocation strategy, making the obtaining of a solution more difficult. Limits are defined in different forms such as the maximum number of reserved VMs that can be launched in an availability zone (20, in the case of Amazon EC2 [5]), or the maximum number of virtual cores to be used in a region (20, as established by Azure [6]).

When multiple providers, VM types and limits exist, finding which levels of workload should be supported by reserved VMs and which others should be supported by on-demand ones is a complex task. In this paper, we present LLOOVIA (Load Level based OptimizatiOn for VIrtual machine Allocation), an allocation strategy to determine the optimal allocation of the VMs required by a service in a multi-cloud environment. LLOOVIA is formulated as an optimization problem that takes the level of performance to be reached by the service as input and generates a VM allocation as output. The calculated allocation minimizes the service cost, guaranteeing the required performance for the service, and complying with the limits imposed by the provider.

LLOOVIA is designed to be applied in two different timeframes, one of them corresponding to the long term, and the other, to the short term. In the long-term period, the optimal allocation of reserved VMs is determined. Then, for each short-term period, the optimal allocation of on-demand VMs (which are required to supplement the computing capacity provided by the reserved VMs) is calculated. The long-term and the short-term allocation strategies could be executed by a scheduler, whose

implementation in not considered in this paper. The long-term strategy is executed off-line, for example, before the start-up of a service, or when a new reservation period begins. The short-term one is executed on-line at regular time intervals, for example, in periods of one hour, coinciding with the billing period of some important providers, such as Amazon EC2 [7].

LLOOVIA provides the following main contributions:

- To the best of our knowledge, LLOOVIA is the first allocation strategy that takes into account the following three aspects simultaneously:
  1. Using as input a service workload expressed in a common performance metric, such as requests per second, in contrast to other works that express the workload directly as the number of required VMs ([8], [9], [10], [11] or [12]).
  2. Managing an allocation space made up of any number of VM types, supplied by any number of cloud providers, and considering regions, availability zones and different kinds of limits within providers.
  3. Generating a VM allocation for a whole reservation period, distributing the allocation in regular time slots.

- Two types of solutions are provided:
  1. The optimal solution based on a novel approach that uses histograms of load levels.
  2. An approximated solution that groups load levels in bins to compute the histogram. Using this approximation, the size of the problem may be significantly reduced, and a solution for the allocation strategy may be obtained in a much shorter time.

LLOOVIA is implemented as a Python module, which is provided in the supplementary material. Likewise, the data and the implementation of all the case studies presented in this paper are also provided, facilitating the repeatability of this research. LLOOVIA has been tested using real workloads encompassing a whole year, as is shown in the case studies based on real traces from Wikipedia.

LLOOVIA has the following limitations:

- Variable prices for the VM types are not considered for the long-term, that is, the prices of VM types remain constant during the whole period for which the long-term allocation strategy is applied.

- The service (or component of a service) to be allocated with LLOOVIA must be considered as perfectly parallelizable, with infinite horizontal scalability. Therefore, it must be composed of an arbitrary number of identical tasks running on a mix of VM types. This architecture perfectly matches with a stateless component of a service (such as the stateless web server component of an on-line service), which is a common element in cloud computing patterns [13].

The rest of this paper is organized as follows. Section 2 discusses the related work. In section 3, the model for the cloud environment and the proposed solution for the allocation strategy are explained, while section 4 explains the methodology to use the strategy using a synthetic example. Section 5 presents a set of experimental case studies that show the benefits of the proposed approach. Finally, Section 6 provides the main conclusions of the paper.

2. Related work

Cloud Computing has evolved quickly in recent years and in that time it has developed broadly [14]. As a result, the number of technical works in the literature is vast. The problem considered in our paper is related to IaaS, one of the fastest growing fields in Cloud Computing. In [15] the authors present a survey on this subject and the open challenges in resource management. Within the IaaS field our paper focuses on an optimization problem of the cloud resources, while taking into account the three factors indicated below.

1. Our optimization problem is analyzed from the cloud service user's perspective, rather than the provider's point of view. A broad coverage of the cloud service user's perspective can be found in [16].
2. In the optimization problem considered in our work, the characteristics of the cloud application are known and the analysis is focused on obtaining the best allocation of VMs in the cloud providers. Several approaches to solve this kind of problem are analyzed in [17]. A different strategy, not considered in our work, would consist of analyzing the profile and characteristics of the cloud applications in order to find the resources required. A survey on this type of strategy can be found in [18].
3. Our optimization problem considers economic cost as the objective in the optimization analysis. Other works use other requeriments as the objectives of their optimization problems, such as task execution time, resource utilization, energy consumption or availability. When more than one requirement must be met simultaneously the multi-criteria analysis must be used, as explained in [19] and [20]. Some examples where this kind of analysis is used for VM allocation are [21], [22] or [23].

The optimization problem treated in our paper focuses on cost, while guaranteeing the fulfillment of performance requirements. Cost is one of the most important factors from the user's perspective. This factor is complex enough because in the cloud market it is possible to find many

cloud providers who offer different types of VM both in performance and price. Furthermore, there are several types of pricing policies and discounts. All these factors make it difficult for the user to find the most cost-effective VM selection and allocation.

In the literature, there are several papers that approach the optimization problem of VM allocation in a similar way to our paper, considering also the three factors indicated above. These papers are enumerated in Table 1. This table also classifies and compares the related works with the work done in our paper. The aspects considered are the following:

- Multi-cloud. Whether the method can be applied to several cloud providers simultaneously.
- Richness of the proposed method. This topic involves the following elements: whether the method can support different types of applications all of them with different resource requirements; whether the method can manage different VM types, or it is limited to a unique type; whether the method considers limits imposed by the cloud provider; and finally, whether the method supports reserved VMs in addition to on-demand ones.
- Optimality of the result obtained. This aspect indicates whether the method provides the optimal number of reserved and/or on-demand VMs needed.
- When the analysis is carried out. Whether the method is applied before the execution of the real workload (offline analysis) or while the real workload is being served (on-line analysis).
- Workload. How the workload is expressed, and whether the method has been validated using real workloads, and if so what their extension was.

In Table 1 a black circle means that the work fulfills the corresponding characteristic, a white circle that it misses it and a half-filled circle that the characteristic is partially supported. Finally, the papers in Table 1 are organized in two groups according to the kind of workload they consider.

The first group of papers analyzed ([8], [9], [10], [11] and [12]) represents the workload as the number of VMs requested in each period. In transactional applications, in which the workload is typically represented in requests per second, the approach of these works solves only part of the problem, namely, to find the optimal allocation of VMs to providers. However, this approach does not address the issue of determining the appropriate type and number of VMs. This issue in itself is another optimization problem and the works in this group do not explain how to solve it, so it is left to the analyst. In addition, if both optimization problems are solved separately, the final solution might not be globally optimal.

In [8] the authors propose an optimal cloud resource provisioning algorithm 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 cloud applications, as long as each application uses the same VM type, that is, one application can not be supported simultaneously by different VM types. Also, 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, this is not the way in which VMs are really purchased: they are leased from 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 the algorithm the authors refer to a realization phase, when the real demand is applied. In this phase the real demand is compared with the number of reserved VMs and if more VMs are needed they are hired on-demand. However, in [8] it is not clear how this process is carried out.

In [9], the authors extend their previous work by applying heuristics instead of stochastic integer programming, in order to reduce the resolution time. In this new work they also consider several reservation periods. It is a more general model, but the previous shortcomings remain.

While the previously cited articles use mainly linear programming and some approximations to solve the problem in a reasonable time, the following articles rely on heuristics as a way to reach a solution. In [10] and [11], the authors have carried out a similar work, and follow the model described in [9]. In both cases they minimize the cost of a resource placement in a cloud environment. They consider reserved and on-demand price schemes, and both solve the problem in two phases, one of them to obtain the prediction of the VM demand, and the other to find the reserved and the on-demand requirements. The VMs are again specified as a set of resources, rather than VM types. The main difference between [10] and [11] is the heuristic used to carry out the optimization process, and how they make the demand prediction.

The last related work where the workload is given as number of requested VMs is [12]. In this work, 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 the stochastic 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. In [31] the author explains the developed model in more depth. The main drawback of this model is that it is limited to only one VM type and cloud provider, and the model does not consider any limit in the number of VMs that can be hired.

The second group of papers analyzed in the Table 1

	Multi-Cloud	Different application types	Multiple VM types	VM Limits	Considers reserved VM	Optimizes # reserved VM	Optimizes # on-demand VM	Offline analysis	On-line analysis	Real workload	Workload length	Kind of workload
Chaisiri 2009 [8]	●	●	◐	◐	●	●	◐	●	◐	◐	N/A	VMs
Chaisiri 2012 [9]	●	●	◐	◐	●	●	◐	●	◐	◐	N/A	VMs
Mark 2011 [10]	●	●	◐	◐	●	●	◐	●	◐	◐	5 months	VMs
Yousefyan 2013 [11]	●	●	◐	◐	●	●	◐	●	◐	○	—	VMs
Nodari 2016 / 2015 [12]	○	○	◐	○	●	◐	○	●	○	◐	one month	VMs
Tordsson 2012 [24]	●	○	●	○	○	○	●	○	●	●	N/A	Jobs
Lucas-Simarro 2013 [25]	●	○	●	○	○	○	●	○	●	○	—	Req/seg
Wang 2015 [26]	○	○	○	○	●	◐	●	○	●	●	one month	Jobs/resources
Bellur 2014 [27]	●	○	○	○	●	◐	●	○	●	●	N/A	Req/seg
Srirama 2014 [28]	●	○	●	●	○	○	●	○	●	●	one week	Req/seg
Nan 2012 [29]	●	○	●	○	●	○	●	○	●	○	—	Req/seg
Kavitha 2015 [30]	○	○	○	○	○	○	●	○	●	○	—	Subtasks
LLOOVIA	●	○	●	●	●	●	●	●	●	●	one year	Req/seg

Table 1: Comparison of LLOOVIA with the most related papers. The black circle means that the characteristic is reached, a white circle that it is not reached and a half-filled circle that it is partially supported.

represents a more common problem of cloud resource optimization. In this case, the workload is represented as a request arrival rate which must be served using the allocated VMs ([24], [25], [26], [27], [28], [29] and [30]).

Cost optimization of VM allocation has often been included as part of a brokerage strategy. The cloud broker is the intermediary between the cloud user and the cloud provider, and depending on the workload requirements of the users, it applies the optimization algorithm in order to find the most cost-effective VM allocation in the provider. In [24] the authors present an approach for a brokerage service that optimizes the placement of VMs in a multi-cloud scenario. The authors consider a static situation where user and provider conditions do not change over time, so they obtain the number of required on-line VMs. Their model supports different VM types, but it is limited to on-demand VMs.

A similar approach is considered in [25]. In this paper, the authors modified the broker architecture in order to work with different scheduling strategies for optimal deployment, while considering different optimization criteria and constraints. The developed model makes a dynamic decision every sample period in order to manage changing conditions, such as user requirements and cloud provider prices. In this work, a special price schema offered by Amazon and based on a bidding process, called spot price, is considered.

In [26] the authors propose a cloud brokerage service that aggregates the cloud user demands to take advantage of cheaper prices of reserved VMs. The service serves 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.

In [27] 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 price schemes. However, it only uses one VM type in the analysis. As the linear programming model is NP-complete a greedy algorithm is used as heuristic to find the optimal solution. The algorithm proceeds by merging sites on candidate cloud providers that meet the Quality of Service (QoS) requirements. In this way, the algorithm aggregates resource requirements and maximizes the total number of reserved VMs.

In [28] the authors present a resource provisioning policy that can find an optimal cost setup. They consider different types of VMs and their performance properties. In their model, the authors take into account the lifetime of each running VM, which makes it possible to make decisions about changing the state of the VM according to the workload conditions, switching on a new VM or turning it off. The authors also introduce the problem of VM limits, that is, the maximum number of VMs that can be hired from a cloud provider. The model only applies those limits

to on-demand analysis, since it does not consider reserved VM types.

In [29] 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 (RTT) requirements. They propose two optimal schemes for VM allocation for both single-site and multi-site clouds. The optimal schemes are NP-hard, so for each case they provide a greedy algorithm that finds a suboptimal solution within a short time. In this work, both reserved and on-demand prices 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.

Finally, in [30] the authors propose a hybrid optimization algorithm for dynamic resource allocation in cloud computing. The main characteristic of this work is that it divides the task in several subtasks which are assigned to different independent resources. For these resources, it considers two costs: economic cost, related to cloud maintenance, and computational cost. The optimal cost is obtained by combining two optimization algorithms. In this paper, the only resources considered are on-demand VMs; reserved VMs are disregarded.

There are other less related works, not shown in Table 1, where cost is not the main objective to minimize, but it is also important. In [32] the authors analyze how improvements in workload prediction influence cost. In [33], the authors investigate how the allocation of distributed tasks to VMs influences the execution time and the cost it implies. Finally, auto-scaling algorithms are another example where the main target is to manage the workload variations efficiently, where cost is also considered. These algorithms determine when new VMs should be leased depending on the workload evolution ([34] and [35]). However, the auto-scaling problem is based on increasing or decreasing the number of the same VM type, but it may be more cost-effective to change the type of the VM used.

As can be seen in Table 1, LLOOVIA approaches the cost optimization problem in a more complete way than previous works: it can be used in multi-cloud environments; it takes into account how cloud providers support different VM types and resource limits for reserved and on-demand price schemes; and it is the only work of those that express the workload as arrival rates that provides both the optimal number of reserved and on-demand VMs. Finally, LLOOVIA is the only strategy that has been evaluated with a real workload for a whole reservation period of one year.

3. Model and resolution

3.1. Overview

The problem to solve is to find the optimal allocation of VMs for each time slot of the next reservation period (usually, the reservation period is one year and the time slot is one hour). A feasible allocation is a set of numbers stating the number of VMs of each type, in each cloud provider and each region, which does not exceed the limits imposed by the cloud provider, and that is capable of giving enough performance to serve the load for each time slot. An optimal allocation is a feasible allocation with minimum cost.

To solve this problem two phases are required. In Phase I, the optimal allocation of reserved VMs is found (the optimal allocation of on-demand VMs is also found, but it is discarded). This phase requires a prediction of the load for each time slot in the next reservation period. We call this the "Long-Term Workload Prediction", or LTWP. For this phase, it is assumed that the types of VMs, and their characteristics and prices do not vary during the reservation period. This phase requires solving a problem with a huge number of variables, so some strategies to reduce the problem size are presented, such as histogramming and binning.

In Phase II the number of reserved VMs is fixed (by the result of Phase I) and the optimal number of on-demand VMs is found. This phase requires solving a much smaller problem and no special strategy to reduce its size is required. In addition, Phase II does not require a long-term prediction, but only a prediction for the next time slot, which will be much more accurate. We call this the "Short-Term Workload Prediction", or STWP. Since Phase II is carried out at each time slot, independently from previous time slots (except for the reserved VMs which are fixed), there is no need to assume that the types, characteristics or prices of the on-demand VMs are fixed; they can be different for each time slot.

Note that Phase I introduces several sources of error (such as inaccuracy of the LTWP, simplifications and approximations), which can cause the solution to be suboptimal. Hence, the result of both phases as a whole will produce a solution which is also globally sub-optimal, although Phase II will produce an optimal solution for each time slot, for the number of reserved instances given by Phase I.

Table 2 summarizes the key differences between the two phases of the resolution. In the next section we formally define the kind of information required by each phase. The concepts and notation are the same for both phases, although the particular values of the parameters, as well as the formulation of each optimization problem can be different at each phase.

3.2. Architecture and pricing models

Cloud providers generally offer different types of virtual machines in different regions. Some providers (eg., Amazon) have two pricing schemes for most of their VM types: on-demand VMs, with a pay-per-use schema whose cost is proportional to the time the VM is allocated, and reserved VMs with a much lower price, but one that must

		Phase I	Phase II
	When	Offline (before deploying)	Online (real-time)
Inputs	VM properties	Fixed	Could vary
	VM prices	Fixed	Could vary
	Reserved alloc.	Unknown	Given (by Phase I)
	On-demand alloc.	Unknown	Unknown
	Load prediction	Whole year (LTWP)	Next hour (STWP)
	Output	Allocation for reserved VMs	Allocation for on-demand VMs
	Challenges	LTWP accuracy NP-hard Huge problem size	STWP accuracy NP-hard
	Simplifications	Histogram Binning	None

Table 2: Summary of characteristics of Phases I and II

be paid even if the machines are not allocated. The payment of the reserved VMs can be divided into an upfront amount to be paid in advance, plus a periodic payment (which does not depend on the usage of those VMs, as said).

In addition, most cloud providers impose limits on the total number of VMs or cores in use in each region or availability zone. For example, Microsoft Azure imposes a limit on the total number of cores per region, Amazon EC2 imposes a limit on the total number of on-demand VMs per region, and a different limit on the total number of reserved VMs per availability zone. We unify these kinds of constraints with the concept of "Limiting Set", to be defined later. Furthermore, they also limit the maximum number of VMs for each VM type. All these limits have default values which can be increased (but not removed) by negotiating with the cloud provider.

The service to be run on that infrastructure is considered to be perfectly parallelizable, with infinite horizontal scalability. It is thus composed of an arbitrary number of identical tasks running on a mix of VM types, some reserved and some on-demand, across different regions and zones.

The performance of each VM type depends on aspects such as memory, number of cores, kind of processor, etc. These different aspects are summarized in a single number (e.g., transactions per second), which is the performance of the service when running in that kind of VM. These numbers can be determined by benchmarking or monitoring, and are considered to be known in advance for our model. Usually the performance of each VM type does not depend on the region or zone in which it exists, but our model does not require this assumption.

Next, we provide some formal definitions and notation about these architectural concepts. Table 3 summarizes this notation and the one used in Subsections 3.3 and 3.4.

- LS_j denotes a **Limiting Set**, with $j = 1, \ldots, N^{\mathrm{LS}}$. Each VM type is deployed within a limiting set which imposes a limit on the maximum number of VMs and/or cores which can be running simultaneously inside that set. This concept translates to different terms in different cloud providers. For example, for Amazon on-demand VMs, LS_j are regions, while for Amazon reserved VMs, LS_j are availability zones (which are inside a region) [36]. In both cases, the limit is on the total number of VMs deployed in the corresponding set. For Microsoft Azure, LS_j are regions, and the limit is on the number of cores.

  Each LS_j defines thus 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. Note that both kinds of limits could be used simultaneously, but usually cloud providers enforce only one of them. In this case we can assume the other to be infinite.

- IC_i denotes **Instance Class** i. This represents a type of VM on a particular cloud provider and particular limiting set, under a particular pricing schema (reserved or on-demand). An on-demand c4.large on Amazon EC2, on region us-east-2, or a reserved c4.large on Amazon EC2, on availability zone us-east-1b are examples of different Instance Classes. Each Instance Class defines the following attributes:

  - 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_i is the performance of that class under the considered kind of load, and expressed in the same units as the load. For example, if the load is measured in "requests per second", the performance is the requests per second which this class can serve. As said before, this value 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 the index of the limiting set LS_j to which this instance class belongs.
  - type_i is the VM type of this class, for example c4.large.
  - $\max_i$ 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.

Symbol	Meaning
LS_j	Limiting Set ($j = 1, \ldots, N^{\mathrm{LS}}$)
$\mathrm{LS}_j^{\mathrm{vms}}$	Limit on the number of running VMs
$\mathrm{LS}_j^{\mathrm{cores}}$	Limit on the number of cores in use
IC_i	Instance Class ($i = 1, \ldots, N^{\mathrm{IC}}$)
p_i	Price of IC_i
perf_i	Performance of IC_i
rsv_i	Boolean. True if IC_i is reserved, false otherwise
c_i	Cores provided by IC_i
ls_i	Limiting set to which IC_i belongs
type_i	VM type of IC_i
max_i	Limit on the number of type_i VMs in ls_i
$\bigcirc^{\mathrm{dem}}$	Superindex denoting on-demand ICs
$\bigcirc^{\mathrm{res}}$	Superindex denoting reserved ICs
T	Reservation period (typically one year)
t	Time slot length (typically one hour)
t_i	i-th time slot
l_i	Workload for timeslot t_i
h_l	Number of time slots in which load level l appears
L	Effective load levels: set of l for which $h_l > 0$
Y_i	Number of reserved VMs of class IC_i in the solution, for any load level
X_{il}	Number of on-demand VMs of class IC_i in the solution, for the load level l
C	Cost of the solution

Table 3: Summary of the notation used in the model and optimization problem

To improve readability, the set of all possible instance classes can be split into two disjointed subsets: the reserved instances (those with attribute rsv_i true) and the on-demand instances (those with attribute rsv_i false), renumbering the indexes in each subset. We introduce new notation to refer to these subsets.

- Let N^{res} be the total number of reserved instance classes, and $\mathrm{IC}^{\mathrm{res}} = \{\mathrm{IC}_i^{\mathrm{res}}\}, \quad i = 1, \ldots, N^{\mathrm{res}}$ the set of all reserved instance classes. All the attributes of these instance classes have $^{\mathrm{res}}$ as a superindex. For example p_i^{res} is the price of the i-th reserved instance class, $\mathrm{perf}_i^{\mathrm{res}}$ is its performance, $\mathrm{ls}_i^{\mathrm{res}}$ is its limiting set, and so on.
- Let N^{dem} be the total number of on-demand instance classes and $\mathrm{IC}^{\mathrm{dem}} = \{\mathrm{IC}_i^{\mathrm{dem}}\}, \quad i = 1, \ldots, N^{\mathrm{dem}}$ the set of all on-demand instance classes. All the attributes of these instance classes have $^{\mathrm{dem}}$ superindexes.

3.3. Load model

The model does not make any assumption about the kind of workload, as long as it is expressed in the same units as the performance of the instance classes. For example, if the performance is measured with a benchmark like OLDIsim, part of PKB [37], which can give a "requests per second" metric, the load will also be expressed in requests per second.

We assume time divided into *slots* of length t (e.g., 1 hour), and denote each of these time slots by t_i.

For Phase I, we consider a reservation time T (e.g., one year), and a prediction of the load is assumed to be known in advance for all time slots in T (e.g., for all hours in a year). As stated before, we call this the *Long-Term Workload Prediction* (LTWP). This paper does not describe nor assume any way to predict this workload; any of the methods mentioned in the related work section can be used.

The LTWP is a set of numbers, LTWP $= \{l_i\}$, each one representing the predicted load for time slot t_i. The size of this set is equal to the number of time slots in T, i.e: T/t, for example 8760 when $t = 1$ hour and $T = 1$ year.

In order to reduce the problem size, we propose representing the LTWP as a histogram, which is a novel approach for this kind of problem. We denote this histogram by $H = \{h_l\}$, h_l being the number of different slots in which the load l appears in the LTWP. We keep only the load levels which appear at least once, and denote that set by L, the set of effective load levels. Formally, $L = \{l : h_l > 0\}$.

For Phase II, once the optimal number of reserved instances has been determined, we do not need the predicted load of all future slots, but only the expected load for the next slot. As stated before, this prediction will be called the *Short-Term Workload Prediction* (STWP), and can be computed by different means, more accurate, than the LTWP.

For comparison against the theoretical optimum, we will refer sometimes to the concept of a "Perfect Prediction". This is a LTWP given by an "oracle" which has perfect knowledge of the future, which means that the STWP is the same as the LTWP for each time slot. If such a perfect prediction were available, Phase II would not be neccessary, because, as we will see in next section, Phase I also produces the optimal number of on-demand VMs as output. However, since the perfect prediction is only a theoretical construct, Phase II is used to refine the results of Phase I using a more precise STWP.

3.4. Optimization problem for Phase I

The optimization problem can be formulated as an integer linear programming problem, with the following unknown variables:

- Y_i is an integer representing the number of reserved VMs of class $\mathrm{IC}_i^{\mathrm{res}}$ to be purchased at the beginning of the reservation period T. Since they are paid even when deallocated, we will consider these VMs always active, even when the load is low and they are not

required. Therefore, they are active for any possible load level l.

- X_{il} is an integer representing the number of on-demand VMs of class IC_i^{dem} to be purchased at any time slot for which the predicted load is l. Thus, this number can be different for each time slot, as the predicted load varies.

Although the objective of Phase I is to find out the optimal number of reserved VMs (Y_i), it is also necessary to introduce X_{il} because the optimal number of reserved VMs also depends on the characteristics (price and performance) of on-demand VMs. So, solving Phase I will also produce the optimal number of on-demand VMs at each time slot. However, this number would only be valid if the LTWP were a perfect prediction, which it is not, and if no simplifications or approximations were introduced in Phase I, which is not usually the case.

The cost of a given allocation is composed of the cost of the reserved VMs, which is a fixed amount for each time slot, plus the cost of the on-demand VMs, which is in general different for each time slot.

The cost of the reserved VMs is simple to compute, given the price per slot of each reserved VM, since they are paid for all slots: $C^{\text{res}} = \sum_{i=1}^{N^{\text{res}}} Y_i p_i^{\text{res}} T/t$, T/t being the number of time slots in the reservation period (e.g., 8760 hours in a year).

The cost of the on-demand VMs depends on the cost of the allocation for each particular slot with predicted load l, which is $\sum_{i=1}^{N^{\text{dem}}} X_{il} p_i^{\text{dem}}$, multiplied by the number of times that the predicted load l is repeated in all time slots, i.e., the value of the histogram $H(l) = h_l$. Note that this computation assumes a fixed price for on-demand VMs, but this assumption can be removed in Phase II. Adding the costs of the on-demand VMs for every load l, the cost of the on-demand VMs (C^{dem}) is calculated:

$$C^{\text{dem}} = \sum_{i=1}^{N^{\text{dem}}} \sum_{l \in L} X_{il} p_i^{\text{dem}} h_l$$

The total cost of the allocation for the whole reservation period is:

$$\begin{aligned} C &= C^{\text{res}} + C^{\text{dem}} \\ &= \sum_{i=1}^{N^{\text{res}}} Y_i p_i^{\text{res}} T/t + \sum_{i=1}^{N^{\text{dem}}} \sum_{l \in L} X_{il} p_i^{\text{dem}} h_l \end{aligned} \quad (1)$$

Equation (1) is the function to minimize, subject to the following restrictions:

- **Performance restriction**. The performance of the allocation for each time slot should be enough to fulfill the predicted workload for that time slot. Alternatively, this can be formulated in terms of load levels, i.e., the allocation for each predicted load level should guarantee enough performance to serve that load level. This can be expressed with the following equation:

$$\sum_{i=1}^{N^{\text{res}}} \text{perf}_i^{\text{res}} Y_i + \sum_{i=1}^{N^{\text{dem}}} \text{perf}_i^{\text{dem}} X_{il} \geq l \qquad \forall l \in L \quad (2)$$

- **Maximum number of VMs** restriction. The number of machines instantiated at each time slot (alternatively, for each predicted load level) must respect the limits imposed by the cloud provider. This restriction is modelled with three different equations.

  First, for each VM type the cloud provider can impose a limit (in Amazon, this limit does not apply to reserved VMs [5], so Eq. (3) would not be used):

$$Y_i \leq \max_i^{\text{res}} \qquad \forall i = 1, \ldots, N^{\text{res}} \quad (3)$$

$$X_{il} \leq \max_i^{\text{dem}} \qquad \forall l \in L, i = 1, \ldots, N^{\text{dem}} \quad (4)$$

  Also, the total number of VMs active in a limiting set, regardless of their type, has a maximum:

$$\sum_{i \in S_j^{\text{res}}} Y_i + \sum_{i \in S_j^{\text{dem}}} X_{il} \leq \text{LS}_j^{\text{vms}} \qquad \forall l \in L, j = 1, \ldots, N^{\text{LS}} \quad (5)$$

  Finally, the total number of cores active in a limiting set can also have a limit:

$$\sum_{i \in S_j^{\text{res}}} Y_i c_i + \sum_{i \in S_j^{\text{dem}}} X_{il} c_i \leq \text{LS}_j^{\text{cores}} \qquad \forall l \in L, j = 1, \ldots, N^{\text{LS}} \quad (6)$$

  In the last two restrictions, some new notation is introduced. S_j^{res} denotes the set of subindexes of all reserved instance classes which share the same limiting set LS_j, or more formally $S_j^{\text{res}} = \{i : \text{ls}_i^{\text{res}} = j\}$. Analogously for the on-demand instances classes sharing limiting set LS_j, $S_j^{\text{dem}} = \{i : \text{ls}_i^{\text{dem}} = j\}$. These instance classes (reserved plus on-demand) must not exceed the limit of running VMs nor the limit of running cores for that limiting set LS_j.

For Phase I, the problem to solve is to minimize the cost given by Eq. (1) subject to the restrictions given in Eqs. (2), (3), (4), (5) and (6).

Note that this phase usually deals with a large number of variables. The actual number depends on the number of different VM types, cloud providers, regions and zones

and the number of different load levels in the LTWP. In particular, the total number of variables is $N^{\text{res}}+N^{\text{dem}}|L|$, $|L|$ being the size of set L, which in the worst case may be as large as the number of time slots in the reservation period. Since the problem of integer programming is already NP-hard, the large number of variables may make the time required to solve the problem unpractical. However, the size of the problem can be reduced by reducing the size of the set L. This involves reducing the number of possible load levels in the LTWP, which in practice means simply aggregating different load levels into the same class mark when computing the histogram of the predicted workload. We will refer to this technique as *binning* in the remainder of the paper. In order to guarantee the performance restriction in Eq. (2) when binning is being used, all load levels in the same bin must be represented by the right extreme of the bin, i.e., they are represented by a worst-case value.

By solving the integer programming problem, the values of variables Y_i and X_{il} are found. If no binning has been performed on the LTWP, the values of Y_i represent the optimal number of reserved VMs for each VM type and each region and zone. If some kind of binning has been applied to the histogram of the LTWP, then the values of Y_i will be an approximation of this optimal number. The accuracy of the approximation depends on the size and number of bins. Their influence will be explored in the experimental section.

The values of X_{il} of the solution represent the optimal number (or an approximation if binning has been used) of on-demand VMs to purchase for each VM type and each predicted load level, but these are usually discarded because the actual load levels to be observed in Phase II will not be exactly those present in the LTWP. This is especially true when binning is used.

3.5. Optimization problem for Phase II

For Phase II the same set of equations is used for each time slot, but the inputs for this phase are different and the problem is much simpler. For Phase II the set L is composed of a single load-level, which is the STWP (the workload expected for the next slot), so no histogram is used and hence binning is not required. Additionally, the values for Y_i are known in this second phase, so these variables can be removed from the problem and the performance restriction accordingly adjusted.

Let the performance given by all reserved VMs Y_i be denoted by Perf^{res}. After Phase I, this is a known quantity, given by $\text{Perf}^{\text{res}} = \sum_{i=1}^{N^{\text{res}}} Y_i \text{perf}_i^{\text{res}}$. Hence, if the predicted load for the next time slot is l, the problem to solve for that time slot is to minimize the cost:

$$C = \sum_{i=1}^{N^{\text{dem}}} X_{il} p_i^{\text{dem}} \tag{7}$$

subject to restrictions:

$$\sum_{i=1}^{N^{\text{dem}}} \text{perf}_i^{\text{dem}} X_{il} \geq l - \text{Perf}^{\text{res}} \qquad \forall l \in L \tag{8}$$

$$X_{il} \leq \max_i^{\text{dem}} \qquad i = 1, \ldots, N^{\text{dem}} \tag{9}$$

$$\sum_{i \in S_j^{\text{res}}} Y_i + \sum_{i \in S_j^{\text{dem}}} X_{il} \leq \text{LS}_j^{\text{vms}} \qquad \forall j = 1, \ldots, N^{\text{LS}} \tag{10}$$

$$\sum_{i \in S_j^{\text{res}}} Y_i c_i + \sum_{i \in S_j^{\text{dem}}} X_{il} c_i \leq \text{LS}_j^{\text{cores}} \qquad j = 1, \ldots, N^{\text{LS}} \tag{11}$$

Note that in the last two restrictions, Y_i is not part of the problem to solve, but known constants, found in Phase I. Also note that, if $(l - \text{Perf}^{\text{res}}) < 0$, the solution is trivially $X_{il} = 0 \; \forall i$, so this particular case does not need to be solved. This represents the case in which the predicted load is already fulfilled by the reserved VMs, and thus no on-demand VMs are required. Note also that p_i^{dem} is the price of the on-demand instances in the next time slot, and it is not required to be the same for all time slots.

The above integer programming problem has to be solved for each time slot in which $(l - \text{Perf}^{\text{res}}) > 0$. All these problems are independent of each other and the number of variables in each one is only N^{dem}, which means that the size of the problem is small enough to be solved exactly by branch-and-cut algorithms, typically implemented in integer programming tools, such as CBC [38], without needing to resort to heuristics or approximations.

The final solution for the allocation problem uses the reserved VMs from Phase I and the on-demand VMs from Phase II. The total performance and cost of each time slot is the sum of the performances and costs of both reserved and on-demand VMs.

4. Methodology and example

This section provides a simple example. More details and explanations about it, along with the code to solve it and generate the plots presented in the paper, can be found in the Jupyter Notebook called `Example.ipynb` provided in the additional material accompanying this paper.

4.1. Required inputs

The input data for both Phases in this example consists of a subset of regions and availability zones (Limiting Sets) from one imaginary cloud provider (see Table 4), and a subset of two VM types ("small" and "large") with their characteristics and prices for all the limiting sets (see Table 5).

https://github.com/asi-uniovi/lloovia

j	LS_j	LS_j^{vms}	LS_j^{cores}
0	Region1	15	∞
1	Region2	15	∞
2	Region2 Z1	12	∞
3	Region2 Z2	12	∞

Table 4: Limiting sets for the example

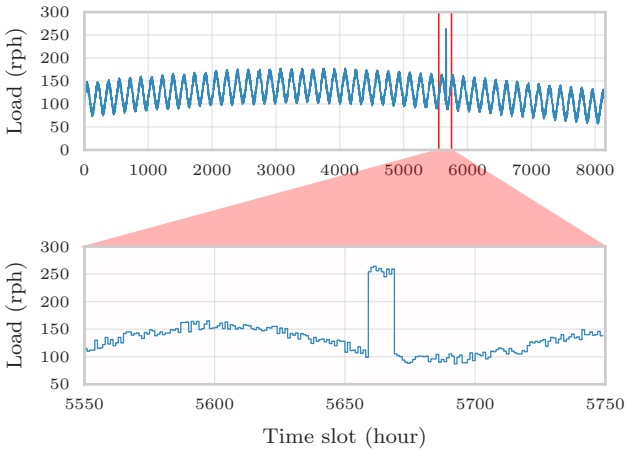

Figure 2: Synthetic LTWP, in requests per hour (rph), for the whole reservation period, with a zoom on a spike in the load

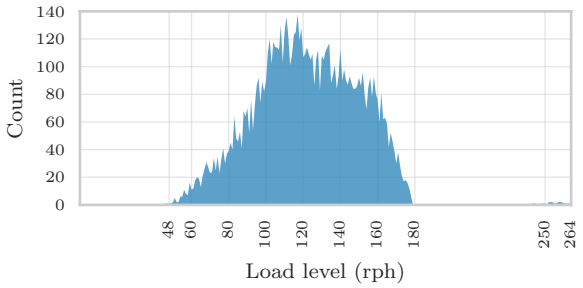

Figure 3: Histogram of the example workload

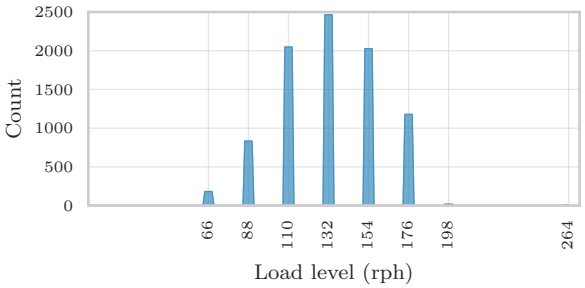

Figure 4: Histogram with 12 bins of the example workload

4.2. Solving Phase I

For Phase I, a prediction of the workload for each time slot in a whole reservation period (LTWP) is required. Let us use the one depicted in Fig. 2, which was synthetically generated as a uniform random variable between 90 and 110, plus a sinusoid of amplitude 30 and period of one week, to represent the variations among weekdays, plus another sinusoid of amplitude 30 and period equal to 573 days, to represent slow variations between years. To make the example more interesting, a spike was added around time slot 5660 in which the load is 150 units larger than usual. The lower part of the same figure zooms on that spike showing 200 time slots. This spike makes it more difficult to intuitively see the "base load" to be covered by reserved VMs.

The histogram of this LTWP example is depicted in Fig. 3. Even without binning, the effective number of load values is 139 in this example, which shows one of the advantages of switching from the time-slot domain to the load-level domain. The size of the input data is reduced from 8760 to 139 in this example. A second advantage is that we have control over the size of the input, which can be further reduced by using binning when computing the histogram. Fig. 4 shows the histogram with 12 bins for this example. The ticks on the x-axis correspond to the values for the eight effective load levels which appear in this case.

In this example, with 4 reserved instance classes and 4 on-demand instance classes, the integer problem to solve will have 4 variables Y_i, plus $4 \times |H|$ integer variables X_{il} (being $|H|$ the number of non-empty bins in the histogram). In our case, if no binning is performed, $|H| = 139$, so the total number of variables in the LP problem is 560. If we use binning with 12 bins, then $|H| = 8$ as already shown, and the total number of variables is reduced to 36.

The number of constraints in the problem is also related to the size of $|H|$, since equations (2), (3), (4), (5) and (6) have to be satisfied for each load level. In this example, without binning, the total number of constraints is 1251. Using 12 bins the number of constraints in the problem is reduced to only 72.

Using Python and PuLP [39], and our Python module `lloovia`, provided as part of the supplementary material, the LP problem is generated, written in disk in `lp` or `mps` formats, and solved by CBC [38] in 0.280 seconds when no binning is used and in 0.032 seconds using 12 bins. The machine in which the analysis was run was an Azure's Standard D2 (2 cores, 7 GB memory), with CPU reported as Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz. In this small example, the solution for the problem without binning is obtained in a very short time, so it is not worthwhile to reduce it with binning. However, when the complexity of the problems grows, the problem without bins might not be solved in a reasonable time, as shown in the next section, while with bins the solution can be reached in seconds.

The solution from Phase I is the optimal number of reserved and on-demand VMs that should be allocated at each possible predicted load level to minimize the cost while satisfying performance constraints and the limits enforced by the cloud provider. This solution is stored in a table, whose index is all the possible load levels (in the

i	IC_i identifier	$type_i$	max_i	ls_i	p_i ($/h)	$perf_i$	c_i	rsv_i
0	small (Region1) [dem]	small	10	0	0.050	5	1	False
1	large (Region1) [dem]	large	10	0	0.110	10	2	False
2	small (Region2) [dem]	small	10	1	0.052	5	1	False
3	large (Region2) [dem]	large	10	1	0.120	10	2	False
4	small (Region2 Z1) [res]	small	∞	2	0.038	5	1	True
5	large (Region2 Z1) [res]	large	∞	2	0.090	10	2	True
6	small (Region2 Z2) [res]	small	∞	3	0.038	5	1	True
7	large (Region2 Z2) [res]	large	∞	3	0.090	10	2	True

Table 5: Instance Classes and their properties for the example

LTWP), and the rows store the optimal number of VMs for each IC_i to allocate for that load level. In addition, the value of the objective function is the optimal cost, assuming that LTWP is a perfect prediction. In this example this cost is $9015.216.

This information can also be represented as a plot, shown in Fig. 5. In this plot, the x-axis shows all possible load levels in the LTWP, and for each one of them the optimal number of VMs is represented as a stacked bar, using different colours for each IC (the legend for this plot is shown separately in Fig. 7). The gaps in Fig. 5 correspond to load levels which do not appear in the LTWP.

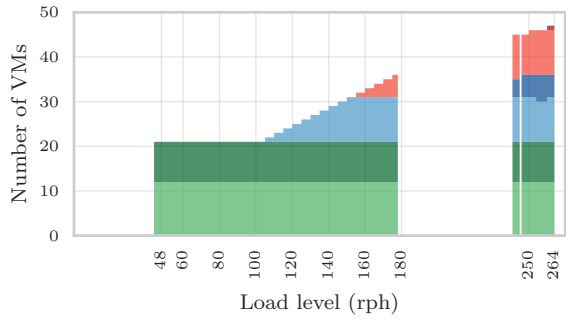

Figure 5: Optimal solution from Phase I, no binning used

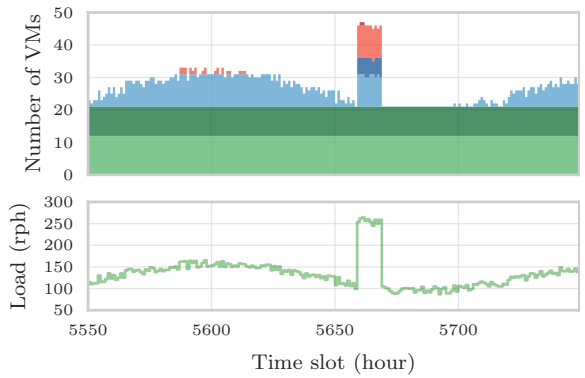

Figure 6: Optimal allocations from Phase I for each time slot

If LTWP were a perfect prediction, then the solution of Phase I would give the optimal allocation for each time slot, shown in Fig. 6, and no Phase II would be required. This figure was built using for each time slot the optimal allocation given by the solution for the load level present in that time slot.

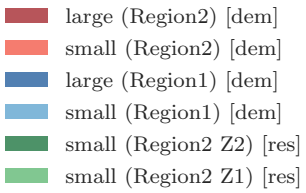

Figure 7: Legend for figures 5 and 6.

4.3. Solving Phase II

The general procedure for Phase II is as follows:

- For each time slot, the number of reserved VMs determined in Phase I is allocated.
- If the load l expected for the next time slot is below the performance given by the reserved VMs, no action is required.
- Otherwise the optimization problem described by the equations in section 3.5 has to be generated and solved.

If binning was used in Phase I the number of reserved VMs obtained is no longer optimal, so the cost of Phase II solution will be greater than the optimal obtained without binning. However, it is our hypothesis that the number of reserved VMs will be very close to the optimal one. In this example, indeed, the difference is only one extra reserved VM of type small in Zone 2, and the cost of Phase II for the whole reservation period is $9031.700. This is very close to the cost of the optimal solution, which was already shown to be $9015.216.

5. Experimental results

5.1. Introduction

An extensive set of experiments has been carried out with the following objectives: 1) to analyze the influence of binning on the time required to solve the linear problem and on the quality of the solution, 2) to study the influence

of the prediction used in Phase I on the final cost of the solution, 3) to analyze how LLOOVIA behaves with the different kinds of workloads typically found in cloud computing systems, 4) to demonstrate that LLOOVIA can be applied to real large-sized cases and 5) to demonstrate that it can work in multi-cloud environments.

In order to test the technique, the number of accesses per hour in one year is required. There are few data sets with this length publicly available so, in order to test LLOOVIA under many different scenarios, part of the experiments has been carried out using synthetic workloads (Section 5.2) and the other part with access data to the Wikipedia (Section 5.3).

All the case studies have been implemented in Python, using PuLP as linear-programming modeler [39] and COIN CBC [38] as solver. They were run in a Standard D2 machine deployed in Azure with the characteristics described in the previous section. All the code and data to reproduce the experiments is supplied in the additional material of this paper.

5.2. Analysis with synthetic workloads

5.2.1. Input data

In [13], a characterization of the workloads typically found in cloud computing is presented. Five patterns are identified: static, periodic, once-in-a-lifetime, unpredictable and continuously changing. In order to test LLOOVIA under all these patterns, a set of synthetic workloads has been generated. The generated traces are grouped in seven types (Fig. 8). The "continuously changing" pattern from [13] has been subdivided in two: one "increasing" following an exponential curve and another "decreasing" following a linear relation. In addition, a type that includes a sum of the variations has been added. Gaussian noise is added to all cases.

The savings in cost that can be accomplished by using reserved instances depend on the average level of the workload and the limits on the number of allowed instance classes. Thus, to test LLOOVIA under different conditions, all of the workload types have been generated using four base levels: $5 \cdot 10^4$, 10^5, 10^6 and $3 \cdot 10^6$ requests per hour. This results in 28 different scenarios.

While the workloads are synthetically generated, the VMs required to support these workloads are selected from a real provider: Amazon EC2. To determine the performance of the selected machines, WikiBench [40], a benchmark that emulates the Wikipedia, has been used. Flux 7 has published [41, 42] an analysis of the performance of several Amazon EC2 VM types under this benchmark. From this data, the approximate number of requests that each VM type can handle at the inflection point (where the average latency starts to increase) was obtained ("Perf." column in Table 6).

For the limits and the prices, two Amazon regions (US West Oregon and EU Ireland) were used. These two regions have a limit of 20 VM per instance class for on-demand instances and 20 reserved VMs in each availability

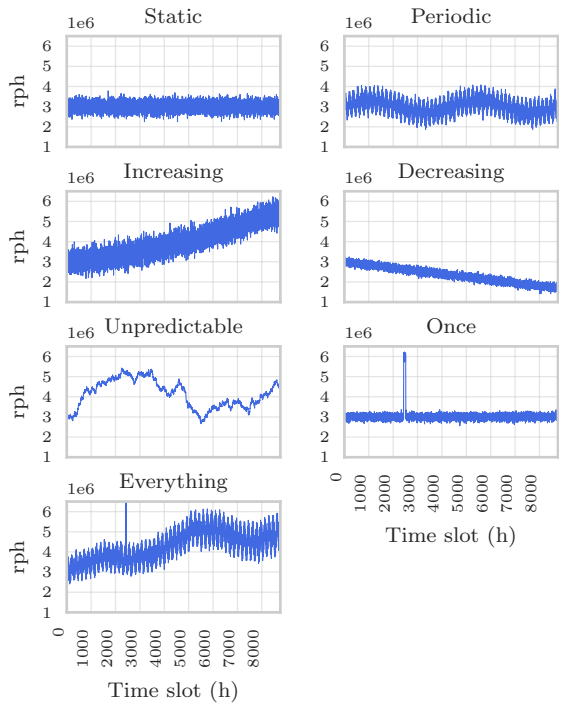

Figure 8: Types of synthetic workloads used in the experimentation. The figure only shows the case for base level = $3 \cdot 10^6$

Type	Perf. (rph)	Region	Price ($/h) On-demand	Reserved
c3.large	18000	US	0.10500	0.06187
		EU	0.12000	0.07660
c3.xlarge	27000	US	0.21000	0.12477
		EU	0.23900	0.15422
c3.2xlarge	46000	US	0.42000	0.24772
		EU	0.47800	0.30753
m3.medium	9000	US	0.06700	0.04030
		EU	0.07300	0.04954
m3.large	11000	US	0.13300	0.08139
		EU	0.14600	0.10011
m3.xlarge	39000	US	0.26600	0.16301
		EU	0.29300	0.19932

Table 6: VM data for the experiments with synthetic load. The US region is West Oregon and the EU region is Ireland

zone. Each region has three availability zones. In total, there are 12 on-demand and 36 reserved instance classes. The prices and characteristics for each VM type in each region can be obtained using an API provided by Amazon. For reserved VMs, a period of one year with all upfront payment was considered (Table 6).

5.2.2. Analysis of the influence of binning

The first set of experiments analyzes the influence of binning on the time required to create and solve the problem and on the quality of the solution, under different workloads.

As there are 36 different reserved instance classes (two regions with three availability zones each and six VM types in each availability zone), 12 on-demand instance classes (two regions with six VM types each) and 8760 hours in a year, the linear problem to solve is huge, with $36 + 12 \times 8760 = 105\,156$ variables. This makes the time required to create (write a file with all the equations for the solver) and solve the problem very long. Using bins, the 8760 factor can be reduced, but the solution may be not optimal.

In order to analyze the relation between time and solution quality, each of the 28 combinations of workload types and levels was used as LTWP and Phase I of LLOOVIA was carried out using different number of bins: 5, 10, 20, 40, 80, 200, 400, 800, 1500, 3000 and 6000. In addition, an experiment without using bins was carried out. The solution to this experiment is the optimal solution so, in order to assess the quality of the solutions with bins, their cost is compared to the cost obtained without bins.

In order to remove the influence of the prediction in this analysis, Phase II was applied using as STWP the same workload used as LTWP (i.e., the LTWP was a perfect prediction).

Fig. 9 shows the time to create and solve Phase I of the problem with different numbers of bins. Notice that both axes are in log scale. The figure shows that creation time increases exponentially with the number of bins. The solving time also increases at least exponentially in most of the scenarios, although some of them (e.g., Everything with level 1 000 000) present anomalies.

A time limit was set in the solver. If the limit is reached and the solution was not found, Phase I is aborted. The limit was set to 15 minutes for the experiments with bins and to 1 hour for the experiments without bins. As can be seen in Fig. 9, some of the experiments were aborted, even for the case without bins.

Fig. 10 plots the increase in cost generated in Phase II due to binning compared to the optimal solution obtained with no bins. When the optimal solution could not be found (because of the 1 hour limit), the lines are plotted in a different style and the solutions are compared to the best lower bound known, as provided by the solver; this means that the plot actually shows an upper bound of the cost increase. When Phase I with bins was aborted (e.g., 1600 bins in the unpredictable workload) because of the 15 minute limitation, no point is plotted because Phase II cannot be carried out for that number of bins.

This figure shows that in all the scenarios the increase in cost is smaller than 2% if more than 20 bins are used. This means that by using bins, a solution very close to optimal can be found in a much shorter time (under 1 second for 20 bins in all of the cases, as seen in Fig. 9). Furthermore, even in cases where the optimal solution without bins cannot be found in less than one hour, a solution very close to optimal can be found with bins in less than one second.

These results show that, leaving out the influence of the prediction, using bins can have great advantages.

5.2.3. Analysis of the influence of the error in the prediction

In the previous analysis, the same workload was used as LTWP and STWP to remove the influence of the prediction, as if the LTWP was a perfect prediction of the STWP. In this section, new workloads are generated for each scenario, using the same parameters as the original workload. These new workloads are used in Phase II as the STWP. This way, the case when the prediction is not perfect is simulated. It is expected that Phase II compensates for the errors in the prediction used in LTWP by using a more precise prediction.

In order to evaluate the impact of the error on the prediction, the optimal solution when the prediction is perfect must be obtained. This has been done with an experiment for each scenario that uses no bins and the same LTWP and STWP. The solution of this experiment represents the optimal solution that a perfect predictor or "oracle" (which cannot exist in a real environment) would find. Therefore, it is a limit for the best solution that can exist. As in the previous analysis, sometimes the optimal solution for the perfect predictor could not be found, and a lower bound had to be used instead.

Table. 7 shows the percentage increase in cost using LLOOVIA with 20 bins compared to the optimal solution or the aforementioned lower bound. The increase in cost is generally very small, under 1%. The pattern "Once-in-a-lifetime", which has a randomly generated peak, is a little higher, but under 2%. The "Unpredictable" scenario has a very large error because, as its name implies, it can not be predicted, so the STWP is totally different to the LTWP. The "Everything" scenario with an a base level of $3 \cdot 10^6$ could not be computed because with the new STWP the problem became unfeasible in Phase II, i.e., the capacity of the available on-demand instances is not enough to support the workload in some time slots, due to the limits imposed by the provider.

For the sake of brevity, the results for other bin sizes, which are similar, are not presented here. They can be found in the extra material.

This analysis shows that LLOOVIA obtains very good solutions except when the workload can not be predicted.

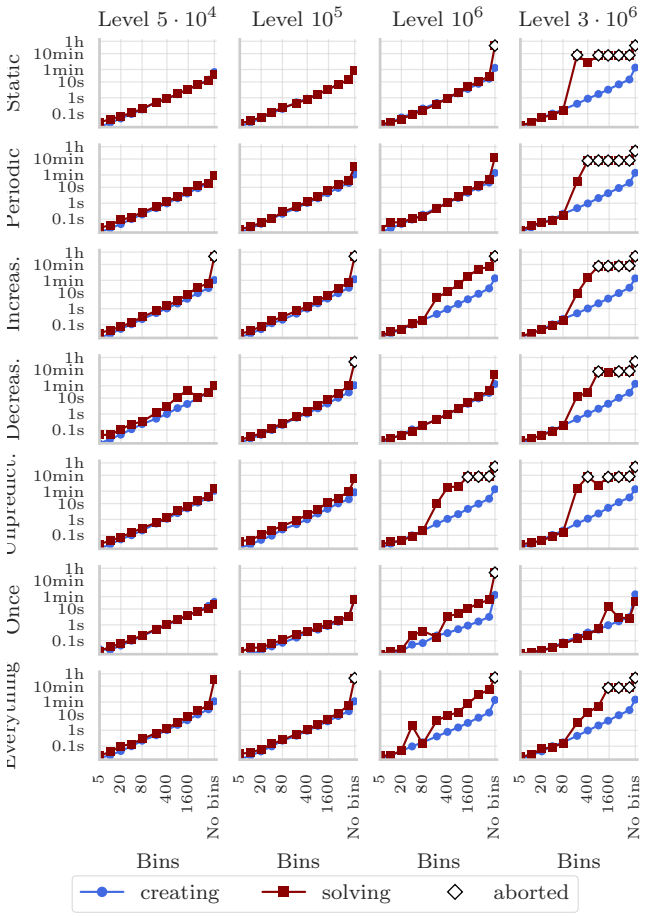

Figure 9: Solving time, depending on the number of bins, for the different scenarios. The solver is aborted if it has not found a solution in a fixed time limit (15m for binning cases, 1h for non-binning cases)

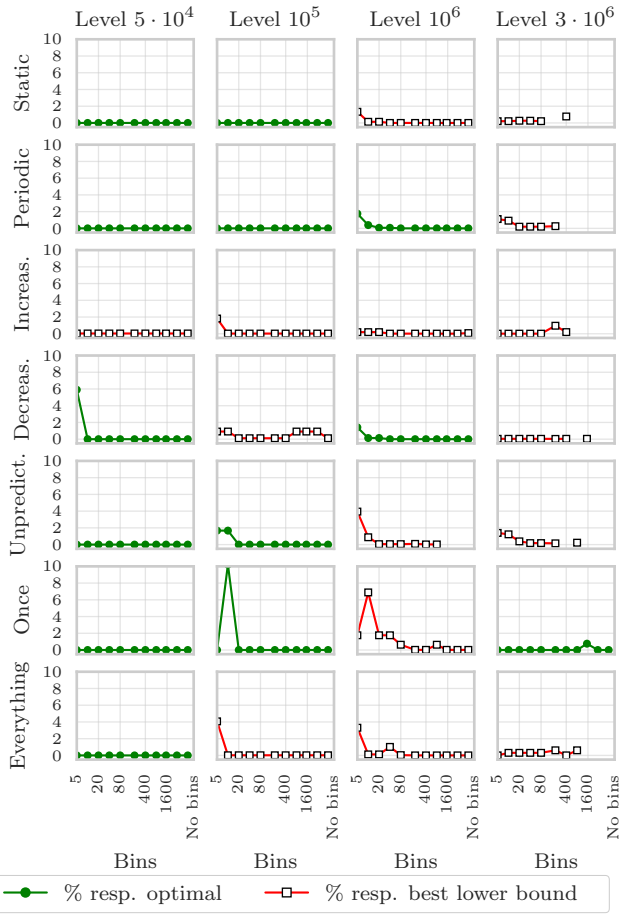

Figure 10: % overcost respect optimal (or lower bound) cost, depending on the number of bins for the different load scenarios

Base level → Case	$5 \cdot 10^4$	10^5	10^6	$3 \cdot 10^6$
Static	0 %	0 %	0.14%	0.24%
Periodic	0 %	0 %	0.04%	0.20%
Increasing	0.04%	0.03%	0.21%	1.07%
Decreasing	0 %	0.10%	0.13%	0.04%
Unpredictable	11.67%	44.80%	63.94%	114.21%
Once	0 %	0 %	1.73%	0 %
Everything	0 %	0.13%	0.40%	NA

Table 7: % increase in cost with respect to the optimal solution obtained by a "perfect predictor"

5.3. Real workloads

5.3.1. Input data

To demonstrate that LLOOVIA works with real workloads, this section presents experiments carried out using the number of requests per hour to the English Wikipedia [43] as workload. This workload has been chosen because it is a publicly available source of accesses that encompasses several years; thus, the impact of real daily, weekly, monthly and yearly variations can be taken into account.

5.3.2. Analysis of the influence of binning

In this section, an analysis similar to the one in Section 5.2.2 is carried out, but this time using the request per hour to Wikipedia in 2014 (Fig. 11), which was the year Flux 7 obtained the performance data with Wikibench. In addition, more regions and availability zones than in previous experiments are used. When the experimentation was carried out, Amazon had 10 different regions available for international commercial customers. The number of availability zones in each region varies between 2 and 5 (see Table 8). This results in a total of 39 limiting sets.

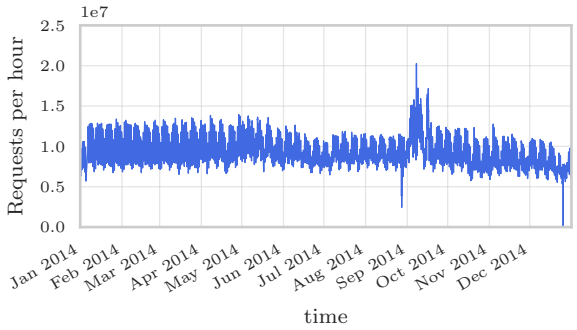

Figure 11: Requests to the English Wikipedia per hour in 2014

Region	Availability zones
US East (N. Virginia)	5
US West (N. California)	3
US West (Oregon)	3
EU (Ireland)	3
EU (Frankfurt)	2
Asia Pacific (Tokyo)	3
Asia Pacific (Seoul)	2
Asia Pacific (Singapore)	2
Asia Pacific (Sydney)	3
South America (Sao Paulo)	3

Table 8: Regions and availability zones in Amazon EC2

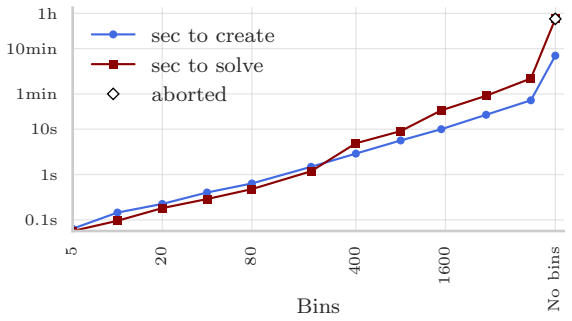

Figure 12: Time to create and solve a problem using the Wikipedia workload in 2014 with different numbers of bins

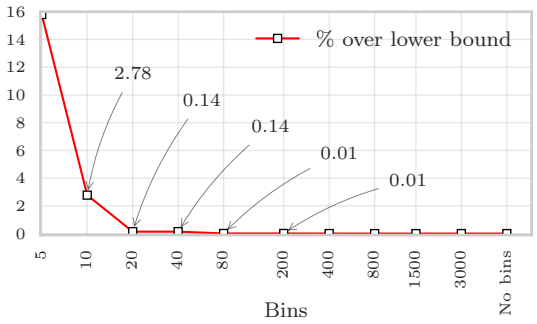

Figure 13: Cost increase as the number of bins used changes in the solution using the Wikipedia workload in 2014 with different numbers of bins

Not all VM types of Table 6 are available in all regions. Combining the on-demand and reserved VM types available in each region, a total of 222 different instance classes can be selected; 162 of them are reserved instances and 60 are on-demand instances. The number of variables for Phase I in the problem, without bins, is $162 + 60 \times 8760 = 525\,762$.

In order to test the capabilities of LLOOVIA to handle different limits, this analysis uses a limit of 50 VMs per type for on-demand VMs, with no more than 100 VMs in total per region, and 100 VMs per availability zone for reserved VMs.

Fig. 12 shows how the creation and solving time increases with the number of bins. The rightmost point, which corresponds to the case with no bins, is aborted because the optimal solution cannot be found before the time limit, set to 1 hour. That means that without bins, the optimal solution could not be found in the allotted time, so we will use the best lower bound of the cost for the comparison.

Phase II was carried out using the same access data used as LTWP as load prediction for each slot (STWP). Fig. 13 shows the increase in cost compared to the lower bound of the optimal solution with the different number of bins. Using as few as 20 bins, although the number of reserved VMs is not optimal, the cost of this solution is just 0.14% higher than the lower bound of the optimal solution.

From Fig. 12, it can be seen that the creation and solving time is significantly increased when more bins are used. On the other hand, Fig. 13 shows that the cost of the solution with bins is very close to a lower bound of the optimal solution obtained without bins when more than 20 bins are used.

5.3.3. Analysis of the influence of error in the prediction

This analysis is similar to the one carried out in section 5.2.3, but this time using the real workload from Wikipedia and the same regions and limits as in Section 5.3.2.

In the first experiment in this section, the Wikipedia data for 2012 is used as a prediction of the load for 2013 (see Fig. 14, which shows the mean number of accesses aggregated per year). This emulates a situation where a site uses the load for the previous year as forecast for the load of the following year. This can be considered a naive predictor, which can be used in the absence of more advanced prediction techniques that would obtain a better forecast. Using such a naive predictor is an adverse situation for LLOOVIA.

Phase I of LLOOVIA is applied to the load of 2012 grouped in 40 bins (because, as shown in previous analyses, with this number of bins the problem is solved fast and with good precision) and the number of reserved VMs is obtained. Then, Phase II is applied using the load of 2013 as STWP, and the number of required on-demand VMs, in addition to the previously computed reserved ones, is computed.

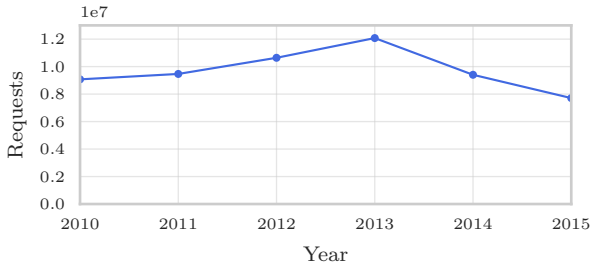

Figure 14: Mean number of requests per hour to the English Wikipedia per year

	Perfect predictor	Naive predictor	No predictor
2012 to 2013	$418 067.59	$423 324.97	$716 580.40
2013 to 2014	$317 770.45	$363 389.28	$542 704.17

Table 9: Influence of LTWP on the costs of the Wikipedia case study

To compare the quality of the result obtained this way, two other values are computed, also using 40 bins:

- Only on-demand: optimal cost using only on-demand VMs. This emulates a situation where no prediction is available, so it is not possible to find the optimal number of reserved VMs. To compute this value, only Phase II is applied, using zero reserved VMs as the result of Phase I.
- Perfect predictor: optimal cost in 2013 if the load were known. To compute it, Phase I and Phase II are applied with the load of 2013 as both LTWP and STWP to obtain the optimal number of VMs. In a real scenario this value cannot be obtained because the load is not exactly known beforehand.

The first row of Table 9 shows the results for this experiment. The cost obtained with a naive predictor is very close to the one obtained with the perfect predictor (only 1.25% higher) and almost half of using no predictor (i.e., only on-demand VMs), which is 71.40% higher than the perfect predictor cost.

The second experiment in this analysis is similar to the first one, but in this case the naive predictor uses the load in 2013 to forecast the load in 2014 (see Fig. 14). This allows studying an inverse relation between the forecast and the real load: in the first experiment the real load was higher than the forecasted one, while in the second experiment the real load is lower than the forecasted one. In addition, the change in the load is significantly bigger and, as the load is smaller, fewer reserved VMs are needed, so the savings are smaller.

The second row of Table 9 shows that the cost with the naive predictor is 14.35% higher than the perfect predictor, but much lower than using no predictor, which is 70.78% higher than the perfect predictor.

These two experiments demonstrate that significant savings can be obtained with LLOOVIA, even with naive forecasting techniques. It should be noted that the savings could be improved if better load forecasting techniques were used, but this is outside the scope of this paper.

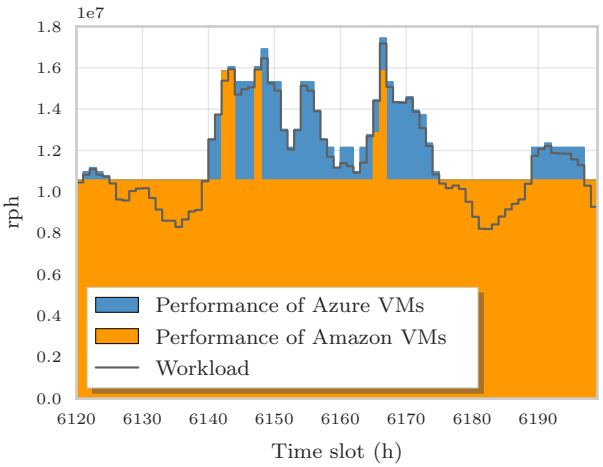

Figure 15: Performance given by each cloud provider vs. workload, in the multi-cloud scenario

5.3.4. Analysis in multi-cloud environment

This analysis shows how LLOOVIA performs in a multi-cloud environment. Although the previous analyses could be considered as multi-cloud environments, because the virtual machines of different regions of Amazon EC2 have different prices and performances, this analysis introduces another provider, Microsoft Azure, which adds a different price schema and different ways of setting limits.

Regarding the price schema, Azure does not have reserved VMs, but has a discount of 5% if a purchase larger than $6000 is paid upfront for a 12-month term. To take this into account, LLOOVIA can be applied with the discount; if the total amount to purchase from Azure according to the solution is less than the discount threshold, then that solution is not valid and LLOOVIA should be applied again with the non-discounted price to find the correct solution.

Limits in Azure are based on the number of cores rather than on the number of VMs, so this case study exercises a part of LLOOVIA that was not tested in previous analyses.

As the performance data for WikiBench in Azure is not available, a different benchmark was selected: specifically, OLDIsim, because it is a web benchmark integrated in Perfkit Benchmarker [37], a tool that makes executing benchmarks in several cloud providers easier. OLDIsim is aimed at obtaining a scaling factor when increasing the number of workers used. We modified it to report the number of queries per second (QPS) using only one worker and executed it in different VM types of EC2 and Azure. The performance obtained for each one is shown in Table 10.

The experiment uses two clouds: one region in Azure and another region, with three availability zones, in EC2, and uses the default limits set by the providers: 20 cores in total in the Azure region, 20 machines of each type for

Provider	VM type	QPS
Amazon	c4.2xlarge	2592.50
	c4.xlarge	1468.00
	m4.2xlarge	2510.00
	m4.large	636.00
	m4.xlarge	1316.00
Azure	A5	260.67
	A6	594.00
	Basic_A0	56.67
	Basic_A1	81.33
	Basic_A2	178.00
	Basic_A3	657.33
	ExtraSmall	54.67
	Large	561.33
	Medium	299.00
	Small	109.00
	Standard_D11_v2	660.00
	Standard_D12_v2	1621.67
	Standard_D1_v2	440.67
	Standard_D2_v2	682.33
	Standard_D3_v2	1560.00

Table 10: Number of queries per second (QPS) different Azure and Amazon VM types can handle for OLDIsim

	Cost ($)	% increase (vs. multi-cloud)
Only Azure	$3490.99	42.45%
Only Amazon	$2582.30	5.37%
Multi-cloud	$2450.69	-

Table 11: Comparison of costs with multi-cloud

on-demand VMs in the Amazon region, and 20 reserved machines in each availability zone for reserved VMs in Amazon.

The Wikipedia load of 2014 was used both as STWP and LTWP. 40 bins were used in Phase I. To assess the benefit of using a multi-cloud environment, tests using only the Azure region and using only the Amazon region were carried out in addition to the multi-cloud test. Table 11 shows that a multi-cloud solution is the most economical, around 5% cheaper than using only Amazon and almost 42.5% cheaper than using only Azure. The reason for this is that, in general, Amazon VMs are cheaper than Azure VMs, especially the reserved VMs, which Azure does not provide, so in the multi-cloud environment most of the machines used are from Amazon and very few are from Azure, as can be seen in Fig. 15, which shows the load and the performance obtained from Amazon and Azure machines in time slots of the year where the load is high (between hour 6120 and 6200, considering hour 0 to be the first hour of Jan 1 2014). For some time slots with low load, only the reserved instances in Amazon were used. With higher loads, LLOOVIA selects the best combinations of machines, sometimes using only Azure or Amazon on-demand VMs, sometimes using both.

This experiment shows that LLOOVIA is able to take advantage of the varied types of machines offered by different cloud providers in multi-cloud environments, while respecting the limits.

6. Conclusions and future work

Cloud Computing has evolved widely in the last years, but the problem of how the user makes cost-effective use of cloud services is still a very important subject. In this paper we have presented LLOOVIA, an allocation strategy which determines the number, type, price schema and the optimal allocation of the VMs required by a service to minimize cost in a multi-cloud environment.

The optimization problem was formulated using integer linear programming, where the number and the type of VMs are unknown. The problem constraints were the performance that the system has to meet and the maximum number of VMs that could be leased. The problem was solved in two phases. Phase I determines the number and type of the reserved VMs to lease at the beginning of each reservation period; this phase is carried out offline and working with a long-term workload prediction. Phase II determines the number and type of on-demand VMs to lease at each time slot; this phase is carried out online for every time slot and working with a short-term workload prediction.

LLOOVIA uses an approximation based on binning to substantially reduce the size of the integer linear programming problem to solve. A broad experimental work, using seven kinds of workloads combined with four workload levels, has been carried out to analyze the influence of binning and the error in the workload prediction. The analysis has shown that, with the exception of the unpredictable kind of workload, the approximation is very effective: lower than 0.24% error in most of the analyzed cases using an aggregation of as few as 20 bins. In addition, binning reduces greatly the problem resolution time and makes it possible to solve problems that could not be solved without it.

A similar experimental analysis was used to find an allocation for the real workload represented by Wikipedia for a reservation period of one year. In this case the binning approach produces an error lower than 0.14% using as few as 20 bins. Finally using the OLDIsim benchmark, LLOOVIA was tested in a multi-cloud environment with the limits and price schemes used by different providers. It was proved to be highly effective.

Future work has two main directions: firstly, to investigate new approximations and heuristics to avoid the limitations of integer linear programming. Secondly, to further increase the realistic working conditions of LLOOVIA. This last aspect will take a twofold dimension: considering the allocation for applications made up of more than one component (that is, different VM images), and taking into account extra requirements for the VMs, such as type of operating system or amount of memory. All these

working conditions will be incorporated into the model as constraints to the problem.

Acknowledgements

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

References

[1] Amazon, Amazon web services (AWS) - cloud computing services (2016) [cited 2016-06-15].
URL http://aws.amazon.com/

[2] Microsoft, Microsoft Azure (2016) [cited 2016-06-15].
URL https://azure.microsoft.com/en-us/

[3] Google, Google Cloud Platform (2016) [cited 2016-06-15].
URL https://cloud.google.com/

[4] Amazon, Amazon EC2 reserved instances (2016) [cited 2016-06-15].
URL https://aws.amazon.com/ec2/purchasing-options/reserved-instances/?nc1=f_ls

[5] Amazon, Amazon elastic compute cloud (amazon EC2) limits (2016) [cited 2016-06-15].
URL URL:http://docs.aws.amazon.com/general/latest/gr/aws_service_limits.html#limits_ec2

[6] Microsoft, Azure subscription and service limits, quotas, and constraints (2016) [cited 2016-06-15].
URL https://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/

[7] Amazon, Amazon EC2 pricing (2016) [cited 2016-06-15].
URL https://aws.amazon.com/ec2/pricing/

[8] S. Chaisiri, Bu-Sung Lee, D. Niyato, Optimal virtual machine placement across multiple cloud providers, IEEE, 2009, pp. 103–110. doi:10.1109/APSCC.2009.5394134.

[9] S. Chaisiri, B. S. Lee, D. Niyato, Optimization of resource provisioning cost in cloud computing, IEEE Transactions on Services Computing 5 (2) (2012) 164–177. doi:10.1109/TSC.2011.7.

[10] C. C. T. Mark, D. Niyato, T. Chen-Khong, Evolutionary optimal virtual machine placement and demand forecaster for cloud computing, in: International Conference on Advanced Information Networking and Applications, IEEE, 2011, pp. 348–355. doi:10.1109/AINA.2011.50.

[11] S. Yousefyan, A. V. Dastjerdi, M. R. Salehnamadi, Cost effective cloud resource provisioning with imperialist competitive algorithm optimization, in: 2013 5th Conference on Information and Knowledge Technology (IKT), IEEE, 2013, pp. 55–60. doi:10.1109/IKT.2013.6620038.

[12] A. Nodari, J. K. Nurminen, 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. doi:10.1145/2851613.2851869.

[13] C. Fehling, F. Leymann, R. Retter, W. Schupeck, P. Arbitter, Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications, Springer Publishing Company, Incorporated, 2014.

[14] R. Buyya, J. Broberg, A. Gościński (Eds.), Cloud Computing: Principles and Paradigms, Wiley, Hoboken, N.J, 2011.

[15] S. S. Manvi, G. Krishna Shyam, Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey, Journal of Network and Computer Applications 41 (2014) 424–440. doi:10.1016/j.jnca.2013.10.004.

[16] Z.-u. Rehman, O. K. Hussain, F. K. Hussain, User-side cloud service management: State-of-the-art and future directions, Journal of Network and Computer Applications 55 (2015) 108–122. doi:10.1016/j.jnca.2015.05.007.

[17] Z. A. Mann, Allocation of virtual machines in cloud data centers—A survey of problem models and optimization algorithms, ACM Computing Surveys 48 (1) (2015) 1–34. doi:10.1145/2797211.

[18] R. Weingärtner, G. B. Bräscher, C. B. Westphall, Cloud resource management: A survey on forecasting and profiling models, Journal of Network and Computer Applications 47 (2015) 99–106. doi:10.1016/j.jnca.2014.09.018.

[19] Z. Rehman, F. Hussain, O. Hussain, Towards multi-criteria cloud service selection, in: Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on, 2011, pp. 44–48. doi:10.1109/IMIS.2011.99.

[20] M. Whaiduzzaman, A. Gani, N. B. Anuar, M. Shiraz, M. N. Haque, I. T. Haque, Cloud service selection using multicriteria decision analysis, The Scientific World Journal 2014 (2014) 1–10. doi:10.1155/2014/459375.

[21] G. A. Geronimo, R. B. Uriarte, C. B. Westphall, Towards a framework for VM organisation based on Multi-Objectives, 15th IC on Networks–ICN. IARIA (2016) 1–6doi:10.13140/RG.2.1.1645.4160.

[22] B. Xu, Z. Peng, F. Xiao, A. M. Gates, J.-P. Yu, Dynamic deployment of birtual machines in cloud computing using multi-objective optimization, Soft Computing 19 (8) (2015) 2265–2273. doi:10.1007/s00500-014-1406-6.

[23] M. E. Frincu, S. Genaud, J. Gossa, On the efficiency of several VM provisioning strategies for workflows with multi-threaded tasks on clouds, Computing 96 (11) (2014) 1059–1086. doi:10.1007/s00607-014-0410-0.

[24] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, I. M. Llorente, Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers, Future Generation Computer Systems 28 (2) (2012) 358–367. doi:10.1016/j.future.2011.07.003.

[25] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, I. M. Llorente, Scheduling strategies for optimal service deployment across multiple clouds, Future Generation Computer Systems 29 (6) (2013) 1431–1441. doi:10.1016/j.future.2012.01.007.

[26] W. Wang, D. Niu, B. Liang, B. Li, Dynamic cloud instance acquisition via IaaS cloud brokerage, IEEE Transactions on Parallel and Distributed Systems 26 (6) (2015) 1580–1593. doi:10.1109/TPDS.2014.2326409.

[27] U. Bellur, A. Malani, 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, 2014, pp. 689–696. doi:10.1109/CLOUD.2014.97.

[28] S. N. Srirama, A. Ostovar, Optimal resource provisioning for scaling enterprise applications on the cloud, in: Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on, IEEE, 2014, pp. 262–271. doi:10.1109/CloudCom.2014.24.

[29] X. Nan, Y. He, L. Guan, Optimal allocation of virtual machines for cloud-based multimedia applications, in: Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on, IEEE, 2012, pp. 175–180. doi:10.1109/MMSP.2012.6343436.

[30] B. Kavitha, P. Varalakshmi, Cost optimization using hybrid evolutionary algorithm in cloud computing, Research Journal of Applied Sciences, Engineering and Technology 10 (7) (2015) 758–769.

[31] A. Nodari, Cost optimization in cloud computing, Master's thesis (2015).

[32] R. V. D. Bossche, K. Vanmechelen, J. Broeckhove, Optimizing IaaS reserved contract procurement using load prediction, in: 2014 IEEE 7th International Conference on Cloud Computing, IEEE, 2014, pp. 88–95. doi:10.1109/CLOUD.2014.22.

[33] L. Thai, B. Varghese, A. Barker, Executing bag of distributed tasks on the cloud: Investigating the trade-offs between performance and cost, in: Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on, IEEE, 2014, pp. 400–407. doi:10.1109/CloudCom.2014.29.

[34] J. Yang, C. Liu, Y. Shang, B. Cheng, Z. Mao, C. Liu, L. Niu, J. Chen, A cost-aware auto-scaling approach using the workload prediction in service clouds, Information Systems Frontiers 16 (1) (2014) 7–18. doi:10.1007/s10796-013-9459-0.

[35] T. Lorido-Botran, J. Miguel-Alonso, J. A. Lozano, A review of auto-scaling techniques for elastic applications in cloud environments, Journal of Grid Computing 12 (4) (2014) 559–592. doi:10.1007/s10723-014-9314-7.

[36] Amazon, Regions and availability zones (2016) [cited 2016-07-12].
URL http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html

[37] PerfKit Benchmarker team, Perfkit benchmarker (2016).
URL https://github.com/GoogleCloudPlatform/PerfKitBenchmarker

[38] CBC team, COIN-OR branch-and-cut MIP solver (2016) [cited 2016-06-15].
URL https://projects.coin-or.org/Cbc

[39] PuLP team, PuLP: an LP modeler written in python (2016) [cited 2016-06-15].
URL https://projects.coin-or.org/PuLP

[40] E.-J. van Baaren, WikiBench: A distributed, Wikipedia based web application benchmark, Master's thesis (2011).
URL http://www.wikibench.eu/

[41] Flux7 Labs, Performance comparison of wikibench on 'm3' instances (2014) [cited 2016-06-27].
URL http://blog.flux7.com/performance-comparison-of-wikibench-on-m3-instances

[42] Flux7 Labs, Wikibench performance on 'c3' instances (2014) [cited 2016-06-27].
URL http://blog.flux7.com/blogs/benchmarks/wikibench-performance-on-c3-instances

[43] Wikitech, Analytics/data/pagecounts-raw — Wikitech (2015) [cited 2016-06-27].
URL https://wikitech.wikimedia.org/w/index.php?title=Analytics/Data/Pagecounts-raw&oldid=243874