# Accepted Manuscript


Reducing the price of resource provisioning using EC2 spot instances with prediction models

Javier Fabra, Joaquín Ezpeleta, Pedro Álvarez

PII: S0167-739X(18)31166-X
DOI: https://doi.org/10.1016/j.future.2019.01.025
Reference: FUTURE 4718

To appear in: *Future Generation Computer Systems*

Received date : 13 May 2018
Revised date : 25 October 2018
Accepted date : 15 January 2019

# Reducing the Price of Resource Provisioning using EC2 Spot Instances with Prediction Models

Javier Fabra, Joaquín Ezpeleta, Pedro Álvarez

*Aragón Institute of Engineering Research (I3A)*
*Department of Computer Science and Systems Engineering*
*Universidad de Zaragoza, Spain*

**Abstract**

The increasing demand of computing resources has boosted the use of cloud computing providers. This has raised a new dimension in which the connections between resource usage and costs have to be considered from an organizational perspective. As a part of its EC2 service, Amazon introduced spot instances (SI) as a cheap public infrastructure, but at the price of not ensuring reliability of the service. On the Amazon SI model, hired instances can be abruptly terminated by the service provider when necessary. The interface for managing SI is based on a bidding strategy that depends on non-public Amazon pricing strategies, which makes complicated for users to apply any scheduling or resource provisioning strategy based on such (cheaper) resources. Although it is believed that the use of the EC2 SIs infrastructure can reduce costs for final users, a deep review of literature concludes that their characteristics and possibilities have not yet been deeply explored. In this work we present a framework for the analysis of the EC2 SIs infrastructure that uses the price history of such resources in order to classify the SI availability zones and then generate price prediction models adapted to each class. The proposed models are validated through a formal experimentation process. As a result, these models are applied to generate resource provisioning plans that get the optimal price when using the SI infrastructure in a real scenario. Finally, the recent changes that Amazon has introduced in the SI model and how this work can adapt to these changes is discussed.

*Keywords:* Cloud computing, Provisioning, Spot Instances, Amazon EC2, cost constraints

## 1. Introduction

The cloud-computing paradigm has changed the traditional way in which software systems are built by means of the introduction of a new model in which infrastructures, platforms, applications and services are served on demand [1]. The consolidation of this new approach in the industry as well as in research and academic environments has arisen the need to reconsider the way technological resources are used in organizations, integrating cloud-computing along with these resources [2, 3, 4, 5].

The cloud-computing approach promotes an on-demand model for the provisioning of resources such as virtual servers, services or an application platform, for instance. This model is being widely adopted because of the features it offers, such as elasticity, flexibility or pay-per-use. At the same time, Infrastructure-as-a-Service (IaaS) providers have introduced some additional variables related to price, performance and reliability in the resources located on the cloud. These providers deploy cloud resource management systems in data centers distributed worldwide. Some of these providers also offer a special type of computing resource in order to take advantage of unused cycles on their datacenters looking at maximizing their benefits. The price of these resources can vary over time and can provide with important savings with respect to the corresponding on-demand alternatives. The most well known cases of this practice are the Google Cloud Preemptible Virtual Machine Instances and Amazon EC2 Spot Instances (EC2 SI) [6, 7].

The Google Cloud Preemptible Virtual Machine model is based on the limited provisioning of instances, which are available at certain instants of time depending on the data center load. The price of these instances is fixed, being significantly lower than the on-demand model. Once the user launches an instance, it can be running during a maximum of 24 hours. The instance can be terminated at any time by the provider with a previous notification that allows saving or moving the processes and data carried out. On the other hand, Amazon Spot instances are offered through an auction mechanism. The maximum price willing to pay for a SI as well as other constraints are set. Then, an auction process is carried out and the instance is launched in case the request is fulfilled. Otherwise, the request is postponed until both conditions are fulfilled or the user withdraws the bid. The instances will run until either they are terminated by the user or the provider preempts the resources because of instance market fluctuations.

While the Amazon EC2 SI model allows setting the maximum cost for a particular instance and it does not impose any execution time limit, the model of Google Cloud Preemptible Virtual Machine has fixed prices and an execution time limit of 24 hours. In both models, the user must assume the risk that the execution can be terminated at any time. However, Amazon SI model is stricter than Google's one, since the mechanism of expulsions in the last one depends solely on the policy established by Google and not relies on the variations of the spot market. In both cases users are responsible for implementing the necessary checkpointing mechanisms as the way of avoiding data loss.

Although SIs do not ensure a reliable execution, a good analysis and offer strategy can drastically reduce the execution costs of systems when compared to on-demand costs (between a 50% and a 90%) [8]. The capacity and performance of applications could be increased with the same budget, or even allow the use of new applications or configurations previously discarded because of economic reasons. The use of SIs perfectly fits on a vast variety of scientific computing experiments, from genomic sequence analysis to data distribution, physic simulations or bioinformatics, for instance. From an enterprise point of view, there also exist some companies that take advantage of the use of SIs, such as Yelp, NASA JPL, FINRA, and Autodesk. DNAnexus is an application case that bases their systems on the use of spot instances to carry out genomic analysis and clinical studies on a highly scalable environment [9]. Netflix is also a well-known case of the multimedia industry. They use SIs in order to improve the broadcast of billion of data of their contents network [10].

In this work we propose the analysis of Amazon EC2 Spot Instances mechanisms to provide a history-based pricing model allowing final users to predict Amazon SI prices for the different availability zones. To this end, we have built a system that analyzes price variations on all regions and zones where SIs are offered. As a result, different models for price prediction are provided for the different zones. These models rely on the historic fluctuations of the SI market. We have used this system to define and execute real provisioning plans in different regions and moments. Given a deadline and cost constraints, the system provides the user with a complete overview of the suitability of using spot instances for the deployment of an experiment. We have also detected the existence of certain patterns in this variation that can be used to obtain a significant cost reduction. The main contributions of this work are the following:

- The proposed solution considers and analyzes the SI market during a long-term period, while previous studies only considered short periods of time.

- All availability zones and regions of Amazon SI are analyzed and classified, providing the most suitable price prediction model in each case.

- Provisioning plans are generated according to these models, allowing therefore a best cost execution of processes given a deadline and cost constraints.

- A user-oriented framework with such features is proposed. This framework allows running all the required processes and stages automatically, keeping models and data updated.

Recently, Amazon launched a new pricing model that simplifies the purchasing experience when dealing with SI[1]. Price variations have been now reduced and are less aggressive, but still allow savings similar to the previous mechanism. This has an obvious impact in the interruption mechanism as well, and longer workload runtimes may be possible. With this approach, Amazon tries to avoid the effort done in analyzing historical prices in order to adapt the bidding strategy. In this work we will also detail these new changes and how our proposal fits to them. However, for the sake of clarity we will keep the description of the previous SI mechanism along this paper, as this was the foundations of the work done.

The remainder of this paper is structured as follows. The technical background of Amazon SI is detailed in Section 2. After that, Section 3 presents related work on the analysis of EC2 Spot instances. The framework developed for the analysis of the EC2 SI infrastructure is introduced in Section 4. This framework is used to formally define different price prediction models in Section 5. The proposed models are then validated by means of the experimentation described in Section 6, and these models are used to carry out the generation of provisioning plans using EC2 SI in Section 7. The new pricing model that Amazon launched recently and how our approach adapts to the changes is then detailed in Section 8. Finally, Section 9 enumerates some conclusions and future work.

[1] https://aws.amazon.com/es/blogs/compute/new-amazon-ec2-spot-pricing/

## 2. Technical backgroung: The Amazon EC2 Spot Instances service

The SI provisioning model is part of the Amazon EC2 (Elastic Compute Cloud) service in the Amazon Web Services (AWS) ecosystem [7]. This service is responsible for providing cloud computing resources through its data centers located in three main areas: America, Asia Pacific and the last area composed of Europe, the Middle East and Africa. Each data center located in one of these areas is called a *region*, and offers its own service catalog.

Each region in Amazon EC2 is divided into one or more *availability zones*. An availability zone runs on its own separate and physically distinct infrastructure and is designed to offer high levels of reliability. Common failure points, such as power generators or cooling equipment, are not shared between the availability zones of the same region. In addition, they are located in different physical locations, so that in the event of any natural disaster occurring only a zone is affected. Therefore, each availability zone is totally independent, and the prices of the infrastructure they provide fluctuate in an independent way with respect to the others. As will be detailed later in Section 5, this characteristic allows observing curious behaviors related to costs within the same region.

At the end of 2009 Amazon launched a new type of instances at a lower cost compared to traditional on-demand instances, with the aim of getting better performances from its data centers by increasing the activity of their resources. Amazon offers these types of instances at a relatively lower cost but reserving the right of preempting the resources when needed by the service. Currently, *Amazon EC2 Spot Instances* or *Amazon EC2 SI* is the trade name that Amazon uses to name these instances, which are deployed in machines that are idle [8]. The number of SIs Amazon can offer in any given data center will depend on the number of on-demand instances it provides at each instant of time, offering the remaining capacity of the data center under the SI model. Amazon EC2 SIs are completely integrated onto the AWS ecosystem, being compliant with all services that currently AWS offers. Figure 1 depicts a scenario of use of SIs that considers typical AWS elements and services such as S3 (storage), auto-scaling (computing) or DynamoDB (databases).

This type of instances allows to increase the activity and productivity of data centers, as the number of idle machines can be reduced, and therefore decrease downtime. An auction model is used for the allocation of SIs that

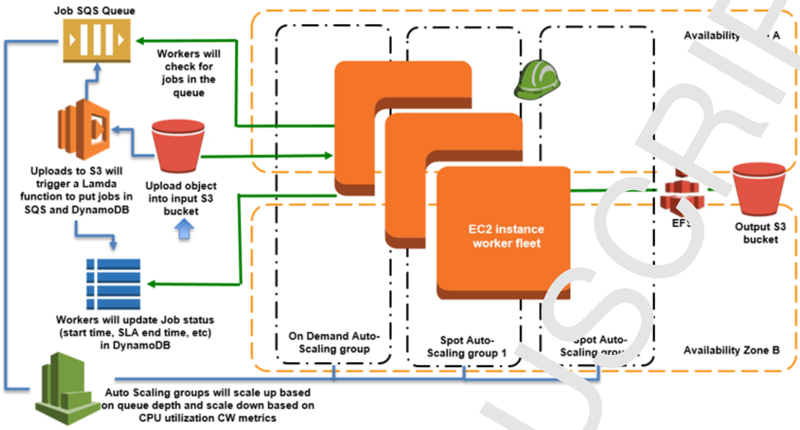


Figure 1: Abstract view of the process for using Spot Instances.

allows customers to participate by setting the maximum price they are willing to pay for a particular instance, called *bid price*. Additionaly, the request will include other parameters such as the type of instance and the data center in which it will be deployed, as sketched in Figure 2.

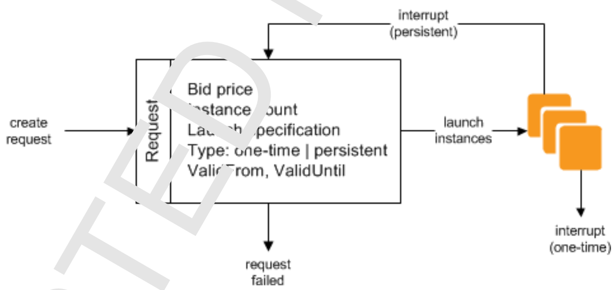


Figure 2: Spot Instance request process.

AWS continuously evaluates the *spot market*, analysing how many SIs are available in each *spot instance pool*, monitoring the bids that have been made for each pool and provisioning the available SIs to the highest bidders. If the requested instance is available and the bid exceeds the auction price, called *spot price*, the instance will be immediately hired. However, if the spot price exceeds the price set by the customer, the request will remain waiting for the imposed conditions to occur. The SIs created will run until either they are terminated by the user or there is a *spot instance termination*, which is a termination forced by the Amazon EC2 service mainly due to spot market fluctuations. Once a SI has been marked for termination, a *termination notice* is sent. This is a two-minute warning window before it terminates. The user is responsible for implementing the relevant checkpointing mechanisms to avoid the loss of important information.

Amazon offers a set of tools allowing users to constantly monitor and keep track of the price. These tools are intended to assist the end user in making decisions regarding SIs, selecting the appropriate instance type, an appropriate bid price allowing the application to run longer, and so on. Among these tools is the *Spot Instance Pricing History* [11], depicted in Figure 3, which relates the evolution of prices of a specific instance type over time: a graph with the behavior of prices showing their volatility and the frequency with which peaks occur in the auction. A limitation of this tool is that it only offers information from the last 90 days. This prevents the study of cyclic or temporal behaviors that repeat over time in long-term, and which can also decisively influence the behavior of the auction.

Another tool offered by Amazon is the *Spot Bid Advisor* [12], which allows analyzing auction price histories and helping the user to determine a bid price fitting their requirements. This tool also displays the frequency with which the bids are exceeded for each type of instance. This information can help the user to set an appropriate bid strategy because the lower the frequency with which they exceed the bid in an instance type the more likely it is to run without interruption.

Finally, the *Spot Block Model* [13] is a mechanism that allows guaranteeing the availability of the instance for a time up to a maximum of 6 hours, thus increasing the versatility of the SI mechanism. The operation is identical to the previous one, except that the user has the possibility to establish the amount of minutes that he wants the instance to be running, and the maximum price he is willing to pay per computation hour. Amazon evaluates the request, and once the capacity dedicated to SIs allows to guarantee the availability during the requested duration, Amazon will deploy it. The instance will end when the established duration is fulfilled, or sooner if the user decides to terminate it. This model is very interesting for those jobs or processes that need to run continuously for up to 6 hours.

On November 2017, Amazon announced at the re:Invent event some important changes in the SI mechanisms that are being gradually incorporated during year 2018. These changes try to give a synchronous view of the model and simplifies the way SIs work. The provider tries to avoid spending time on

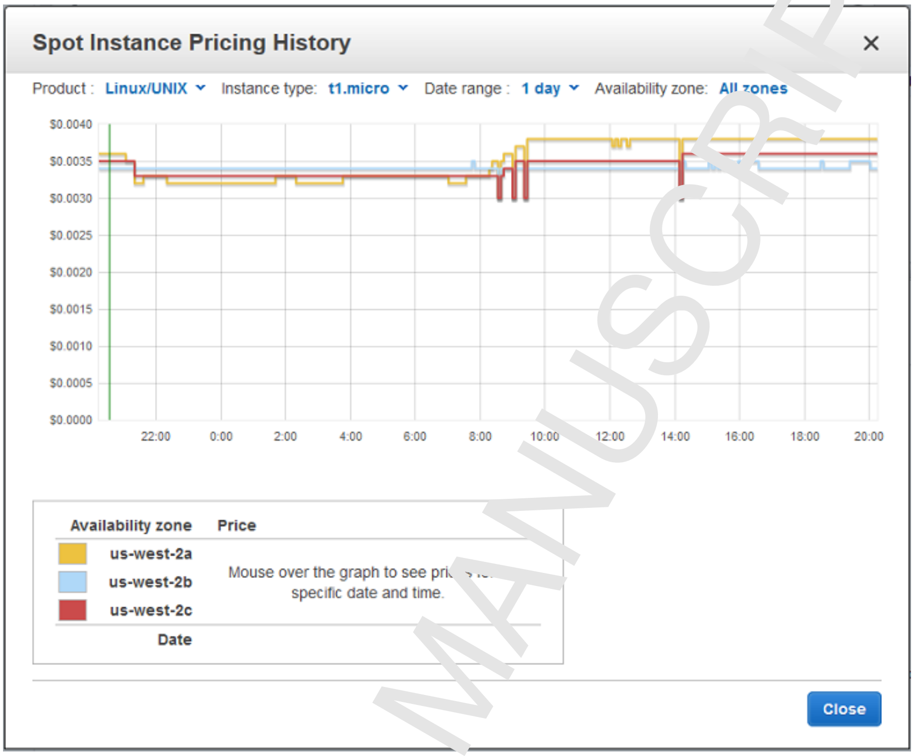


Figure 3: A screenshot of the Spot Instance Pricing History tool.

understanding spot markets, bidding strategies and other advanced details related to SIs. The way that spot prices change was moving to a model where prices adjust more gradually, and instance hibernation was also introduced. We will go over the new features and detail how the approach presented in this work fits with the new SI model and still allows important savings in Section 8.

## 3. Related work

The scientific community has shown a significant interest in the use of SIs for workloads that are either fault-tolerant or not-time-sensitive. However, the effective use of this mechanism requires the study and analysis of the fluctuation of prices over time. Some research has focused on the use of SIs to reduce computing costs when dealing with complex problems [14, 15]. In [14] authors are especially sensitive to the reliability of SIs, and manage checkpointing strategies to avoid data loss when instances are terminated because of overbidding. The economics of adding additional resources to dedicated clusters during peak periods was studied in [15]. Authors defined different provisioning policies based on the use of SIs and compared them to on-demand instances in terms of cost savings and total breach time of tasks in the queue.

Time price variation of SIs has been deeply studied in [16, 17], although authors have not given specific conclusions. [16] considers that prices vary on real time and there is not a pattern for this variation. On the other hand, a reverse engineering technique is used in [17] in order to build a price model based on auto-regression techniques. In [18] regression techniques are also used to analyze and predict the prices of SIs with a small data set. Authors propose to obtain the value of regression parameters using a gradient descent algorithm. The estimated price is computed by analyzing the current change in price using neural networks as well and an experimental set up based on Matlab scripting has been provided. The relation between Cloud Service Brokers and pricing is analyzed in [19], where authors discuss how performance variations in virtual machines of the same type and price raises specific issues for end users, which in the end affects the final price for resource allocation.

There are some papers that achieve a statistical analysis as well as a modeling of price variation of SIs. A very interesting approach is presented in [20, 21], where authors conducted an analysis of SI prices and its variations limited to four specific regions of Amazon EC2. All different types of SIs in terms of spot price and the inter-price time (time between price changes) as well as the time dynamics for spot price in hour-in-day and day-of-week were studied. Authors proposed the characterization of their behavior through a statistical model and evaluated it by means of simulation techniques.

A very interesting recent work is [22], in which the authors considered switching regimes of spot prices for forecasting, and propose a set of Markov regime-switching autoregressive based forecasting methods. In order to conduct the forecast price, a dynamic-autoregressive integrated moving average model is developed as well. The authors perform a clustering of the spot prices to determine the number of regimes when building their model. One of the conclusions obtained through its detailed work is that none of the proposed algorithms can predict the long-term prices for certain classes of prices where the regime switching pattern is hard to obtain. In all other cases, the predictions are very promising.

Other recent research works focus on the analysis of price variation and the proposal of models based on machine learning approaches. Some works have been based on the use of regression random forests (RRFs) to predict the prices of the SIs. In [23], authors use this approach in order to predict one-week-ahead and one-day-ahead spot prices, later extending it to longer-term predictions to demonstrate the effectiveness of their method. The authors also perform an evaluation of non-parametric machine learning algorithms with random forest based predictions, concluding in their case that prediction accuracy of Random Fo-rests outperform prediction accuracy of Neural Networks and Support Vector Machines in a period of 120-150 days forecasts. A very detailed and comprehensive state of art about the use of machine learning techniques in the SI price prediction scenario can be found in [23].

On the other hand, the use of recurrent neural networks (RNNs) to provide better accuracy than standard statistical approaches has been a topic very studied in the literature. The use of long/short-term memory (LSTM) recurrent neural networks to predict the prices of SI has given very good results in [24], allowing margins of error of 5%. The use of LSTM is based on the fact that the LSTMs are able to identify and remember the latent features over an unspecified number of time periods, making them a versatile tool in time series prediction. In [25] the use of a neural network-based back propagation algorithm to use the past spot pricing history is proposed. The authors use this technique to achieve an efficient scheduling in bag-of-tasks (BoT) problems. The results show that a very good error rate of between 5 and 6% is obtained, and a cost reduction of 38% in the experimentation carried out.

With respect to the generation of SI provisioning policies, authors propose in [26] a decision-based model to improve performances, costs and reliability under the restrictions imposed by a Service Level Agreement (SLA). In [27] the use of SIs is also proposed to improve a map-reduce execution system, and a Markov chain model is proposed to predict the lifetime of a running SI. Authors focus on fail situations and propose provisioning policies for these cases, which is also the base for the work presented in [15]. Similarly, in [28] a Constrained Markov Decision Process (CMDP) is formulated in order to derive an optimal bidding strategy. Based on this model, authors obtain an optimal randomized bidding strategy through linear programming. Finally, in [29] Markov spot price evolution is also analyzed. A job is modeled as a fixed computation request with a deadline constraint in order to formulate the problem of designing a dynamic bidding policy minimizing the average cost of job completion. Finally, the analysis of the bidding system of Amazon SI and the consequences of instance termination has been the focus of the research presented in [14, 30, 31, 32].

In this work we aim at the analysis of SI prices considering costs and time constraints. Most research has focused on the impact of SI termination and other aspects such as reliability. Other authors have concentrated their efforts to study general price variations in the spot market, considering specific availability zones and instance types. Our work focuses on the building of a provisioning plan for the final user with different options fitting his/her requirements and constraints. The proposed framework considers all available data for every availability zone as well as every instance type and operating system. In this sense, it can be considered as *global*. In this work we also provide with different price prediction models based on a regression technique and depending on the spot market fluctuations, which we demonstrate to be different on the availability zones for each instance type. Finally, it should be remarked that the models proposed here consider a long-term period of time, whereas existing work have considered short periods of a few weeks or months.

## 4. A framework for the analysis of Spot Instances

Let us now briefly describe the developed framework. The architecture of the system is depicted in Figure 4. There are three main levels: the *API level* (top of Figure 4, in blue), which represents the entry point to the system; the *service level* (EC2 SI components, in red), which connects the API layer with the underlying component; and the *process level* (components located at bottom of Figure 4, in green and transparent colour), which contains independent components in charge of downloading, storing or processing the information related to SIs.

The EC2 SI Launcher module, which is located in the service layer, is the module that allows deploying SIs on Amazon EC2 transparently. To do this, this module requires certain information such as the bid price, the availability zone where the instance has to be deployed, the type of instance and the operating system or the Amazon Machine Image (AMI) to use. Then, it uses two modules of the domain layer. On the one hand, the EC2 Data Storage module, which allows retrieving the information necessary for the deployment of SIs. On the other hand, the SI Launch Request module,

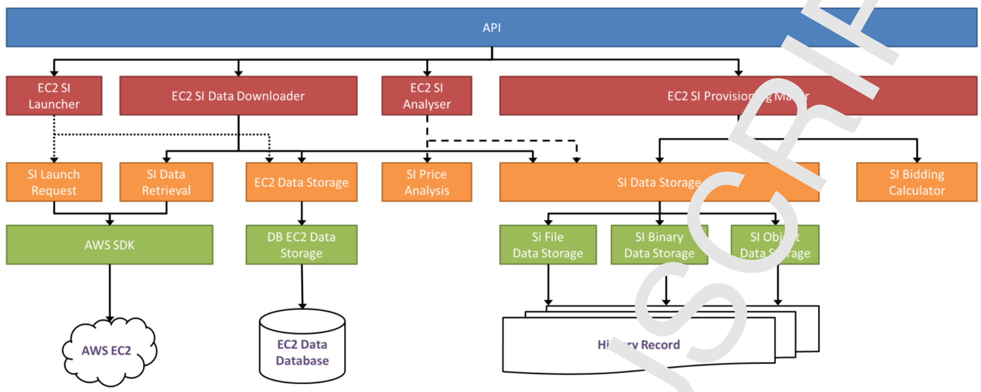


Figure 4: Framework architecture.

which is responsible for performing the corresponding bid through the web services interface offered by the EC2 API [33].

The EC2 SI Data Downloader module is responsible of downloading and storing price variations of any type of instance in every region and zone of EC2 using the previous API. This module runs periodically, once a week, recovering only the new data that has not been stored in the system yet and saving it by means of the Data Storage module. The EC2 SI Data Downloader module connects to the SI Data Retrieval module, which tracks the download as well as possible faults. In case of failure, the download data is marked as failed and the module communicates to the EC2 SI Data Downloader module. Then, the Data Retrieval module retries the download up to a maximum of three times. If the fault persists, the data will be marked as non-available and will not be used in future simulations or executions.

Once a download finishes, the Data Downloader module will process the available data and will store it properly. During the process, it adds some information to the download such as the number of price variations, the number of retries and the final state of the download for each type of instance. The full set of data is then stored in the database. The EC2 Data Storage module provides an interface to manage the information stored in this database.

The stored price variations will be used by the SI Pricer Analysis module to perform an analysis of the price variations over time considering different variables. It also allows a more detailed study of the price variations that have taken place in each region and availability zone for any type of instance, providing statistical information such as average deviation, average price, maximum and minimum price, etc. Both this module and the SI Data Storage module will be used by the EC2 SI Analyzer module to compose the service that allows the functionality of performing statistical analysis of price variations for a region or zone of availability with any type of instance and operating system.

The Provisioning Maker module is the module responsible for proposing a provisioning plan, which consists of a list of time periods for which the proposed execution is feasible. Given a deadline, an EC2 availability zone or region, an instance type, the number of required hours for execution and the maximum price per hour, the module will perform an internal simulation to generate a list of feasible daily times. A feasible time can be interpreted as the time for which the simulator estimates that the bid would be successful (this is, it could be accepted with a price less than or equal to the maximum price established), and therefore, it would be possible to obtain a SI for the selected type of instance also satisfying that it will not be preempted during the specified execution time (so it will finish its execution before the deadline expires). A first implementation of this component was previously presented in [34]. This work is going to focus on how the SI Pricer Analysis module conducts its analysis and price prediction as well as how the Provisioning Maker module integrates these results in order to generate provisioning plans based on EC2 SI.

Obviously, the specified maximum price is one of the key values in this process. The Bidding Calculator module is responsible for determining the best price at each time instant for a specific region, a type of instance and an operating system. This module provides a bid price using a model that allows predicting the future price of the bid based on a series of characteristics and explanatory variables that characterize the price evolution of the auction for a specific region. Due to the fact that price evolution of an auction depends on past prices this model will be generated using the registered history. In the next section, the model generation will be detailed.

Finally, all the services of the system are offered through the API module, which is responsible for offering the services through a REST interface. This allows to connect another systems in order to communicate, integrate and access the services of the framework. For instance, an advanced user module has been designed and developed in order to facilitate the interaction of users with the system [34]. This module exposes a Web application that consumes the REST services provided by the API. It facilitates the administration of the whole framework and allows using the different services such as the generation of provisioning plans or even launching instances using the SI mechanism.

## 5. Modeling EC2 Spot Instances provisioning costs

In this section, the process to obtain a suitable SI pricing model is detailed. This model will be used by the SI Bidding Calculator component and the SI Pricer Analysis module in order to carry out simulations and to generate provisioning plans. The aim of the model is to allow predicting the future spot price over time in order to locate a SI resource at low cost. In addition, the model will also facilitate getting better provisioning plans than the simple use of the price of the on-demand instances (EC2 price) or the current spot price. The system will be able then to select the ones that meets cost and time requirements for a given set of constraints.

### *5.1. Methodology*

Figure 5 sketches the methodology proposed for the modeling of SI provisioning costs. First, SI prices are retrieved from the EC2 Data Storage dabase depicted in Section 4. These prices are then processed in order to classify them and to provide with an homogeneous representation of price variations. During the preprocessing stage the hourly price and some statistical data are generated. All the information available is then split considering the operating system, the instance type, the region and the availability zone. Finally, the extended information about the pricing histories is stored in a database.

For every type of instance and operating system a statistical analysis process of availability zones is performed, retrieving the corresponding data from the database. This analysis is complemented with the analysis of the evolution of spot prices, which allows the characterization of the availability zones. This characterization allows the identification of behavioral patterns and specific characteristics of each zone that could be incorporated to the final model by their explanatory capacity. As a result, a set of zone behavioral patterns are generated.

These patterns are then classified using a clustering technique, which generates a set of *zone classes*. Each zone has a unique behavior and characteristics, so a different price model has to be defined for each class. This is an important step in the proposed approach, as providing a different model

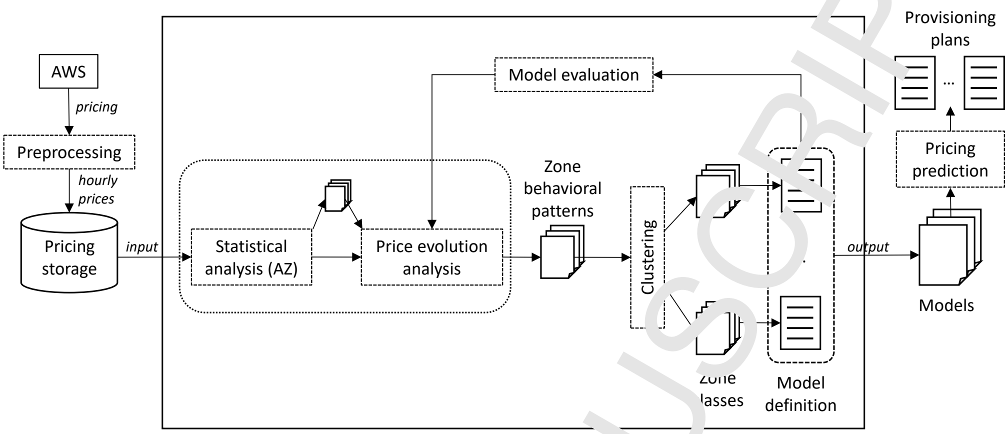


Figure 5: Process followed for the modeling of SI provisioning costs.

for each class differentiates this one from other works. As it will be detailed, each model is faithfully adapted to the class it represents, and has its own characteristics that justify the creation of as many models as classes are identified. Finally, the obtained models are used to generate the SI provisioning plans, as it will be detailed on next section.

The whole process is fully automated and the approach presented in the paper is applicable to different types of machines and operating systems. To this end, the system has been deployed in a cloud environment using EC2 on-demand instances for the long-term and stable components and SIs for those modules that execute punctually. This allows us to keep and updated system available through a graphical interface as well as through its REST-like API, while maintenance costs are lower than keeping the whole system running over on-demand instances. There is a huge amount of information to manage if we consider the whole set of availability regions, zones and instance types in Amazon EC2, so the Amazon Relational Database Service [35] is used to store both the downloaded data as well as the pricing information (once processed). Then, the system runs periodically to update the information stored as well as to respond to user's request to perform a price simulation or to generate a provisioning plan. It is even possible to deploy and run a spot instance directly using the graphical interface, as it is depicted in Figure 6.

There are several alternatives to implement the different stages depicted in Figure 5, such as using scripting or workflow-like tools. In this work, they have been implemented by means of a set of scripts that allow to automate

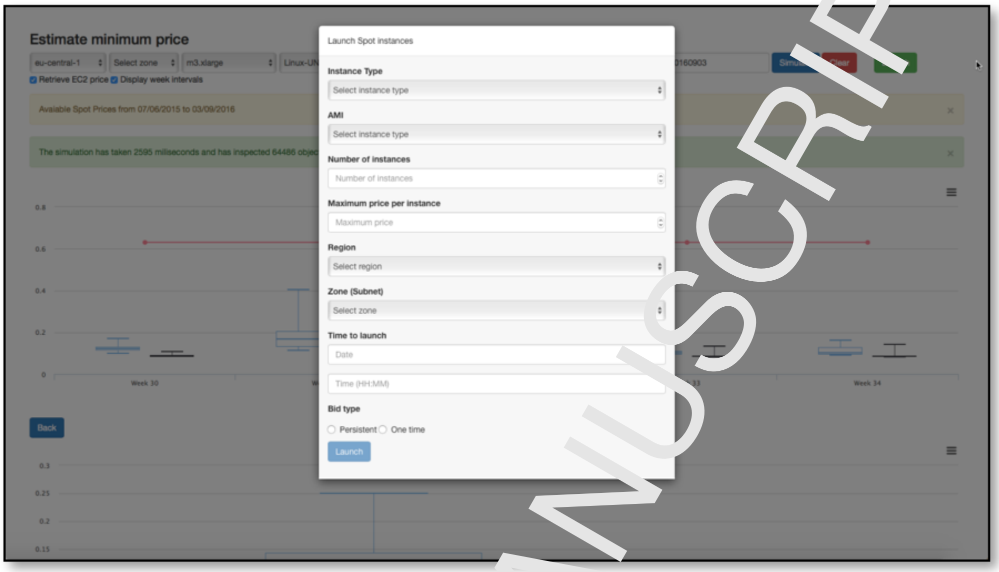


Figure 6: Deplyment of a SI using the graphical interface of the system.

the process through the different availability zones and SI types. These scripts are able to run a specific process or start an external development environment, such as the R statistical analysis tool, in order to carry out the clustering process or the construction and validation of the proposed models, for instance. This allows us to be able to manage different scenarios depending on user's input and being able to deal with the full collection of options available in EC2.

Amazon offers a broad catalog of different types of virtual machines and operating systems in each region. Specifically, we can differentiate among the following instance families: general purpose, compute optimized, memory optimized, storage optimized, accelerated computing (FGPUs) and, finally, bare metal. Each family then contains different configurations, allowing to fit the instance type to the requirements of specific problems. Instances can be configured to run a Linux-based or a Windows operating system, or even to run a pre-configured Amazon Image (AMI). AMIs contain custom configurations and allow to deploy specific services or images in a container-like way.

In this paper, the use of m3.xlarge instances running a Linux operating system was a requirement for the case use that will be presented in Section 7. Therefore, the study we are presenting concentrates on the m3.xlarge instances with that operating system in order to provide the reader with a specific instance type that, in the following, will drive the process depicted in Figure 5. m3.xlarge instances are under the General Purposes instances category, and provide a good trade-off among computing, memory and network resources, making it a good choice for many different applications. This causes that the m3.xlarge instances are present in the vast majority of the availability zones and, as it will be observed in next section, the spot price of this type of instance constantly fluctuates.

### 5.2. Preprocessing stage

Due to the wide magnitude of price variations to be analyzed, it has been necessary to apply some techniques to reduce the size of the problem. Multiple spot price variations can be produced during an hour. However, the only representative price for that hour will be the maximum value reached. One hour is the minimum unit of time that Amazon uses during auctions. Therefore, any bid that was below the maximum price registered for that hour would be either discarded or the instance evicted. This has reduced the initial dimension of the problem by using the maximum price of each hour. As an example, having 30, 665, 548 price variations for m3.xlarge machines from June 2015 to October 2017, only 64, 488 data values have been considered for each zone. However, applying reductions may require a pruning to be made. The reason is that Amazon applies a series of sweeps at certain moments of time, thus generating very high spot price peaks to evict as many instances as possible. This behavior responds to Amazon's internal policies and strategies regarding the use of VMs.

Before storing the price information in the database, the preprocessing stage also separates the information provided by AWS related to each instance type, operating system, region and availability zone. This will allow to ease and speed the access to the information stored during the following stages.

### 5.3. Characterization of the availability zones

As it was stated previously, characterizing availability zones is a step required to generate the model. The characterization process aims to identify behavioral patterns as well as common and specific characteristics of each zone. To do that, a statistical analysis is conducted for each availability zone in every region. This analysis calculates the minimum, average and maximum prices ($), as well as the standard deviation.

We mentioned before that we are using m3.xlarge/Linux instances to exemplify the proposed approach. According to this, Tables 1, 2 and 3 show the resulting information from the analysis process for these type of instances. There is a significant variance between the availability zones. We can distinguish three main trends with respect to spot price dispersion, varying from low spot price dispersion to zones in which dispersion is very high. The relation among the statistical variables as well as the percentiles (90% and 99%) allows to establish an initial classification of the availability zones. Table 1 depicts zones with lower price dispersion. Zones with medium dispersion are shown in Table 2. Finally, the zones with greater price variability with their corresponding statistical properties are shown in Table 3.

| **Zone** | **Min** | **Average** | **Max** | **90%** | **99%** | **STD** |
|---|---|---|---|---|---|---|
| ap-northeast-1c | 0.0404 | 0.0463 | 4.0500 | 0.0530 | 0.0681 | 0.0546 |
| ap-southeast-1b | 0.0422 | 0.0507 | 3.9200 | 0.0512 | 0.0557 | 0.1112 |
| ap-southeast-2a | 0.0401 | 0.0482 | 1.0000 | 0.0567 | 0.0916 | 0.0181 |
| ap-southeast-2b | 0.0402 | 0.0502 | 3.5000 | 0.0559 | 0.0885 | 0.1100 |
| eu-central-1b | 0.0401 | 0.0451 | 3.3200 | 0.0430 | 0.0530 | 0.0895 |
| eu-west-1a | 0.0401 | 0.0647 | 3.0800 | 0.0442 | 0.0556 | 0.2593 |
| eu-west-1c | 0.0401 | 0.0533 | 3.0800 | 0.0447 | 0.0993 | 0.1621 |
| sa-east-1a | 0.0401 | 0.0465 | 2.0000 | 0.0434 | 0.0503 | 0.0661 |
| sa-east-1c | 0.0401 | 0.0486 | 2.9900 | 0.0523 | 0.0625 | 0.0863 |
| us-east-1b | 0.0320 | 0.0433 | 0.3000 | 0.0526 | 0.0921 | 0.0212 |
| us-west-1a | 0.0321 | 0.0391 | 1.8750 | 0.0468 | 0.0674 | 0.0197 |

Table 1: Statistical analysis of the Amazon EC2 availability zones with low spot price dispersion.

Considering each availability zone individually, the statistics do not reveal whether the spot prices in a zone are stable over time, since the standard deviation depends on the average price and this is greatly influenced by the price peaks. For example, stable areas with few peaks but whose price is very high will have high deviations, while in less stable areas, those that have suffered a large number of peaks but of a less significant price will be lower.

| Zone | Min | Average | Max | 90% | 99% | STD |
|---|---|---|---|---|---|---|
| ap-northeast-1a | 0.0403 | 0.0505 | 0.6841 | 0.0568 | 0.1000 | 0.0513 |
| ap-southeast-1a | 0.0406 | 0.0528 | 3.9200 | 0.0733 | 0.1031 | 0.0757 |
| eu-west-1b | 0.0401 | 0.0483 | 3.0800 | 0.0469 | 0.2052 | 0.0737 |
| sa-east-1b | 0.0403 | 0.0550 | 1.9900 | 0.0581 | 0.2227 | 0.0908 |
| us-east-1a | 0.0334 | 0.0604 | 2.8000 | 0.0822 | 0.2800 | 0.1067 |
| us-west-1b | 0.0321 | 0.0462 | 3.0000 | 0.0549 | 0.1369 | 0.0553 |
| us-west-2b | 0.0324 | 0.0552 | 2.1000 | 0.0686 | 0.2800 | 0.0974 |

Table 2: Statistical analysis of the Amazon EC2 availability zones with medium spot price dispersion.

| Zone | Min | Average | Max | 90% | 99% | STD |
|---|---|---|---|---|---|---|
| eu-central-1a | 0.0404 | 0.1655 | 3.3200 | 0.1097 | 3.3200 | 0.5614 |
| us-east-1c | 0.0323 | 0.0604 | 2.8000 | 0.0612 | 0.4620 | 0.1734 |
| us-east-1e | 0.0322 | 0.1265 | 2.8000 | 0.1052 | 2.8000 | 0.4102 |
| us-west-2a | 0.0328 | 0.0644 | 2.3000 | 0.0710 | 0.5000 | 0.1292 |
| us-west-2c | 0.0326 | 0.0701 | 2.3000 | 0.0713 | 0.5000 | 0.1939 |

Table 3: Statistical analysis of the Amazon EC2 availability zones with high spot price dispersion.

The analysis also reveals that not only the evolution of prices in each region is different, but there are also important differences between the availability zones inside the same region. Each zone shows a unique behavior and the selection of the appropriate area is crucial to obtain higher cost reductions. Let us focus in the region of Virginia (us-east-1). In this region, the average spot price of zone us-east-1e (\$0.1265) is almost three times the price of the cheapest zone within the same region (us-east-1b, \$0.0433). A significant reduction in costs can be obtained by deploying SIs in the appropriate zone within the desired region. Another relevant consideration is the existence of zones where the price is stable over time, and others whose variations would discourage their use, as it can cause the number of expulsions to increase in a considerable way. It can also be deduced that there is not a clear relationship among zones of the same region.

The previous analysis also allows us to classify the zones according to their stability during the entire period of time recorded, but does not allow observing temporary patterns or the most recent price evolution. To do this, an analysis of the way prices evolve is conducted monthly, weekly and daily. This allows to automatically identify temporary patterns shared between zones as well as specific behaviors. Let us exemplify this with Figure 7, which shows the evolution of spot prices over time in the us-west-1a and us-west-1b zones. As it can be seen, we could intuitively validate the initial hypothesis about price stability in an area. us-west-1a zone (Figure 7-left) has a low spot price dispersion, whereas us-west-1b zone (Figure 7-right) has a medium one. Figure 7 shows a good example of peaks, trends and price variations for that zone over time.

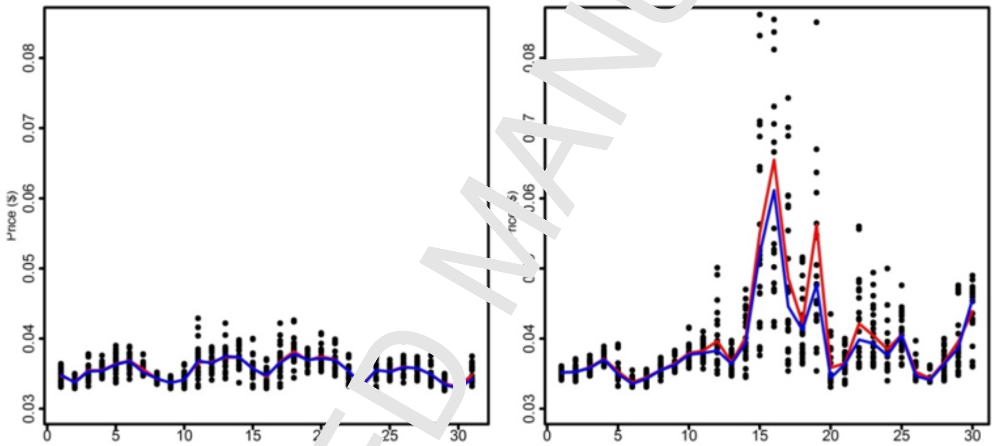


Figure 7: Time vs price variations and patterns observed for the us-west-1a (left) and us-west-1b (right) zones in June 2016. X-axis: price ($); X-axis: time (hours).

A more detailed analysis showed us that this behaviour is shared among other instance types and is not related to m3.xlarge instances exclusively. The previous statistical analysis is executed using Python scripts and the native numeric libraries. However, we have implemented some algorithms based on existing literature for segmentation and identification of patterns as well [36, 37]. These algorithms allow to conduct a more refined analysis of possible patterns in the data series we are considering. These patterns could help in predicting the spot price in a specific time, as they have a numeric representation by means of trigonometric functions when dealing with the model definition in Section 5.5. The existence of patterns in the spot price series has been previously identified in the literature [20].

A common temporary pattern that is evident when observing the monthly evolution of prices in consecutive months is the existence of a cyclical weekly pattern. Another observed pattern in the monthly evolution is the increase in prices on the last days of the month. While it is true that the weekly cycle is maintained in the last weeks of the month, the average price increases in certain zones. The most reasonable explanation is that different organizations use spot instances to carry out billing and payroll processes at the end of the month, which can cause the growth of demands and prices of SIs.

Let us now concentrate on the weekly evolution. This analysis reveals the existence of a weekly cycle common to the vast majority of the availability areas as discussed above. The identified pattern shows that prices rise at the beginning of the week and begin to decline when they reach the equator, reaching the lowest prices on weekends. One reason for that behavior could be that the demand is higher in working days, increasing the price. It can also be seen that the prices in worldwide holidays, such as Christmas or New Year, are lower than the prices that would correspond looking at the week day they happen. To corroborate such intuition, a subset of the initial data was been collected, consisting only of those data that were registered on Christmas and New Year (as they correspond to the same weekday). An aggregation function was used to compute the average price of the auctions in all availability zones for the data selected. Considering the obtained results, it can be deduced that the two days with the lowest average price registered correspond to New Year's Day and Christmas Day.

Finally, the daily evolution of auction prices has been analyzed as well. However, no clear common pattern has been established. On the one hand, there are areas whose prices rise at specific times of the day, others whose daily price remains stable, and some that do not follow any specific pattern due to the large dispersion of prices. Maybe the reason is that the service is used worldwide, which do not share day hours.

### *5.4. Clustering*

The results of the analysis phase show that the behavioral differences between the availability zones make the generation of a common model for all the zones difficult. Therefore, in this work we propose a partition and classification of the availability zones in groups that allows modeling each group of zones individually.

The number of subsets for each instance type is unknown and can vary over time. We propose the use of a clustering technique not requiring a prior knowledge on the established number of clusters. For that, the *hierarchical clustering* analysis technique has been used. This technique consists of establishing a metric or function of dissimilarity between the elements to classify. The choice of this metric will determine how close they are to each other. This grouping technique does not require establishing an initial number of groups, but groups the elements hierarchically generating a *dendrogram*. In order to establish the dissimilarity between each one of the zones, statistical information relative to the average, the percentiles and the standard deviations of the prices collected during the previous phases has been used and normalized.

The hierarchical clustering technique has been implemented in *R*. The resulting script is then executed for each instance type with the data contained in the system each time new data is added to the databases. However, the interpretation of the resulting hierarchical structure is context-dependent and from a theoretical point of view it is complex to determine which one is the best one. While an ad hoc interpretation can be achieved using visual criteria such as silhouette plots, this can not be easily automated. Therefore, we have applied the Hubert's gamma numerical criteria to determine the optimal number of clusters for each experimentation [38, 39]. Other well-known numeric criteria include Dunns validity index, G2/G3 coefficient, or the corrected Rand index, for instance.

Let us detail the clustering process by means of our running example. The dendrogram depicted in Figure 8 allows establishing a hierarchical grouping of the availabity zones for m3.xlarge spot instances. The height determines the distance between zones, and the height chosen to cut the dendogram determines the final grouping of zones. As it is shown in Figure 8, we obtained a height of $h = 4$. This clearly identifies three types of singular zone classes that will be detailed in the following.

The experimentation carried out with all the available data allowed us to observe that no more than three clusters were ever defined. The relationship between the statistical data for the availability zones (Table 1, 2 and 3 for the example of the m3.xlarge instances) and the identified clusters allow us to generalize the existence of three types of zone classes. These classes are dependent on the type of instance and the operating system we are analyzing, but common among all the family available in EC2. Let us identify these classes as *stable zones class*, *semi-stable zones class* and *unstable zones class*:

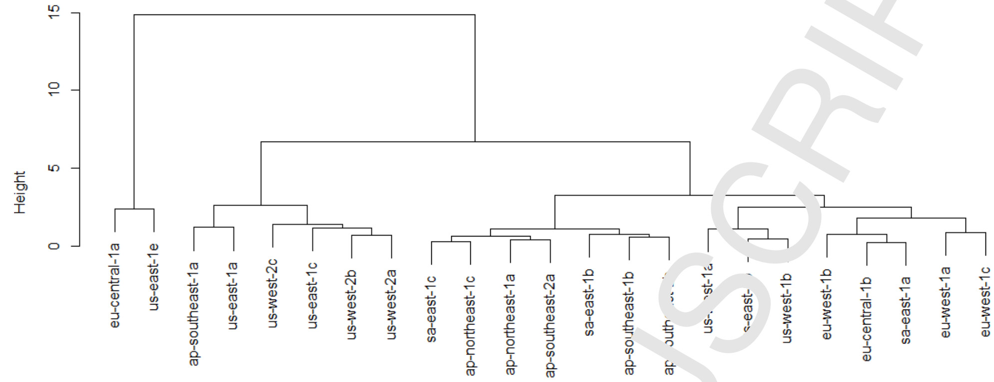


Figure 8: Dendogram of the hierarchical clustering of availability zones

- *Stable zones* are those that have few price variations, that is, those with sporadic peaks of demand and also whose maximum price is relatively low. A high percentage of their prices, around 99%, is between a small price range, showing a high probability for the price to stay in that range.

- *Unstable zones* are the ones with a high price variation. These areas have large spikes in demand that greatly alter the bid values. In addition, this type of zones neither have a clear common behavioral pattern, since they stand out due to the high dispersion of the prices. It is possible that these areas correspond to those that offer a smaller number of instances under the auction model and, therefore, the demand far exceeds the offer causing such price variations.

- Finally, the *semi-stable zones* are those that are framed between the two previous types. This type of zone has a higher price variation than the stable zones, and therefore, if the percentiles 90 and 99 are observed they are higher. They have a higher price fluctuation interval in addition to a greater number of peaks. But unlike the unstable zones, this type of zones do have a pattern of stable behavior that is repeated over time, since they do not have such dispersion of prices.

As it was previously mentioned, this characterization has been implemented in *R* as part of the hierarchical clustering scripts, allowing us to run it across all available instance types and operating system in our approach.

### *5.5. Model definition*

After the clustering process, a model is built for the zone classes that have been identified. The model generation starts from an initial *complex model* that includes those characteristics or explanatory variables that define the zones that compose each class. Those variables are the spot prices for the period stored in the database as well as the patterns identified for each zone, represented by means of trigonometric functions. This complex model is then refined applying a step-by-step linear regression technique. The regression analysis enables the identification of relationships among the parameters that compose the model. During the process, those parameters that do not contribute in a significant way to the model representation are automatically discarded on each iteration.

The step-by-step process finishes when no parameters are discarded. The parameters of this refined model, called step-by-step model, have been adjusted by eliminating those explanatory variables that were not significant through the step-step regression technique. Then, a final iterative process is executed over this model in order to reduce the complexity of the model as well as the linear relationship between the explanatory variables. This process performs small variations in the weights of the parameters, minimizing the margin of deviation of the prices obtained by the resulting model with respect to the prices available in the system. As a result, we obtain the *final model* for each zone class we consider.

Returning to our example, the system has generated a set of models for our zone classes. As three zone classes were identified for the m3.xlarge instance type with Linux, three models have been built, one for each class. The final model proposed for the stable class (*stable zone model*) is detailed in equation 1. As it is shown, for a given time $t$, this model depends on the spot price of the previous two hours, the spot price of the last day (24 hours ago), two days ago (48 hours), a week ago (168 hours, 7 days) and two, three and four weeks ago (336 hours, 504 and 672 hours ago, respectively). This equation also models a pattern that has been identified in the zones belonging to the stable zones class, which is represented by the sine and cosine functions.

$$\begin{aligned} y_t &= \beta_0 + \beta_1 t + \beta_2 y_{t-1} + \beta_3 y_{t-2} + \beta_4 y_{t-24} + \beta_5 y_{t-48} + \beta_6 y_{t-168} \\ &+ \beta_7 y_{t-336} + \beta_8 y_{t-504} + \beta_9 y_{t-672} + \beta_{10} sin(\frac{t2\pi}{7 \times 24}) + \beta_{11} cos(\frac{t2\pi}{7 \times 24}) \end{aligned} \tag{1}$$

The description of the parameters that appear in model equations is detailed in table 4.

| Parameter | Description |
|---|---|
| $y_t$ | Dependent or explanatory variable: spot price in dollars for the instant of time $t$ |
| $\beta_0, \beta_1, .., \beta_n$ | Weights that measure the influence of explanatory variables |
| $sin(\frac{t2\pi}{24})$ | Sinusoidal function of daily period (24 hours), which added to the function $cos(\frac{t2\pi}{24})$ describes the daily trend of the dependent variable |
| $cos(\frac{t2\pi}{24})$ | Cosine function of daily period (24 hours), which added to the function $sin(\frac{t2\pi}{24})$ describes the daily trend of the dependent variable |
| $sin(\frac{t2\pi}{7\times24})$ | Weekly sinusoidal function (168 hours), which added to the function $cos(\frac{t2\pi}{7\times24})$ describes the daily trend of the dependent variable |
| $cos(\frac{t2\pi}{7\times24})$ | Weekly cosine function (168 hours), which added to the function $sin(\frac{t2\pi}{7\times24})$ describes the daily trend of the dependent variable |
| $mean(price_{week-1})$ | Average price of the week before the given time instant |

Table 4: Parameters used in the modeling of the availability zones

Similarly, the *semi-stable zone model* and the *unstable zone model* are presented in equations 2 and 3, respectively. It can be seen that the semi-stable model is similar to the stable one, but in this case the dependency relies only up to a three weeks ago spot price. The patterns found are different, so the representation differs as well. An interesting observation to be made in equation 2 is that the spot price depends on the mean spot price of the previous week, which is represented by the $mean(price_{week-1})$ function. This function stands for the average price of the week before the given time instant.

In the case of the unstable zone model, depicted in equation 3, it can be seen that the model cannot rely on data older than a week ago ($y_{t-168}$). This is because of the dynamic nature of the unstable zones, where spot prices may vary too often. However, even in these dynamic zones there is a repeating behavior, which is included in the model by means of the corresponding trigonometric functions. Note that this behaviour may vary over time, so a data update could add new patterns and dependencies to the equations when they are generated.

$$\begin{aligned} y_t = \beta_0 + \beta_1 t + \beta_2 y_{t-1} + \beta_3 y_{t-2} + \beta_4 y_{t-24} + \beta_5 y_{t-48} + \beta_6 y_{t-168} \\ + \beta_7 y_{t-336} + \beta_8 y_{t-504} + \beta_9 sin(\frac{t2\pi}{24}) + \beta_{10} cos(\frac{t2\pi}{24}) \\ + \beta_{11} sin(\frac{t2\pi}{7\times 24}) + \beta_{12} cos(\frac{t2\pi}{7\times 24}) + \beta_{13}(mean\_price_{week-1}) \end{aligned} \quad (2)$$

$$\begin{aligned} y_t = \beta_0 + \beta_1 t + \beta_2 y_{t-1} + \beta_3 y_{t-2} + \beta_4 y_{t-24} + \beta_5 y_{t-48} + \beta_6 y_{t-168} \\ + \beta_7 sin(\frac{t2\pi}{7\times 24}) + \beta_8 cos(\frac{t2\pi}{7\times 24}) \end{aligned} \quad (3)$$

The construction of the models is a dynamic and flexible process that it is initiated by a set of Python scripts which build a set of $R$ scripts for each identified zone class depending on the information stored in the database and the patterns that were identified during the characterization stage. With all this information, the initial complex model is generated, and then the $R$ software is run in order to execute the step-by-step technique. When it finishes, the final model for each identified class has been defined. Then, a model validation process is carried out in order to ensure the correctness of the models.

### *5.6. Model validation*

The validation of the models generated in the previos step is achieved through a *cross folding validation technique*. Cross validation allows to compare the models and select the one that is more representative. This technique consists of first establishing a training group adjusted to the model and leaving a validation group out of it. Then, the mean errors of this group is calculated with respect to the adjusted model. This validation allows to check if the average error of the predictions is reasonable, and to compare different models to select the one whose representation is better. The importance of this validation lies in establishing a validation group that is not used in the training phase, and that is what allows to compare the prediction capacity of each of the models.

The final models are compared with respect to the complex and the step-by-step models in order to verify that the reduction of the explanatory variables does not affect their predictive capacity. As it is shown in Figure 9, the cross validation is done by dividing the data into two sets: the training set (90%) and the validation set (10%). The data is taken from the pricing database. and a test dataset is defined as well. This test dataset will be used in next section in order to test the correctness of the proposed models. Finally, the models will be used to perform a real spot price prediction over a experimentation setup.

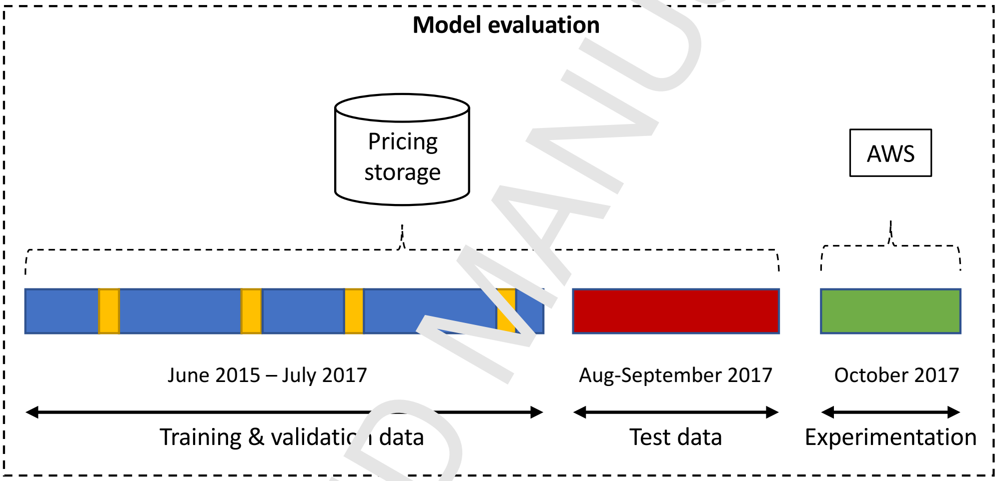


Figure 9: Detail of the evaluation stage.

The phases of model validation and the evaluation of the results are automated, using a set of scripts developed in R. For the running example, the results in Table 5 show that the models obtained previously for the stable and unstable zones adequately represent the desired behavior. However, as one could expect, the accuracy of the prediction for semi-stable zones is notoriously worse. The coefficient of determination, $R^2$, is used to measure the percentage of variation of the response variable, and then this value is adjusted to the predictors (adjusted $R^2$). These values are notoriously lower in the zones classified as semi-stable. This may be due to the fact that this type of area is the most unpredictable: sometimes the price is stable for long periods of time while in others changes are very frequent. In addition, it should be considered that in this types of area there are also high price spikes that drastically alter the auction model.

| $R^2$ | Model | | |
|---|---|---|---|
| | Complex | Step-by-step | Proposed |
| Stable | 0.71220815 | 0.712204854 | 0.71[illegible]333 |
| Semi-stable | 0.37280601 | 0.372775607 | 0.[illegible]7254[illegible]66 |
| Unstable | 0.68867093 | 0.688586516 | 0.68[illegible]307 |

| Adjusted $R^2$ | Model | | |
|---|---|---|---|
| | Complex | Step-by-step | Proposed |
| Stable | 0.71217787 | 0.7121[illegible]8703 | [illegible]71215112 |
| Semi-stable | 0.37264098 | 0.37266[illegible]2 | 0.[illegible]7239758 |
| Unstable | 0.68842504 | 0.6884[illegible]988 | 0.68819183 |

| Validation | Model | | |
|---|---|---|---|
| | Complex | Step-by-step | Proposed |
| Stable | 0.0023967[illegible] | [illegible]327756 | 0.00226816 |
| Semi-stable | 0.00693645 | 0.[illegible]07451680 | 0.00856331 |
| Unstable | 0.1944[illegible] | [illegible]193326882 | 0.19110068 |

Table 5: Results of the validation of the models

The average errors with the cross validation technique show that the more instability is the more complicated is to adjust the model, while the average error increases. The reduction of the least explanatory variables does not produce the average error to increase in a considerable way. Therefore, it can be concluded that the proposed model is adequate. The values of $R^2$ and adjusted $R^2$ for the semi-stable zones are very small. Given the hypothesis that this is due to peaks with a price much higher than the price of the on-demand instances (Amazon sweeps and peaks were previously introduced in the preprocessing stage), a pruning technique with respect to the training data is applied. This tries to remove those data considered as spurious from the training data, that is, those peaks of prices that exceed the price of the on-demand instances. The objective is to eliminate those instants of the auction such that the price exceeds that of the instance on-demand instances, as on-demand instances are more adequate for those cases. In this case, only 3.6% of the data are considered as spurious. This means that applying the pruning process does not reduce the generality of the models of each group of zones. Once the spurious data have been removed the training, construction and validation phases are executed again. In this case, the final models use the same explanatory variables but have different weights. The results of the validation of the resulting models are shown in Table 6. Comparing the results from Tables 5 and 6 it can be seen that all the proposed models improve the corresponding $R^2$ and adjusted $R^2$ values. This means that they adjust the behavior in a better way. In the case of semi-stable zones the improvement is noticeable.

| $R^2$ | Model | | |
|---|---|---|---|
| | Complex | Step-by-step | Proposed |
| Stable | 0.73863411 | 0.73863[illegible]073[illegible] | 0.73860977 |
| Semi-stable | 0.70101314 | 0.70[illegible]12867 | 0.70079427 |
| Unstable | 0.70232884 | 0.702254[illegible]6 | 0.70213269 |

| Adjusted $R^2$ | Model | | |
|---|---|---|---|
| | Complex | Step-by-step | Proposed |
| Stable | 0.73860601 | 0.738606383 | 0.73858916 |
| Semi-stable | **0.7009[illegible]899** | **0.70[illegible]934327** | **0.70072129** |
| Unstable | 0.70204[illegible]98 | 0.702083137 | 0.70198073 |

| Validation | Model | | |
|---|---|---|---|
| | Complex | Step-by-step | Proposed |
| Stable | 0.0[illegible]190142 | 0.001980913 | 0.00238365 |
| Semi-stable | 0.00584118 | 0.005659768 | 0.00617250 |
| Unstable | 0.14625798 | 0.145183440 | 0.14538662 |

Table 6: Results of the validation of models with pruning

## 6. Evaluation and Experimentation

The evaluation stage depicted in Figure 5 in the previous section allows to study the alignment between the predicted values for the test dataset and the real data in such set. In this section, the evaluation phase is described with respect to the results obtained when using the models to predict the behavior for the *test dataset* described in Section 5.6. After that, we conducted an experimentation process to identify the behaviour of the proposed models under real circumstances.

Similarly to the previous phases, the evaluation stage is conducted using the R software with a set of scripts that allow the complete automation of the process.

## 6.1. *Model testing*

As explained in the previous section, the validation over the models with and without applying the pruning technique was carried out. Let us now verify if the use of the pruning process is useful in those zones with high $R^2$ and adjusted $R^2$ values. The results for the data we are managing for m3.xlarge instances are shown in Table 7.

| Validation | Model | | | Test | Model | | |
|---|---|---|---|---|---|---|---|
| | Complex | Step-by-step | Proposed | | Complex | Step-by-step | Proposed |
| Stable | 0.0028 | 0.0026 | 0.0025 | Stable | 0.0024 | 0.0023 | 0.0023 |
| Semi-stable | 0.0073 | 0.0084 | 0.0086 | Semi-stable | 0.0460 | 0.0455 | 0.0456 |
| Unstable | 0.1711 | 0.1640 | 0.1602 | Unstable | 0.2381 | 0.2340 | 0.2362 |

Table 7: Test of the models generated without pruning.

The same validation is carried out after the pruning. The results depicted in Table 8 show that the average errors in the test phase are lower in the models that were trained with pruning. This means that pruning helps in the representation and predictive capacity of the model.

| Validation | Model | | | Test | Model | | |
|---|---|---|---|---|---|---|---|
| | Complex | Step-by-step | Proposed | | Complex | Step-by-step | Proposed |
| Stable | 0.0018 | 0.0019 | 0.0022 | Stable | 0.0020 | 0.0020 | 0.0020 |
| Semi-stable | 0.0041 | 0.0038 | 0.0045 | Semi-stable | 0.0480 | 0.0481 | 0.0462 |
| Unstable | 0.1131 | 0.1086 | 0.1092 | Unstable | 0.2331 | 0.2340 | 0.2356 |

Table 8: Test of the models generated with pruning.

The distribution of the error for each validation was also obtained. The average error is a biased measure, because the price peaks badly detected by the model are those that contribute almost exclusively to the average error. That is, a large number of predictions have a lower and practically disposable error, and the average error is accumulated by a small number of estimates that correspond to misidentified price peaks. The distribution of the error is shown in Table 9. As it can be observed, in the stable and semi-stable zones the error is distributed in a similar way independently of the pruning criteria in the training phase. However, in the unstable areas the trained model after pruning makes better predictions, since in 80% of the predictions it has an absolute error less than 0.07, while the other model makes an absolute error of approximately 0.14 in 80% of the predictions. Again, it seems that the best model is the one on which the pruning was applied previously to the training because its values of $R^2$ and adjusted $R^2$ are more significant and its error is also better distributed among the predictions.

| Test error distribution (without pruning) | Error distribution (proposed model) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 50% | 75% | 80% | 85% | 90% | 95% | 100% |
| Stable | 0.001 | 0.002 | 0.003 | 0.004 | 0.007 | 0.011 | 2.521 |
| Semi-stable | 0.008 | 0.022 | 0.031 | 0.048 | 0.095 | 0.162 | 2.232 |
| Unstable | 0.011 | 0.131 | 0.139 | [illegible] | 0.542 | 1.268 | 2.103 |

| Test error distribution (with pruning) | Error distribution (proposed model) | | | | | | |
|---|---|---|---|---|---|---|---|
| | 50% | 75% | 80% | 85% | 90% | 95% | 100% |
| Stable | 0.001 | 0.002 | 0.003 | 0.004 | 0.005 | 0.011 | 2.521 |
| Semi-stable | 0.008 | 0.028 | 0.035 | 0.041 | 0.098 | 0.154 | 2.231 |
| Unstable | 0.008 | 0.061 | 0.069 | 0.168 | 0.231 | 1.034 | 2.103 |

Table 9: Test error distribution in the model developed without and with pruning, respectively.

The model after applying the pruning was then automatically selected as the best one for spot price prediction. Let us now use the model proposed by the system to study the evolution of spot prices for m3.xlarge Linux spot instances between August and September 2017 in the different zone classes. Figure 10 shows a summary of the spot prices against predictions (prices in black; predictions in blue) for stable zones. A detailed analysis of the data obtained allows to confirm that the model correctly reproduces the evolution of prices. Most of the time the prices are within the interval set in the graph and prices have little dispersion.

In the case of semi-stable zones, it can be observed how a greater number of peaks are produced. However, they seem to follow a predictable pattern. The model for these areas is able to detect these peaks, but does not reach such prices, since they were omitted from the training when pruning. Figure 11 shows the results of the test performed (prices in black; predictions in blue). The model is able to reproduce the evolution of prices.

Finally, in the case of unstable zones, it can be seen that there is a greater

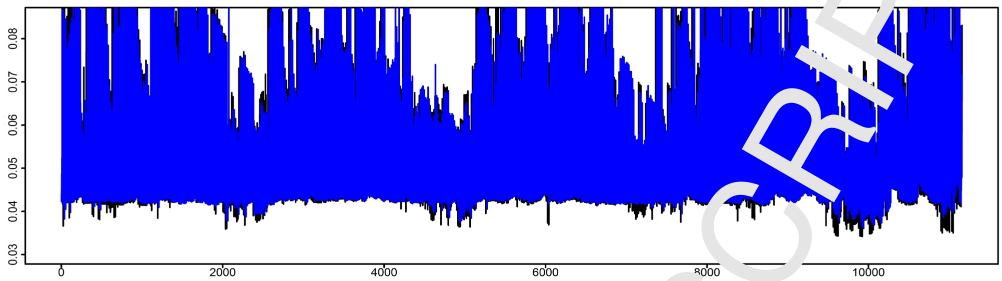


Figure 10: Price predictions in stable zones for the testing period. X-axis: price ($); X-axis: time (hours).

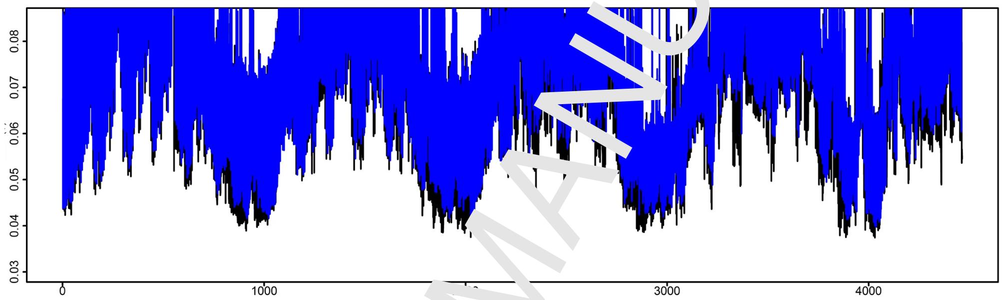


Figure 11: Price predictions in semi-stable zones for the testing period. X-axis: price ($); X-axis: time (hours).

number of peaks that do not follow any type of observable pattern. However, the model for these areas is capable of detecting when most of these peaks occur, but as in the previous case, it does not reach such prices. Figure 12 shows the results of the test performed (prices in black; predictions in blue). As it is shown, the model is able to reproduce the evolution of prices most of the time.

*6.2. Experimentation*

Let us now compare the predictions obtained from the models with real data we have obtained bidding in the EC2 market during October 2017. The objective of this analysis is to check whether the evolution of the predictions corresponds to the real evolution. In addition, we will check the percentage of the time the prices suggested by the model are above the auction price, that is, those instants of time in which the SI would be granted to the end user. This is an important factor, because if the prices estimated by the model

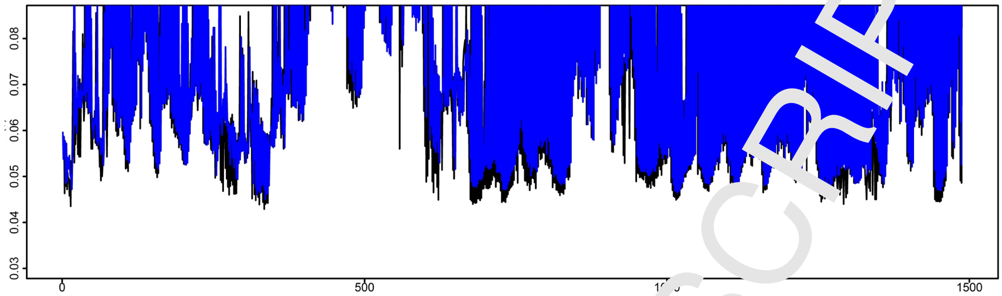


Figure 12: Price predictions in unstable zones for the testing month. X-axis: price ($); X-axis: time (hours).

are lower than the real prices, regardless of whether they represent well the behavior of the auction, the estimated prices would not serve to deploy the instances. Therefore, different correction factors have been evaluated on the estimated price in order to know which factor allows a high allocation success rate without increasing the price excessively.

Figures 13, 14 and 15 show the predictions made in the month of October 2017 for stable, semi-stable and unstable availability zones, respectively. The real evolution of spot prices is shown in black, and the prediction made in blue. In red, yellow and green, the prediction is shown by applying a correction factor of 2.5%, 5% and 10%, respectively. Correction factors increase the predictions in a specified percentage. As it can be seen in the figures, the best predictions are made in the stable areas due to the lower price variability.

The final costs exclusively depend on the value of the auction and not on the spot bid price. Therefore, on the following an evaluation of different correction factors on the estimated price in each zone class is proposed. The objective of this evaluation is to know and select for each zone that correction factor with a trade-off between the probability of allocation of the instance and the bid price, in such a way that it increases the success ratio at the expense of increasing the maximum price the user is willing to pay.

### *6.3. Analysis of the results*

The analysis of the prediction of prices with respect to their real evolution allows to deduce that the models fit reasonably well the evolution curves. Approximately 71% of prediction prices are slightly above the real auction values, which means the user will succeed in obtaining the instances requested. These models will also allow to generate provisioning plans for

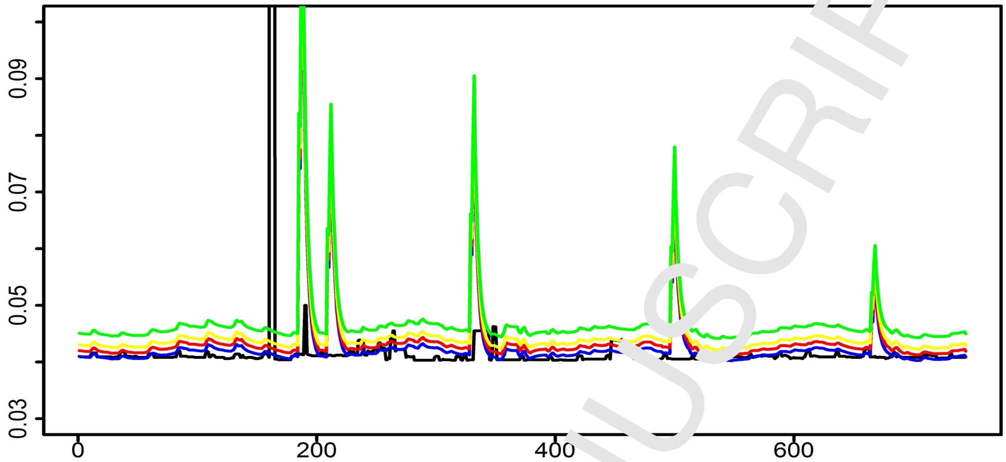


Figure 13: Price prediction for the stable zone eu-central-1b during October 2017. X-axis: price ($); X-axis: time (hours).

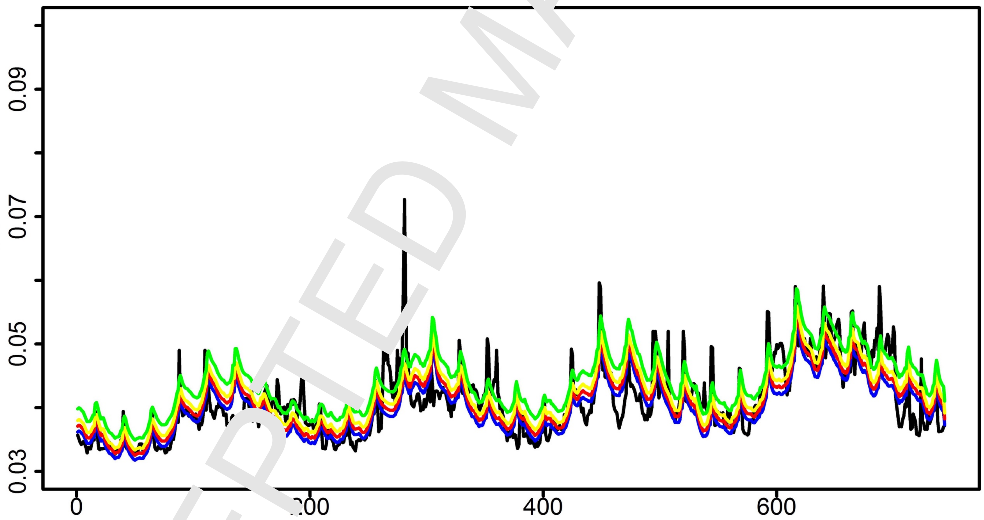


Figure 14: Price prediction for the not-very-stable zone us-west-2b during October 2017. X-axis: price ($); X-axis: time (hours).

executions based on cost and time constraints, checking ranges of hours in which the maximum cost established is above the estimated price. From another point of view, it also allows determining an adjusted price that can guarantee execution during a certain number of hours, as it will be depicted

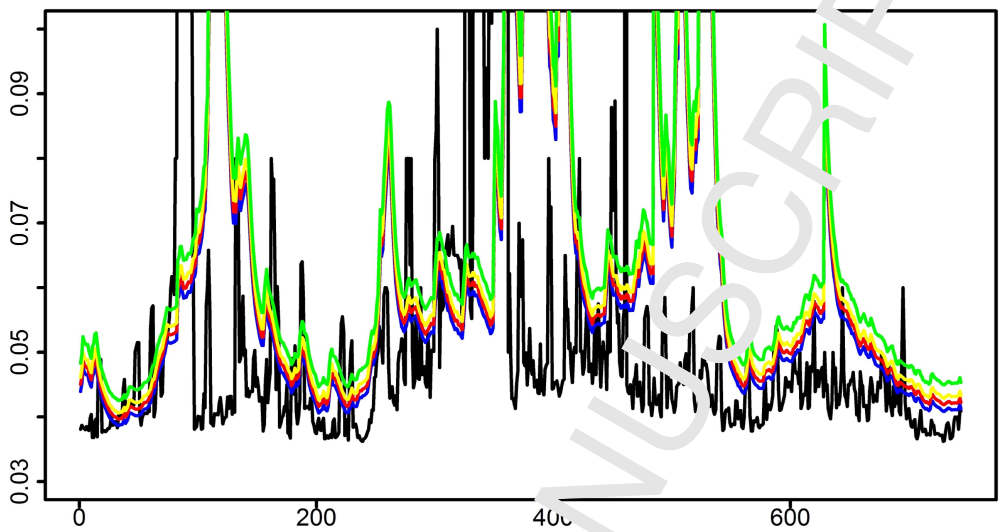


Figure 15: Price prediction for the unstable zone us-east-1e during October 2017. X-axis: price ($); X-axis: time (hours).

in next section when generating provisioning plans.

A *correction factor* has been added to the model. This factor proportionally increases the estimated price in order to also increase the allocation success factor and to extend it as much as possible over time. In return, the total execution costs are also increased, since the instance is obtained at a higher cost in those moments of time when the auction price increases. Again, the previous predictions have been made applying these correction factors to verify the success factor obtained in each type of zone, and to be able to establish one that is a good trade-off between the total execution cost and the probability of expulsion.

According to the data shown in Table 10, stable zones can reach a successful prediction for all availability zones. Depending on the zone, it is necessary to apply higher correction factors in order to obtain similar success rates. This is because these areas have a high number of peaks in prices. The greater the instability of the area is, the greater price increment must be applied to obtain success rates close to 90%. However, because the average price of SI is approximately between 15% and 30% of the equivalent on-demand resources depending on the availability zones, substantial savings close to 60% could be obtained.

| Average up-time | Correction factor (%) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Zone class | 0.0% | 2.5% | 5.0% | 10.0% | 15.0% | 20.0% | 25.0% | 30.0% | 45.0% | 50.0% |
| | Stable | 60.0% | **92.1%** | **96.6%** | 98.4% | 98.9% | 99.0% | 99.1% | 99.2% | 99.2% | **99.3%** |
| | Semi-stable | 62.5% | 71.2% | 77.6% | 85.5% | **89.6%** | **91.6%** | 92.8% | 93.[illegible]% | 94.4% | **94.8%** |
| | Unstable | 70.7% | 75.3% | 78.8% | **83.2%** | **85.5%** | 86.8% | 88.1% | 8[illegible].0% | 89.6% | **90.2%** |
| | **All classes** | 63.6% | 80.1% | 85.0% | 89.8% | 92.0% | 93.2% | 94.0% | 94.6% | 95.0% | 95.3% |

Table 10: Correction factors applied to the different SI zones.

Figure 16 summarizes the influence of correction factors on availability zones. In the case of stable zones, applying a correction factor above 10% of the estimated price does not improve in excess the probability that the spot bid will succeed. However, with a relatively low correction factor of 3%, the probability of eviction can be reduced by up to 33%. Observing the semi-stable and unstable zones, the growth curve of the up-time with respect to the price increase is not so drastic, so correction factors around 15% and 20% respectively would be a good compromise between the price increase and the reduction of the probability of evictions.

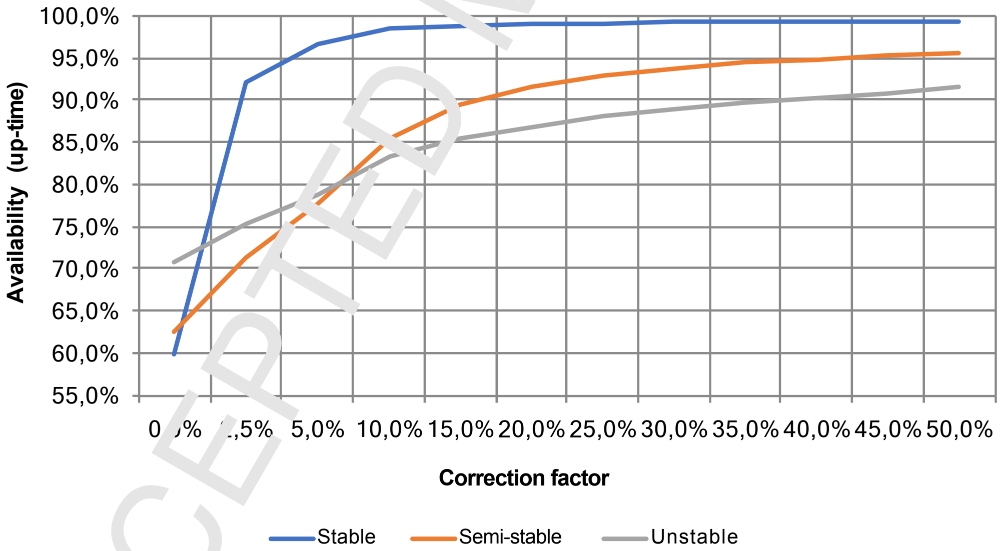


Figure 16: Increase in availability compared to the correction factor by zone class.

Finally, Figure 17 shows the behaviors described for different AWS SI availability zones. The zones classified as stable (eu-central-1b, eu-west-1a and eu-west-1b) have a similar growth curve, reaching up-time growth rates of over 35%. The semi-stable zones (us-west-2b, us-west-2c and us-east-1a) have a growth curve similar to that observed for this type of zones in Figure 16. However, the unstable zones (eu-central-1a and us-east-1e), despite having high availability without applying any correction factor, have a very limited growth curve, around 70%. For this reason, it is not recommended to apply correction factors greater than 20% since the gains in terms of availability are minimal.

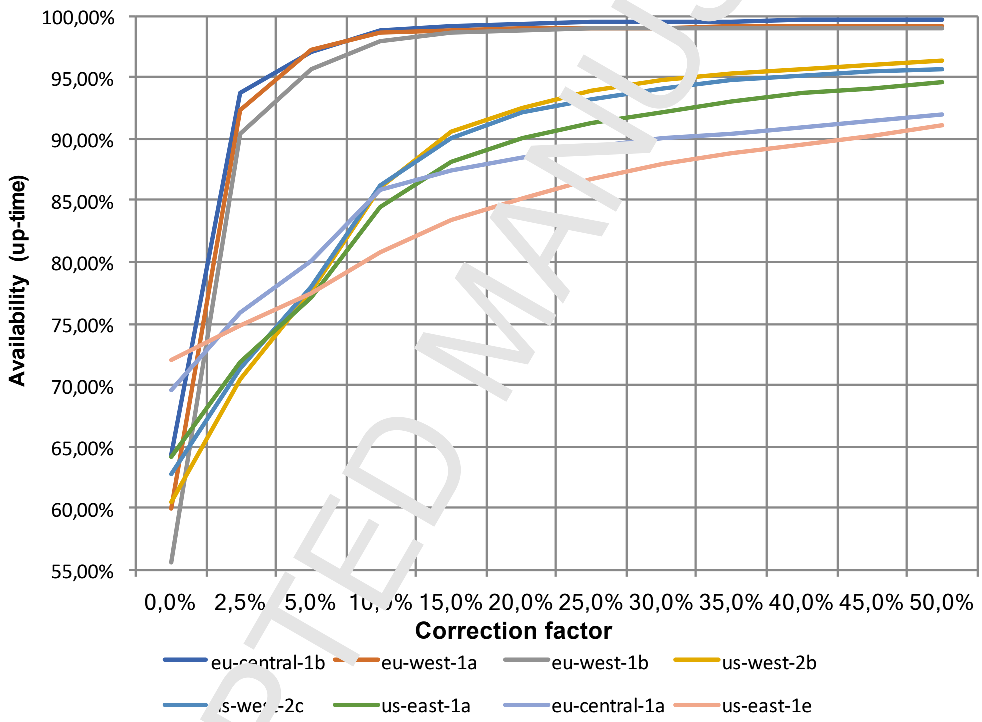


Figure 17: Increase in availability compared to the price increase in different SI zones.

These corrections factors have been incorporated to the proposed models in order to fit better the prediction results. Given the fact that the final price depends on the real final spot price, the correction factors can be included on all models for every class in the EC2 pricing schema. This improves the behavior of the results and still allows to get an excellent tradeoff between spot instance use and cost savings.

## 7. Automatic generation of provisioning plans in Amazon EC2

The prediction models can be applied to the synthesis of provisioning plans in Amazon EC2. A provisioning plan consists of a list of feasible time instants at which a specific instance type can be requested. Given a deadline, an EC2 region or a specific zone, an instance type, the operating system, the number of execution hours and the maximum price per hour, the EC2 SI Provisioning Maker component depicted in Figure 1 uses an internal simulator to generate all feasible hours at which a bid could be placed in the EC2 SI. A feasible hour means that the simulation process estimates that the bid will succeed and, therefore, we would be able to create a SI of the requested type without being preempted.

Given the deadline and cost constraints, the system provides the user with a complete overview of the suitability of using SIs for the deployment of an experiment. We have used this system to construct and execute real provisioning plans in a healthcare-related environment. Conducting immune response studies requires processing patient samples through different techniques and methods [40]. The study of the immune response, among other utilities, serves to see the patient's response to medical treatments. We present a use case in which we analyzed the immune response to select a surgical treatment in patients with cancer.

Figure 18 shows the process followed to study the immune response for patients that meet a given set of inclusion criteria. First, a sample is extracted from the patient. Samples are anonymized and processed through a flow cytometer with different parameters, whose output is recorded in several files. These files are then processed using a specific software with the aim of obtaining information of interest related to a variety of aspects such as the type of cells, their activity or the percentage of death cells, for instance. This information has a very high level of detail. In order to handle it, a process selecting the relevant data must be previously executed. For that, a pattern recognition process, based on machine learning techniques, is applied. The final results are used for the study of the patient's immune response.

The execution of the previous process is very expensive in computational terms. The most complex process is the one that performs pattern recognition using machine learning algorithms. Figure 19 depicts an example of the process that it is achieved at this stage.

Every week, over 50 samples with multiple parameters must be processed, for which the Amazon EC2 computing resources are used together with a

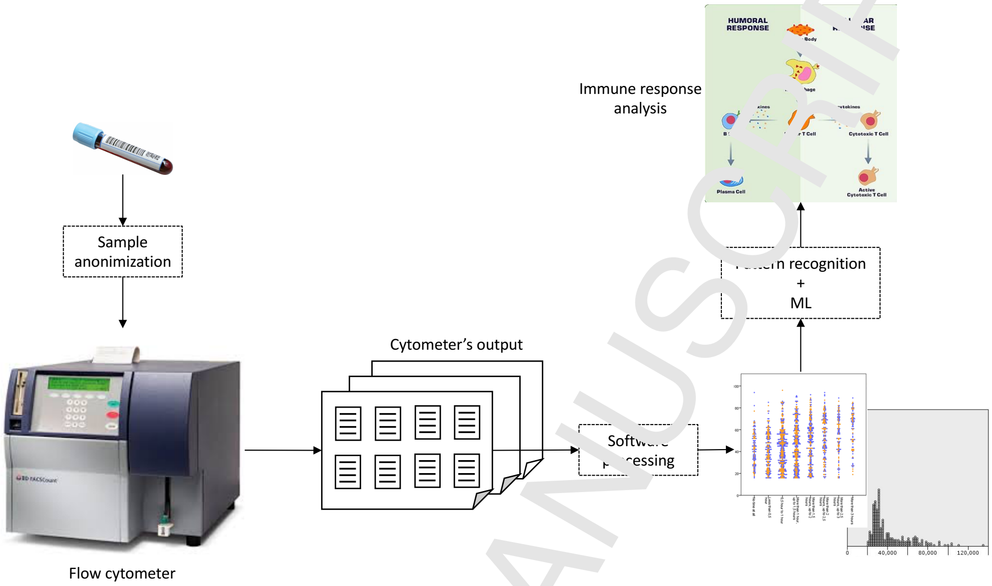


Figure 18: Workflow for the analysis of the immune response.

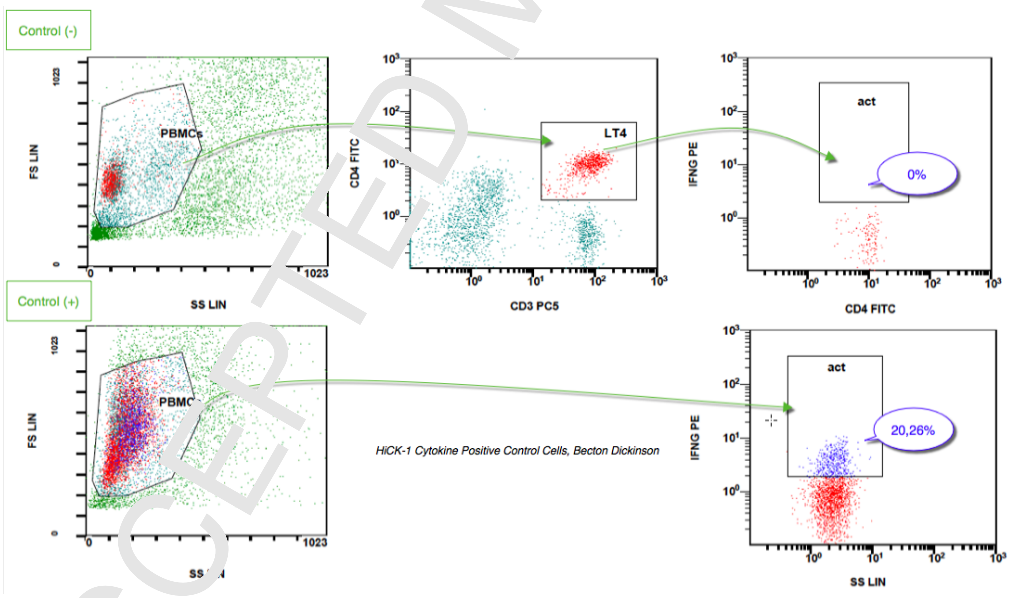


Figure 19: Automatic selection of data set suitable for immune response analysis.

storage solution based on Amazon S3 [41]. The computation of the samples according to the process shown in Figure 18 is done deploying between 15 and 20 m3.xlarge instances running Linux (Ubuntu) in parallel. The analysis of the software requirements justified the use of this type of instances to run the processes. Each machine processes several samples through a specific workflow, taking the sample information from S3 and storing the final results back there. Processing a sample takes between 11 and 14 hours. Considering the empirically observed overhead of process configuration and setup, data storage and retrieving, processing all samples requires an overall execution time in the range between 565 and 715 hours.

The weekly execution of the previous process using on-demand instances in the EC2 service represented a cost of between $152 and $225. In September 2017, the application of SI using the approach presented in this paper was proposed as a feasible alternative. The SI service allows launching the experiments with the same characteristics of hardware, operating system and integration environment (EC2 and S3) than the previous schema, facilitating the migration, and offers the possibility of obtaining lower costs and important savings.

The *Provisioning Maker* component (depicted in Figure 4) was used to generate resource provisioning plans for the same type of instance originally used (m3.xlarge machines with the GNU/Linux operating system). This component's input was the configuration of the instances required, the amount of computation hours required, the deadline (the analysis of a sample cannot be delayed more than two days - 48 hours- since it is received) and, finally, the costs bound. The provisioning maker component generated a price prediction per hour for each EC2 availability zones using the models detailed in Section 5. Figure 20 depicts the whole process.

The correction factors described in Section 6.3 were included in the models proposed by the system. Expulsions had to be avoided, since neither recovery nor checkpointing mechanisms were available in the software used to conduct the use case. The analyst studies the provision plans proposed by the component and selects the one to be executed. This process is normally cost-driven, as the analyst chooses the plan that provides one of the higher success percentages with the lower price. The company has developed an application that processes the supplied plans, chooses one among them and places the corresponding bids in the spot market. If the bids are the winners, the requested instances are launched and the process execution starts. Figure 21 shows a screenshot of the selection of a provisioning plan by the

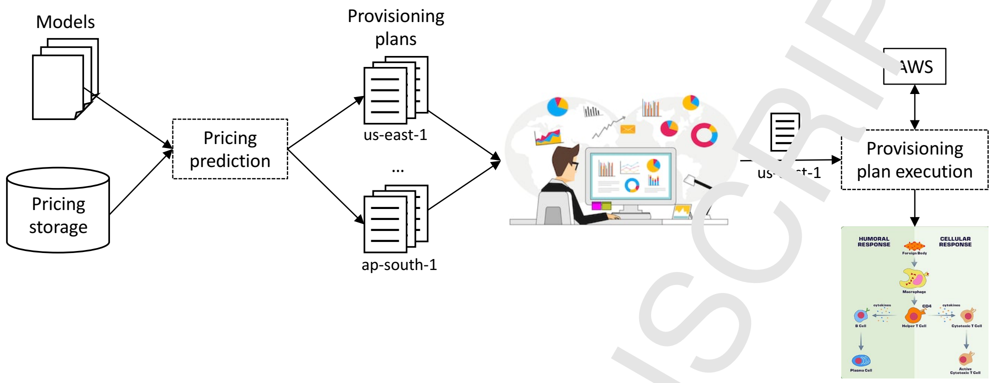


Figure 20: *EC2 SI Provisioning Maker* process.

problem analyst using the proposed framework.

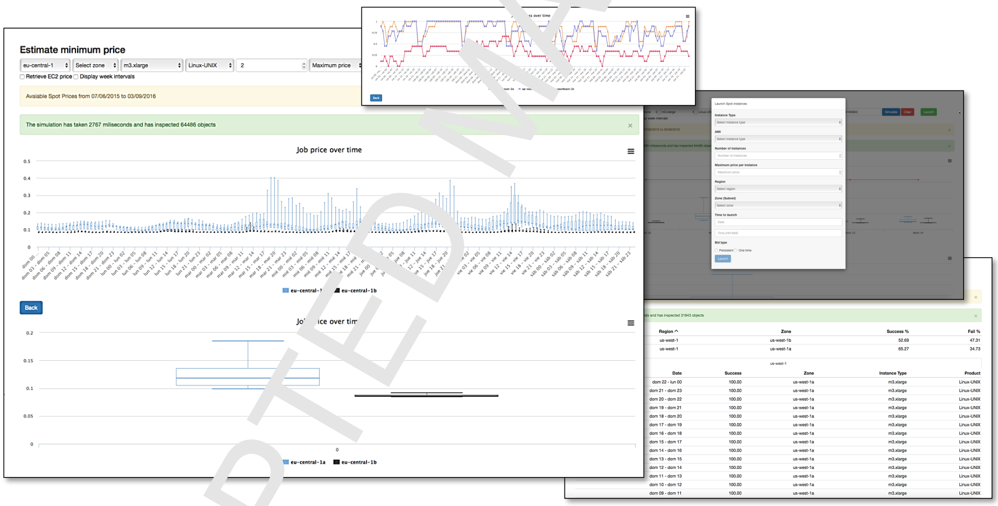


Figure 21: Screenshot of the tool for selecting a provisioning plan.

Table 11 details the costs for six weeks since December 2017 until January 2018. In weeks 1, 2 and 5 the processes have been successfully executed using the generated provisioning plans on semi-stable zones (us-east-1a, us-west-2c and us-east-1a, respectively), with an overall total cost of $111.40 compared to $507.26 that the execution would have cost using on-demand instances. This represents a saving of 78% using the SI with respect to the on-demand model. Weeks 3 and 4 have been executed on stable zones (eu-central-1b and eu-west-1a, respectively), achieving a saving of 80% ($70.52 for SI with respect to $358.57 for on-demand). Finally, in week 6 the execution was carried out on an unstable zone (eu-central-1a), with a cost of $41.70 compared to the $216.41 of the on-demand model, which represents a saving of 80.7%.

| Week | Exec. time (h) | SI cost ($) | SI price ($) | On-demand cost ($) | On-demand price ($) | Savings (%) |
|---|---|---|---|---|---|---|
| Week 1 | 581 | 35.03 | 0.0603 | 154.55 | 0.2660 | 77.3 |
| Week 2 | 623 | 35.95 | 0.0577 | 165.72 | 0.2660 | 78.3 |
| Week 3 | 610 | 35.99 | 0.0590 | 192.15 | 0.3150 | 81.3 |
| Week 4 | 568 | 34.53 | 0.0608 | 166.42 | 0.2930 | 79.2 |
| Week 5 | 703 | 40.42 | 0.0575 | 187.00 | 0.2660 | 78.4 |
| Week 6 | 687 | 41.70 | 0.0607 | 216.41 | 0.3150 | 80.7 |

Table 11: Costs for the execution of the processes during six weeks.

An average saving of 79.3% has been obtained using the provisioning plans with EC2 SI instead of the on-demand instances, with a total cost of $223.63 compared to $1,082.24, respectively. Therefore, it seems evident that the generation and use of the provisioning plans with the proposed models are very useful and effective in actual practice, allowing a very significant saving in the total costs of execution and without altering the configuration or the requirements of the problem to solve.

## 8. The new Amazon EC2 Spot pricing model

Amazon has recently introduced a series of changes in the spot pricing model that affect prices, access and suspension of the instance [42, 43]. On the one hand, Amazon announced that the SI price model was going to move to one in which prices adjust more gradually. This change is oriented towards long-term trends. It is expected that it will still be possible to obtain savings of between 70 and 90% of the price of the on-demand instance [42].

On the other hand, Amazon announced a *streamlined access model* for Spot Instances. This model simplifies the use of SI by the user, allowing to select this type of instances in the same way that an on-deman instance is selected. The request can include the maximum price that is willing to pay per hour per instance, with the default price being the on-demand instance one. Other limitations can also be included, such as the type of instance and the availability zone. If the maximum price is higher than the current spot price for the specified instance and there is available capacity, the request is attended immediately and a SI is launched until the user finishes it or EC2 claims it for an on-demand use. The user will pay the spot price that's in effect for the current hour for the instances that were launched. This change tries to facilitate and promote the use of SI, avoiding having to know the spot markets, the bidding mechanism and the interaction through an asynchronous API, since the new model gives control immediately over the instance in case the request can be served.

An important change is the concept of eviction. In the new model, the *interruption* of a SI is due to factors based on price (the spot price is higher than its maximum price), capacity (there are not enough EC2 instances not used to satisfy the demand of SIs) and restrictions (a restriction may be included in the SI request as a launch group or a group of availability zones). Therefore, the fluctuation of the prices in the spot market are not longer used. Note, however, that the factors that condition the interruption of a SI are still internal to the provider and cannot be known without a deep and detailed understanding and analysis of the AWS infrastructure.

Finally, the option to *hibernate* an instance has been added, although some requirements detailed below must be fulfilled. Now it is possible (for those instances that meet the requirements) to save the memory status of an instance when the instances are interrupted (or *reclaimed* in the new terminology, since they are claimed to be used as on-demand instances), and recover the previous status when capacity is available again. The private IP addresses and the elastic IP of the instance are also maintained during the stop-start cycle.

However, the new model still allows for a more detailed control over the SI mechanism. The maximum price that the user want to spend when the request is made can be specified, and the jobs and applications that use the RequestSpotInstances or RequestSpotFleet API services continue to function correctly. It is also still possible to establish a configuration when requesting and deploying instances in order to diversify the placement of SI across the most cost-effective pools. Therefore, existing research can be adapted to the new model. Although the concepts of spot market and the bidding mechanism disappear, the spot prices are more predictable in the new model. Prices are updated less frequently and are determined by supply and demand for Amazon EC2 spare capacity, not bid prices. It still makes sense, therefore, to be able to predict the price of SIs to anticipate and be able to provide solutions that use SIs and offer significant savings. Although the complexity of the analysis of the spot market and its fluctuations is now lower, the changes in the prices of the instances and their availability continue to respond to internal criteria of the provider (the capacity of the infrastructure or the number of instances available are not public). Therefore, adapting existing approaches to the new pricing model would allow to predict the maximum price to start up a SI with a specific configuration and the best savings.

We have conducted a preliminar analysis with the data from February to April 2018 (which corresponds to the new model of SI) in order to evaluate how the spot prices fluctuate with the new model. We analyzed the available data in each region with all availability zones, instance types and operating systems. To characterize the different classes that can be found, the frequency and the deviation in the price changes (*variations*) were considered. Table 12 shows the different classes that we can set up combining these two measures.

| | Class 1 | Class 2 | Class 3 | Class 4 |
|---|---|---|---|---|
| Frequency of variation | High ( >150 ) | High ( >150 ) | Low ( <30 ) | Low ( <30 ) |
| Deviation of price | Low ( <10% ) | High ( >40% ) | Low ( <10% ) | High ( >40 %) |

Table 12: Region/zone classes using frequency of variation and deviation of price.

With the classification proposed in Table 12 we conducted an analysis of the data collected for the different regions. The results are shown in Table 13. Compared with the previous spot market, it is clear that now prices fluctuate much less, but there are still some differences that are noticeable. The results depicted in Table 13 allow us to observe that there are three well differentiated classes. Class 2 only includes two availability zones, so a more detailed analysis would allow them to be included in one of the previous classes. Therefore, it is appreciated that, although spot prices vary less, their prediction is not so obvious in all cases, and a detailed analysis of each availability zone may be required if precise results are to be obtained. The differences among the classes may justify the generation of different prediction models similarly to how it was conducted in this paper.

| | Class 1 | Class 2 | Class 3 | Class 4 | Total |
|---|---|---|---|---|---|
| | >Freq <Dev | >Freq >Dev | <Freq <Dev | <Freq >Dev | Total |
| ap_northeast_1 | 24 | 0 | 32 | 0 | 56 |
| ap_northeast_2 | 0 | 0 | 24 | 2 | 26 |
| ap_south_1 | 4 | 0 | 16 | 0 | 20 |
| ap_southeast_1 | 22 | 2 | 26 | 4 | 54 |
| ap_southeast_2 | 14 | 0 | 28 | 0 | 42 |
| ca_central_1 | 2 | 0 | 14 | 0 | 16 |
| eu_central_1 | 20 | 0 | 20 | 0 | 40 |
| eu_west_1 | 56 | 0 | 28 | 0 | 84 |
| eu_west_2 | 10 | 0 | 16 | 2 | 28 |
| eu_west_3 | 0 | 0 | 10 | 4 | 14 |
| sa_east_1 | 6 | 0 | 8 | 2 | 16 |
| us_east_1 | 222 | 0 | 32 | 0 | 254 |
| us_east_2 | 66 | 0 | 32 | 0 | 98 |
| us_west_1 | 56 | 0 | 10 | 0 | 66 |
| us_west_2 | 102 | 0 | 24 | 0 | 126 |
| Total | 604 | 2 | 320 | 14 | 940 |

Table 13: Analysis of price variations in the AWS regions with the new model.

In general terms, the framework proposed in Section 4 fits the new Amazon model, as well as the proposed approach does. The prices of the instances vary in each region, which continues to allow a detailed analysis that motivates the realization of a clustering to characterize the different regions. From this analysis, the process would be similar to the one detailed in this work. Predictive models would be generated for each of the classes obtained from the clustering process. These models would be validated and could then be used to make a prediction of the maximum price the user is willing to pay for an instance.

From the predictive models, it is possible to generate a provisioning plan that minimizes the cost of the required infrastructure, combining the maximum prices according to the predictive models with the different zones of availability or regions when establishing the configuration of the request against the Amazon API. The mechanism for generating provisioning plans that has been described in Section 7 could be adapted.

Regarding hibernation, it is important to highlight the requirements that the instance must have [44]. First, for a spot instance request, the type must be *persistent*, not *one-time*. The state of the memory is flushed to the root EBS volume of the instance, so the root volume must be an EBS volume, not an instance store volume, and must be large enough to store the memory (RAM) of the instance during hibernation. In addition, only the following instances are compatible with the spot hibernation mode: C3, C4, C5, M4, M5, R3 and R4, with less than 100 GB of memory. Something similar happens with the operating system, since only the following operating systems are compatible: Amazon Linux 2, Amazon Linux AMI, Ubuntu with an Ubuntu kernel set for AWS (linux-aws) after 4.4.0-1041 and Windows Server 2008 R2 or later.

These requirements limit the level of applicability of the spot hibernation mechanism. In many cases it will still be necessary to provide a checkpointing model that allows not to lose the information processed in case a SI is interrupted. Furthermore, even if the requirements are met and the state of the memory is stored, the conditions of the initial request must be fulfilled in order for the instance to be restarted. In the experiment described in Section 7, a requirement was that the analysis of a sample can not be delayed more than two days -48 hours- since it is received. Taking advantage of the possibility of obtaining better prices in exchange for a possible interruption in which spot hibernation is used (so the above requirements have to be fulfilled) requires adding the condition that the instances must resume their execution again before the deadline expires, which adds complexity to the problem of the generation of provisioning plans and should be evaluated in detail.

## 9. Conclusions

The use of new models for the hiring of computing instances, such as Amazon EC2 Spot Instances and Google Cloud Preemptible Virtual Machine, can drastically reduce the cost of system deployment and execution in cloud infrastructures. However, the inherent low reliability of this class of resources suggests the need for a system that, analyzing the historical evolution of prices and resource preemption events, could generate provisioning plans with an adequate trade-off between the cost and the probability of suffering expulsions so as to be able to satisfy some deadline requirements as well as cost constraints. In the case of the Amazon SI service, the provisioning system should be able to propose a good bidding strategy in order to participate in the auction process for the resources, ensuring some quality aspects to be fit (deadlines and cost bounds).

In this paper, a framework for the analysis of Amazon EC2 Spot Instances has been presented. This framework allows an automated process of data collection and processing of the available spot prices. Based on these data, an analysis is carried out to classify the availability zones, generating a series of well-differentiated zones classes through a clustering process. For each zone class, a predictive model of the spot price is generated. The generation of a predictive model for each zone class instead of a general one allows obtaining more precise results for each availability zone. Moreover, these models are updated each time new data is available.

These predictive models have also been used to define provisioning plans. The paper describes a real experiment that has used zones with quite different behaviors, and which demonstrates that the proposed method can generate important cost savings (above 79%) when compared to the use of on-demand instances for the same tasks.

Currently, the use of alternative analysis techniques such as Markov chains or Machine learning is being considered in order to improve the accuracy of the predictions. Our current and future work is going to concentrate on the adaption of the presented methodology to the changes and the new Amazon EC2 Spot pricing model introduced recently. As it was discussed in Section 8, the new model still allows for a more detailed control over the SI mechanism. The adaption of the approach presented in this work as well as the proposed framework will allow us to deal with the generation of provisioning plans that benefit from the use of SIs over the new model.

**Acknowledgements**

This work has been supported by the TIN2017-84796-C2-2-R project, granted by the Spanish Ministry of Economy, Industry and Competitiveness, and the DisCo-T21-17R project, granted by the Aragonese Department of Innovation, Research and Universities.

**References**

[1] Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I., Cloud computing and emerging IT platforms: vision, hype, and reality for

delivering IT services as the 5th utility, Future Generation Computer Systems 25 (6) (2009) 599–616.

[2] Warneke, D., Kao, O., Exploiting dynamic resource allocation for efficient parallel data processing in the cloud, IEEE Transactions on Parallel and Distributed Systems (2011) 985–997.

[3] de Assuncao, M.D., di Costanzo, A., Buyya, R., Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters, 18th ACM international symposium on High performance distributed computing (2009) 141–150.

[4] Ben-Yehuda, O., Schuster, A., Sharov, A., Silberstein, M., Iosup, A., Expert: Pareto-efficient task replication on grids and a cloud, 26th IEEE International Parallel and Distributed Processing Symposium (2012) 167–178.

[5] Folling, A., Hofmann, M., Improving scheduling performance using a q-learning-based leasing policy for clouds, Euro-Par (2012) 337–349.

[6] Google Compute Engine, https://cloud.google.com/compute/, [Online; accessed in April 2018].

[7] Amazon Web Services EC2 - Simple Cloud Hosting, https://aws.amazon.com/ec2/, [Online; accessed in April 2018].

[8] Amazon Spot Instances, https://aws.amazon.com/ec2/spot/, [Online; accessed in April 2018].

[9] CHARGE Project, https://www.dnanexus.com/usecases-charge, [Online; accessed in April 2018].

[10] AWS Case Study: Netflix, https://aws.amazon.com/es/solutions/case-studies/netflix/, [Online; accessed in April 2018].

[11] Spot Instance Pricing History, http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/usingspotinstanceshistory.html, [Online; accessed in April 2018].

[12] Spot Bid Advisor, https://aws.amazon.com/ec2/spot/bid-advisor/, [Online; accessed in April 2018].

[13] Amazon Spot Block Instances, https://aws.amazon.com/es/blogs/aws/new-ec2-spot-blocks-for-defined-duration-workloads/, [Online; accessed in April 2018].

[14] Yi, S., Kondo, D., Andrzejak, A., Monetary cost-aware checkpointing and migration on amazon cloud spot instances, IEEE Transactions on Services Computing (2011) 236–243.

[15] Mattess, M., Vecchiola, C., Buyya, R., Managing peak loads by leasing cloud infrastructure services from a spot market, 12th IEEE International Conference on High Performance Computing and Communications (2010) 180–188.

[16] Wee, S., Debunking real-time pricing in cloud computing, 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing – CCGrid (2011) 585–590.

[17] Ben-Yehuda, O.A., Ben-Yehuda, M., Schuster, A., Tsafrir, D., Deconstructing Amazon EC2 Spot instance pricing, 3rd IEEE International Conference on Cloud Computing Technology and Science (2011) 304–311.

[18] A. K. Mishra, D. K. Yadav, Analysis and Prediction of Amazon EC2 Spot Instance Prices, International Journal of Applied Engineering Research 12 (21) (2017) 11205–11212.

[19] O'Loughlin, J., Gillam, L., Performance evaluation for cost-efficient public infrastructure cloud use, Economics of Grids, Clouds, Systems, and Services - 11th International Conference (GECON 2014) (2014) 133–145.

[20] Javadi, B., Thulasiram, R.K., Buyya, R., Characterizing spot price dynamics in public cloud environments, Journal of Future Generation Computer Systems 29 (4) (2013) 988–999.

[21] Javadi, B., Thulasiramy, R.K., Buyya, R., Statistical Modeling of Spot Instance Prices in Public Cloud Environments, Fourth IEEE International Conference on Utility and Cloud Computing (UCC) (2011) 219–228.

[22] Z. Cai, X. Li, R. Ruiz, Q. Li, Price forecasting for spot instances in cloud computing, Future Gener. Comput. Syst. 79 (P1) (2018) 38–53.

[23] V. Khandelwal, A. Chaturvedi, C. P. Gupta, Amazon EC2 Spot Price Prediction using Regression Random Forests, IEEE Transactions on Cloud Computing (2017) 1.

[24] M. Baughman, C. Haas, R. Wolski, I. Foster, K. Chard, Predicting Amazon Spot Prices with LSTM Networks, in: Proceedings of the 9th Workshop on Scientific Cloud Computing, ScienceCloud 18, ACM, 2018, pp. 1:1–1:7.

[25] S. G. Domanal, G. R. M. Reddy, An efficient cost optimized scheduling for spot instances in heterogeneous cloud environment, Future Generation Computer Systems 84 (2018) 11 – 21.

[26] Andrzejak, A., Kondo, D., Yi, S., Decision model for cloud computing under sla constraints, 18th IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (2010) 257–266.

[27] Chohan, N., Castillo, C., Spreitzer, M., et al., See spot run: using spot instances for mapreduce workflows, 2nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud10 (2010) 7–13.

[28] Tang, S., Yuan, J., Li, X.Y., Towards Optimal Bidding Strategy for Amazon EC2 Cloud Spot Instance, IEEE 5th International Conference on Cloud Computing – CLOUD '12 (2012) 91–98.

[29] Zafer, M., Yang Song, Kang-Won Lee, Optimal Bids for Spot VMs in a Cloud for Deadline Constrained Jobs, IEEE 5th International Conference on Cloud Computing – CLOUD '12 (2012) 75–82.

[30] Chaisiri, S., Kaewpuang, R., Lee, B.S., Niyato, D., Cost minimization for provisioning virtual servers in Amazon elastic compute cloud, 19th IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems – MASCOTS (2011) 85–95.

[31] Zhang, Q., Gurses, E., Boutaba, R., Xiao, J., Dynamic resource allocation for spot markets in clouds, 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services – Hot-ICE11 (2011) 1–6.

[32] Rahman, M.R., Lu, Y., Gupta, I., Risk aware resource allocation for clouds, technical report 2011-07-11, University of Illinois at Urbana-Champaign.

[33] Amazon Elastic Compute Cloud API, https://docs.aws.amazon.com/AWSEC2/latest/APIReference/, [Online; accessed in April 2018].

[34] J. Fabra, S. Hernández, P. Álvarez, J. Ezpeleta, Á. Recuenco, A. Martínez, A history-based model for provisioning EC2 spot instances with cost constraints, in: J. Á. Bañares, K. Tserpes, J. Altmann (Eds.), 13th International Conference on Economics of Grids, Clouds, Systems, and Services - GECON 2016, Vol. 10382 of Lecture Notes in Computer Science, Springer, 2016, pp. 208–222.

[35] Amazon Relational Database Service (RDS), https://aws.amazon.com/rds/, [Online; accessed in April 2018].

[36] P. Esling, C. Agon, Time-series data mining, ACM Comput. Surv. 45 (1) (2012) 12:1–12:34.

[37] D. Michael, J. Houchin, Automatic eeg analysis: A segmentation procedure based on the autocorrelation function, Electroencephalography and Clinical Neurophysiology 46 (2) (1979) 232 – 235.

[38] M. A. Newell, D. Cook, H. Hofmann, J.-L. Jannink, An algorithm for deciding the number of clusters and validation using simulated data with application to exploring crop population structure, The Annals of Applied Statistics 7 (4) (2013) 1898–1916.

[39] H. Zhao, J. Liang, H. Hu, Clustering Validity Based on the Improved Hubert Gamma Statistic and the Separation of Clusters, in: First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06), Vol. 2, 2006, pp. 539–543.

[40] C. Janeway, P. Travers, Immunobiology: The Immune System in Health and Disease, Current Biology Limited, 1994.

[41] Amazon S3 — Simple Cloud Storage Service, https://aws.amazon.com/s3/, [Online; accessed in April 2018].

[42] Barr, J., Amazon EC2 Update – Streamlined Access to Spot Capacity, Smooth Price Changes, Instance Hibernation, https://aws.amazon.com/es/blogs/aws/amazon-ec2-update-streamlined-access-to-spot-capacity-smooth-price-changes-instance-hibernation/, [Online; accessed in September 2018].

[43] Amazon AWS re:Invent 2017, https://reinvent.awsevents.com, [Online; accessed in September 2018].

[44] Spot Instance Interruptions, https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-interruptions.html, [Online; accessed in September 2018].

**Javier Fabra** received his Ph.D. in computer science from the University of Zaragoza, Spain, in 2010. He holds an Associate Professor position in the Department of Computer Science and Systems Engineering at the University of Zaragoza, Spain, since 2008. His main research areas focus on service-oriented computing and cloud architectures, semantic and scientific computing, and interoperability issues in cluster, grid and cloud scenarios by means of the application of high-level Petri nets.

**Joaquín Ezpeleta** received the M.S. degree in Mathematics and the Ph.D. degree in Computer Science from the University of Zaragoza, Spain. He is a professor of the Dept. of Computer Science and Systems Engineering of the University of Zaragoza, where he conducts lectures on formal methods for sequential and concurrent programming as well as service-oriented architectures. He has worked as a researcher at the Laboratory of Methods and Architectures for Information Systems (MASI), at the University of Paris-6, and the Digital Enterprise Research Institute (DERI), at the National University of Ireland in Galway. His research has focused on the problem of modelling, analysis, and control synthesis for concurrent systems as well as the application of formal techniques to help in the development of correct distributed systems based on Internet and cloud technologies, as well on the parallel processing of data and computing intensive computing problems. He has co-authored more than 80 research papers and participated in numerous program comities of international conferences, being also a reviewer of prestigious research journals.

**Pedro Álvarez** received the Ph.D. degree in computer science engineering from the University of Zaragoza, Zaragoza, Spain, in 2004. He works as Lecture Professor at this University, since 2000. His current research interests focus on two main aspects. First, on integration problems of network-based system (cloud-based and service-based systems, mainly) and the use of novel techniques and methodologies for solving them. And, secondly, on the application of formal analysis techniques to mine event logs and databases (in the domain of e-commerce, e-learning, cybersecurity, or health, for example).

**Journal reference: FGCS Special Issue on the Economics of Computing Services**

**Title: Reducing the Price of Resource Provisioning using EC2 Spot Instances with Prediction Models**

**Authors: Javier Fabra, Joaquín Ezpeleta, Pedro Álvarez**

**Highlights**

- A user-oriented framework that allows to provide history-based models to predict Amazon Spot Instances (SI) prices for the different availability zones is presented.
- The proposed solution has considered and analyzed the SI market during a long-term period.
- All availability zones and regions of Amazon SI have been analyzed and classified, providing the most suitable model for price prediction in each case.
- Provisioning plans are generated according to these models, allowing therefore a best cost execution of processes given a deadline and cost constraints.
- The proposed solution has been applied to conduct a real problem related to immune response studies.
- The new Amazon EC2 Spot pricing model has been detailed as well as how the presented approach adapts to the new changes.