Cloud Capacity Reservation for Optimal Service Deployment

Iñigo San Aniceto, Rafael Moreno-Vozmediano, Ruben S. Montero, Ignacio M. Llorente
Distributed System Architecture Research Group (dsa-research.org)
Dept. de Arquitectura de Computadores y Automática
Universidad Complutense de Madrid, 28040
Madrid, SPAIN
Email: inigosananiceto@pdi.ucm.com, rmoreno@dacya.ucm.es, rubensm@dacya.ucm.es, llorente@dacya.ucm.es

***Abstract*—Cloud computing is a profound revolution in the way it offers the computation capability. The Information Technology organizations do not need to oversize their infrastructure anymore, potentially reducing the cost of deploying their services. The main objective now is to reduce the cost of deploying a service in the cloud. Some research attempts have focus on deploying one service in multiple clouds, to benefit from different billing models. In this work, we propose a way to minimize that cost by using a single cloud provider with an optimal mixture of reserved and on-demand instances to take advantage of different billing models within the same provider. We tested this optimal combination of reserved and on-demand instances with real world workload traces. The results show a 32% deployment cost reduction compared to on-demand deployment.**

***Index Terms*—Cloud computing; capacity reservation; resource provisioning; service deployment; cost optimization**

I. INTRODUCTION

Cloud computing takes advantage of workload consolidation to operate more efficiently the resources and provide a service at lower cost. This itself is a tremendous advantage and makes the cloud computing services competitive in terms of prices. Information Technology (IT) companies used to oversize their resources to meet peak demands but now they have the option of using cloud computing [1][2][3][4][5][6][7].

Apart from the typical on-demand instances, current providers also offer reserved capacity. Although the pricing schemes for this reserved instances vary among cloud providers, they all offer discounts in the hour rates on one side and obligations or one-time payments on the other [8]. This means it is necessary a minimum amount of running hours to reduce the final price compared to on-demand instances.

In this paper, we present a novel algorithm to cover variable computation demands with mixed reserved and on-demand instances with the minimum cost. The idea is to eliminate the over-provisioning in the reserved instances to use the number of reserved instances that minimizes the final cost for the IT companies.

In addition, to avoid the performance degradation of the system, this novel algorithm estimates the number of instances that might be required in the next period, and provisions the instances in advance to hide start-up times. The provisioned instances are started-up and ready to use and they are a combination of reserved and on-demand instances.

The algorithm works as follows. In the first stage, the algorithm has to determine the optimal number of reserved instances.

The algorithm selects the number of reserved instances to reduce the cost of service for the IT companies This number is directly related with the number of running hours each reserved instance has.

Then the algorithm has to evaluate, for each period, the optimum number of provisioned instances. If the number of requested instances is lower than the number of reserved instances, all the provisioned instances are mapped on reserved instances. Otherwise, the difference would be fulfilled with on-demand instances.

To test the algorithm, reservation and provision cost of a standard instance in Amazon cloud provider [8] and real world traces from the Grid Workloads Archive are used [9].

The main contributions of this paper are the following:

1) We present an algorithm that minimizes the final cost of deploying a service in the cloud.
2) We present a model that predicts the optimal number of reserved instances for each period and use an algorithm to reserve them.
3) The model also predicts the optimum number of provisioned instances, and make advanced provision of those instances to hide start-up times.

This paper is organized as follows: Section II presents the state of the art and the current cloud computing market. Section III presents the definition of the problem with an appropriate statistical definition. Section IV presents the statistical analysis. Section V presents the reservation and provisioning algorithm. Section VI presents the improvements and Section VII presents the conclusions of the work and future work.

II. CURRENT CLOUD COMPUTING MARKET AND STATE OF THE ART

Currently, there are different pricing models in the market, being On-demand, Reservation and Spot the most common pricing schemes. Although these are the leading pricing configuration groups, there are differences among different cloud providers.

 ISBN: 978-1-61208-153-3

- **On-demand:** Probably, the most common pricing model. The main idea of this pricing configuration is to pay for the actual use with no other commitments. Most of the large providers offer this pricing model: Amazon [8], GoGrid [10], Rack Space [11] and Cloud Sigma [12] among others. Although the pricing model is similar in all the cloud providers, there are some differences.
  Amazon, for example, has some preconfigured instances with a certain amount of RAM, CPU, Storage, etc. For the following analysis, it is interesting to focus on the standard instance. It has 1.7GB of RAM, 1 Virtual core with 1 GHz and 160GB of storage. For this standard configuration, the price is 0.085$/h in N.Virginia [8].
  The situation in RackSpace is similar to the one in Amazon. There are preconfigured instances with different amount of RAM and storage and each one has a fixed price. The most similar configuration to the Amazon standard instance has 2GB of RAM, and 80 GB of storage and its price is 0.12$/h [11].
  Go Grid also offers preconfigured on-demand instances for 0.19$/h with 1GB of RAM, 1 CPU with 1GHz, and 50GB of storage [10].
  In Elastic Host, the on-demand pricing configuration is slightly different to the previous ones. There are not preconfigured instances, instead the instance types are user defined, and prices for the components are CPU(1GHz) 0.036$/h, RAM (1GB) 0.05$/h, Storage (1GB) 0.20$ /month. Comparing with the Amazon standard instance the price would be 0.164$/h [13].
  In Cloud Sigma, the on-demand pricing configuration is similar to Elastic Host but with price variability. The cost of RAM, CPU, Storage, etc. is not a fixed amount, but it is conditioned by the servers load. The boundary rates, for each charasteristic, are: CPU (1GHz) 0.0121-0.0504$/h. RAM (1GB) 0.0196-0.0579$/h. With this prices, an instance similar to the Amazon standard instance would cost 0.045-0.252$/h [12].
- **Reserved:** It is also a common pricing model. In this price configuration, there are always long-term commitments on one side, and discounts in the hour rates on the other. It is offered by most of the big cloud providers: Amazon [8], Rack Space [11], GoGrid [10] and Cloud Sigma [12] among others. The different providers also present some differences.
  Amazon establishes a one-time payment for the reservation. For each standard instance it is 227.5$ for 1 year reservation and 350$ for 3 year reservation. After the one-time payment, the discounts in hourly rates are fixed for each instance type, and they are approximately of 60%-65% depending on the type. For the standard instance the price reduces from 0.085$/h to just 0.03$/h without any compromise of use, i.e., the hourly price is charged only if the instances are running [8].
  Elastic Host uses a similar price configuration. It establishes one-time payment for reservation as Amazon does. The prices for the subscription are 77.76$ per month or 777.60$ per year, after the payment the instance prices have a fixed 50% discount regardless of the subscription period[13].
  In Cloud Sigma, the situation is different. It offers different discounts depending on the reservation period going from 3% for 3 months up to 45% for a 3 year reservation period, after the reservation is made the user has to pay for each instance as running [12].
  In Go Grid, the reserved price configuration is also slightly difference from the previous ones. It requires a monthly payment to acquire a certain quantity of usage hours. For the smallest instance, this payment is of 199$/month acquiring 2500 RAM hours. Considering 1 month has 744h a instance with 3.35GB of RAM can be used 100% of the time [10].
- **Spot instances:** This price configuration is not available in many cloud providers. The main idea of this pricing configuration is to set the maximum rate for the service hour, called bid price. Depending on the servers load, on the cloud providers, the spot pricing can change so that if the price is smaller than the bid price, the user have set the service will become available, on the other hand, a higher spot price makes the service unavailable. One of the few cloud providers that offer this price configuration is Amazon[8] .

These differences in the pricing models have lead to the creation of multiple cloud brokers. This cloud brokers try to minimize the cost of deploying the cloud service choosing the best price model across different cloud providers. There are many commercial solutions: RightScale, SpotCloud, Kavoo or CloudSwitch among others.

There are also some European projects especially oriented for Multi-Cloud deployment. Mosaic [14] is one of them and offers an open-source cloud API to develop multi-cloud oriented applications, and Optimis [15] is another that offer tools to simplify the construction and usage of hybrid clouds.

Finally, there is also some research works in this area: in [5], Moreno studies with the cost per-job with different cluster configurations. In the work we present, we also target to minimize the cost of deploying the cloud service by choosing the best price models, however, we do not use multiple-clouds; instead, we use the different price models within the cloud: on-demand and reserved instances.

For the availability problems that cloud computing might generate, some studies [16][17] focused on how to avoid availability problems, the algorithm we present faces the problem of availability with advanced provisioning based on load prediction instead of using instance leasing.

In [18], Konstanteli studies the flexible reservation periods to schedule the workflow and maintain the QoS. In this work, we have focused on the Amazon cloud provider and this provider, only offers 1 and 3 year reservation periods. After a brief study we have conclude that a 3 year reservation period is too long because the predictions are not accurate enough, and hence, we have used a 1 year fixed reservation period

 ISBN: 978-1-61208-153-3

In [19], Gardfjäll studies the credit based regulation of grid capacity allocation to avoid the performance loss due to the overuse also known as the "tragedy of the commons".

In conclusion, there are several pricing models in the market, and cloud brokers, take advantage of these differences to reduce the final prices. However, this produces several problems such as compatibility across different cloud providers that researches are trying to solve with some new tools such as Mosaic or Optimis. In this work, we targeted the same price reduction for service deployments, but we use the combination of reserved and on-demand instances in the same cloud provider. Hence, we have not any compatibility issues the multi-cloud environments produce.

III. Definition of the problem

The goal of the IT companies is to reduce the cost of deploying their service in the cloud provider without any performance degradation.

To achieve that, we propose a prediction model based on the historical data. With this prediction, the IT companies will use a mixture of reserved and on-demand instances to cover their demand.

This model estimates the number of reserved instances that minimizes the final cost, and the number of provisioned instances that fulfils the service requirements in 99.95% of cases. With this advance provisioning, it hides the start-up time (2-5 minutes [1]).

In order to create the prediction model, we obtain the cost of the cloud services and define the statistic sample space.

A. Costs

First we obtain the provisioning and reservation costs. For this analysis, we use the price configuration of a standard instance in the Amazon cloud provider [20].

Hence, the reservation cost of an instance is 227.5$/year with a provisioning cost of 0.03$/h, and on-demand instances have just a 0.085$/h provisioning cost [8].

With the previous values, we calculate the cost of the service for the IT companies. This cost will have two parts:

1) The first is the cost of reserving instances in the cloud provider

$$Cost_1(t) = Pres \times [Res(t) - Res(t-1)] \quad (1)$$

where Pres = 227.5$ is the reservation cost and Res(t) is the number of reserved instances at t [8].

2) The second is the cost of provisioning the instances. The cost of provisioning the instances will be $Pc_1 = 0.03$$/h for reserved instances and $Pc_2 = 0.085$$/h for on-demand instances [8].

$$Cost_2(t) = (R(t) \times Pc_1) \quad (2)$$

$$Cost_2(t) = (Res(t) \times Pc_1) + ((R(t) - Res(t)) \times Pc_2) \quad (3)$$

where R(t), is the number of requested instances at t.

The total cost will be

Algorithm 1 Instance mapping algorithm for cost calculation

if $R(t) \leq Res(t)$ **then**
  All the requested instances can be mapped to reserved instances and equation 2 is used.
else
  On-demand instances are necessary and equation 3 is used.
end if

$$Cost_{1year} = \sum_{t=0}^{1year} [Cost_1(t) + Cost_2(t)] \quad (4)$$

This is the equation that should be minimized by optimizing the resource reservation and provisioning of instances.

B. Definition of the Sample Space

In this work, we present a prediction tool. This prediction tool is based in a statistical analysis of the problem and hence it needs a proper definition of the sample space.

For now on, an instance will be treated as an indivisible unit being the total number of requested instances a discrete number. With this assumption the sample space will be finite and numerable. $\Omega = (0, 1, ..., L)$ where L is the maximum number of instances the service might need.

The algorithm provision a certain number of instances dividing the sample space in two relevant subsets representing mutually exclusive events.

The first $A \subset \Omega$ represents the case in which the necessary instances are less than the provisioned instances and, hence, there is no performance degradation. The second $B \subset \Omega$ where the IT companies need more instances than the ones the algorithm has provisioned and, hence, a performance degradation due to start-up time may occur [1]. Obviously the sample space satisfies $A \cap B = \phi$ and $A \cup B = \Omega$ creating a complete event system.

IV. Statistical analysis

As stated in the previous section, the goal of this work is to reduce the cost of deploying a service in the cloud provider without any performance degradation. The first step to get that reduction is to develop a statistical analysis of the workload of the IT companies. In this section, we first introduce the data used for the statistical analysis and then we describe the statistical analysis.

A. Trace data

We get the trace used for the study from the Grid Workload Archive [9]. In this website, different workload traces of different grids around the world are available.

These traces contain historical information about JobID, SubmitTime, WaitTime, RunTime, Number of Processors, Average instances Time Used, Used Memory, Requested Number of Processors, Requested Time, Requested Memory, Status, among other information.

 ISBN: 978-1-61208-153-3

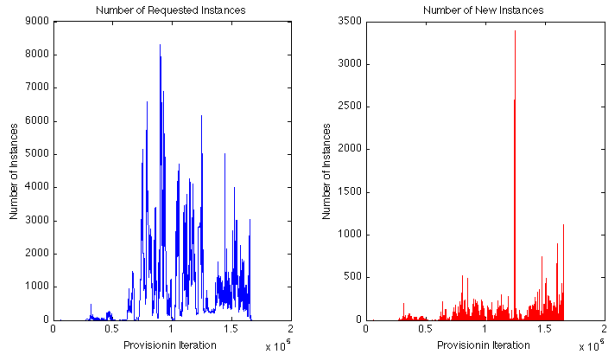

Fig. 1. Number of requested and new instances in 10 min samples from Nordugrid

From this data, we can easily obtain the number of new requested instances at each moment:

$$N(t) = \sum P_i(S_t = t) \quad \forall JobID \tag{5}$$

where N(t) are the new requested instances, P_i is the number of processors of the job i and S_t the submit time.

We can also calculate the number of terminated instances as:

$$F(t) = \sum P_i((S_t + W_t + D_t) = t) \quad \forall JobID \tag{6}$$

where F(t) are the instances that are not requested any more, W_t is the wait time and D_t is the demanded running time. The total number of requested instances at each moment is:

$$R(t) = R(t-1) + N(t) - F(t) \ \ \forall t \tag{7}$$

To evaluate the implementation of the novel algorithm we use a real world trace from NorduGrid [21]. This trace represents the load of the NorduGrid grid for nearly 3 years. With this trace, and using the equations (5) and (7) we get the number of requested instances R(t) and the number of new instances N(t) every 10 min. Figure 1 shows this values for the 3 year period.

This information is the base for the statistical analysis.

B. Statistical data model

To calculate the optimum number of reserved and provisioned instances, it is necessary to know the average usage of each instance. The instance reservation period is 1 year; hence, we use the normal distribution of the requested instances over 1-year to obtain the average utilization [22].

$$f_x(R(t)) = \frac{1}{\sqrt{2\pi\sigma^2}} \times e^{-\frac{1}{2\sigma^2}(R(t)-\mu)^2}, R(t)\epsilon[0,...L] \tag{8}$$

Where μ is the average requested instances, σ is the variance of requested instances and R(t) is the number of requested instances that can be any number from 0 to L.

However, this is not all the statistical information we have. The number of new instances can be statistically modelled as a Poison distribution if we assume that the number of users is large [22].

We model the number of new instances in the period λt with:

$$p_t(N(t)) = \frac{(\lambda t)^{N(t)}}{N(t)!} \tag{9}$$

where N(t) is the number of new instances at the moment and λt is the expected number of new instances in provisioning interval.

This last equation predicts the required number of provisioned instances. The provisioned instances are the instances that are started-up and ready to use. If the number of requested instances is lower than the number of reserved instances, all the provisioned instances will be mapped on reserved instances. Otherwise, the difference would be fulfilled with on-demand instances.

The reason to calculate the expected new instances for the following provisioning interval, and provision that value is to hide the performance degradation that the 2-5 minute launching time [1] might cause.

V. Reservation and Provisioning Algorithm

In the previous section, a statistical analysis of the historical data has been presented. Using that statistical analysis, we present two algorithms to determine the reservation and provisioning values. These algorithms use the historical load data of the IT companies and make the load predictions from different time periods.

1) With the long-term load prediction, the algorithm chooses the optimum number of reserved instances to reduce the cost of the service. The reason to use a long-term prediction is that reserved instances have 1-year utilization.
2) With the short-term load prediction, the algorithm chooses the optimal number of provisioned resources dynamically to hide launching delays. The algorithm start-up the instances the IT companies might need to offer the service to his clients.

A. Reserved Instances

In this section, we present a solution of the reservation problem.

As previously mentioned, due to the importance Amazon has on the cloud computing market, this is the cloud provider that will be used for the study.

We use the expected Differential Reservation Cost (DRM) to determine the reserved instances. Any time the expected differential reservation cost is negative, statistically reserving a new instance will reduce the final cost of deploying the service. On the other hand, if the expected differential reservation cost is positive, reserving a new instance will probably increase the final cost of deploying the service.

The expected DRC represents the difference in the expected statistical value of the cost of reserving one more instance in the cloud provider.

 ISBN: 978-1-61208-153-3

In this section, we explain all the steps given to obtain the optimum number of reserved instances. First, we introduce the expected differential reservation cost and then based on the statistical analysis, we develop a reservation algorithm that minimizes the cost.

1) Expected differential reservation cost: Suppose the IT company has already reserved n instances in the cloud provider. The IT company will provision the following instance only if more than n instances are provisioned.

From the statistics obtained from the users historical workload, the probability of provisioning more than n instances $\rho_{>n}$ can be obtained.

The expected differential reservation cost is:

$$\Delta P_{R:n+1} = Pres - (Res_{hours} \times \Delta P_c \times \rho_{>n}) \quad (10)$$

where $\Delta Pc = Pc_2 - Pc_1$ is the difference in the provisioning cost between reserved and on-demand instances, $\rho_{>n}$ is the probability of having more than n provisioned instances, and Res_{hours} is the period (in hours) that the instance is reserved.

Let us see this with one example: The cost of a standard instance in Amazon is 0.085$/h for on-demand and 0.03$/h for reserved instances. The instances are reserved for 1 year (8760h) paying 227.5$ for this reservation. Imagine that the provisioned instances follow a normal distribution with a mean of 100 and a variance of 10.

An iteration starts checking the first machine to see if is worthwhile reserving based on the expected DRC. In order to make the example concise, we show only the two key iterations:

- Iteration $n°1$ to 100:

$$\Delta P_{R:n} \ll 0 \quad n = 1, 2, ...100 \quad (11)$$

  At the end of the reservation period the IT company expects to pay a lot less in each iteration. In this experiment, the IT company pays a total of 74192.3$ with no reserved machines and 50929.7$ with 100 reserved machines.

- Iteration $n°$ 101: The DRC off adding the 101-th reserved machine is:

$$\Delta P_{R:101} = 227.5\$ - (8760h \times (0.085\$/h - 0.03\$/h) \times (1 - normcdf(100, 100, 10))) = -13.4\$ \quad (12)$$

  where normcdf is the normal cumulative distribution function with the values: test value, mean and variance. At the end of the reservation period the IT company expects to pay 13.4$ less to the cloud provider than reserving 100 machines. In this experiment, it pays a total of 50923.8$ or 5.9$ less than with 100 reserved machines. Hence, it is worthwhile to reserve the 101-th machine.

- Iteration $n°$ 102: The DRC off adding the 102-th reserved machine is:

$$\Delta P_{R:102} = 227.5\$ - (8760h \times (0.085\$/h - 0.03\$/h) \times (1 - normcdf(101, 100, 10))) = +5.8\$ \quad (13)$$

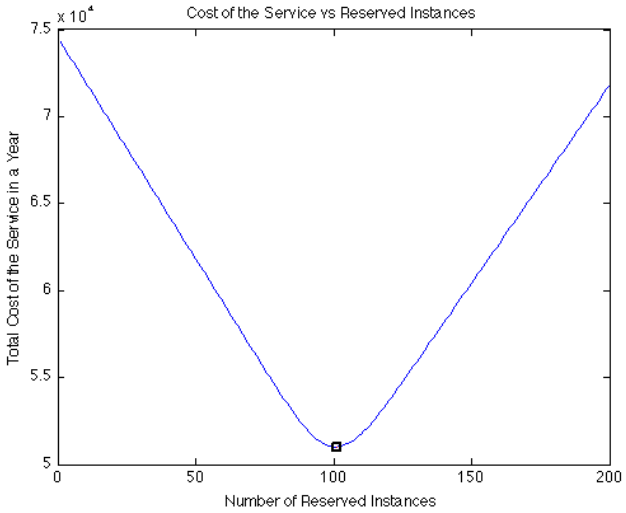

Fig. 2. Total cost in a year vs the number of reserved instances

This time the IT company expects to pay more to the cloud provider because the expected DRC of the 102-th machine is positive. In this experiment it pays a total of 50938$ or 14.2$ more than with 101 reserved reserving 102 machines. The 102 instance will not reach the minimum number of hours that make the reservation worthwhile. Hence, the reservation cost will be higher than the discount obtained from the price difference.

The algorithm conclude that reserving 101 machines is the most economical option. The Figure 2 shows the final cost of the service for the IT company in this experiment after a year with different number of reserved instances.

To make the algorithm faster, the implemented method does not calculate the expected DRC. The method just gets the last reserved instance with a negative expected DRC.

To obtain that number of reserved instances, the method uses the limiting percentage of utilization that provide a negative value of the DRC.

2) Reservation Algorithm: Applying the previous statistics and the utilization value that makes the expected DRC negative we have that:

$$F_x(Res(t)) = \int_{Res(t)}^{\infty} \frac{1}{\sqrt{2\pi\sigma(t)^2}} e^{-\frac{1}{2\sigma(t)^2}(R(t)-\mu(t))^2} dR(t) = \rho_{res}^{min} \quad (14)$$

where Res(t) is the number of reserved instances, μ is the mean of R(t) in the last year, σ is the variance of R(t) in the last year and ρ_{res}^{min} is the minimum load to make the expected DRC negative which in Amazon is 47% [8].

This equation does not get the expected DRC of each instance, however, it calculates which is the last instance that has a negative expected DRC.

In Amazon, we can only change Res(t) to a higher value and it is necessary to wait one year to reduce. Hence, the

 ISBN: 978-1-61208-153-3

following algorithm is used to determine the Res(t) at each moment.

Algorithm 2 Instance reservation algorithm

if $Res(t) \geq Res(t-1)$ **then**
  $Res(t)$
else $\{Res(t) < Res(t-1)$ and No reservation expires$\}$
  $Res(t) = Res(t-1)$
else $\{Res(t) < Res(t-1)$ and n reservations expire$\}$
  if $Res(t) \leq (Res(t-1) - n)$ **then**
    $Res(t) - n$
  else
    $Res(t)$
  end if
end if

In the next section, we present the results for reserved and provisioned instances.

B. Provisioned Instances

In this section, we present a solution for the provisioning problem. This problem has a direct relationship with the possible performance degradation due to the 2-5 minute launch time of new instances [1]. In this paper, we set this parameter to 0.05% because this percentage will produce a negligible performance degradation (the new provisioned instances will be ready to use in 99.95% of the time).

With the Poisson distribution represented in (9), the algorithm sets the value of provisioned instances P as

$$P_t(P(t) - R(t-1)) = \sum_{P_2(t)-R(t)}^{\infty} \frac{(\lambda t)^N(t)}{N(t)!} = \rho_{up}^{opt} \quad (15)$$

where $P(t) - R(t-1)$, is the difference between the requested and the provisioned instances. The reason to use this difference is that this statistical distribution calculates expected new instances at t.

C. Results

The optimum algorithm would provision in advance exactly the same instances that the ones requested and reserve instances in advance, only with more than 47% of load, as explained in the Section V-A. However, this is impossible because it means that the IT company knows his computation needs in advance.

Knowing which is the optimum result, the closer the algorithm is to this result the better it is. A good algorithm is the one that provision close to the requested instances, but always provisioning more than the requested instances, because if it provisions less a possible performance degradation occurs.

Figure 3 shows the number of provisioned and reserved instances compared to the number of requested instances at each moment.

In Figure 4, the percentage of time in which underprovisioning occur is shown. This is a way to show the

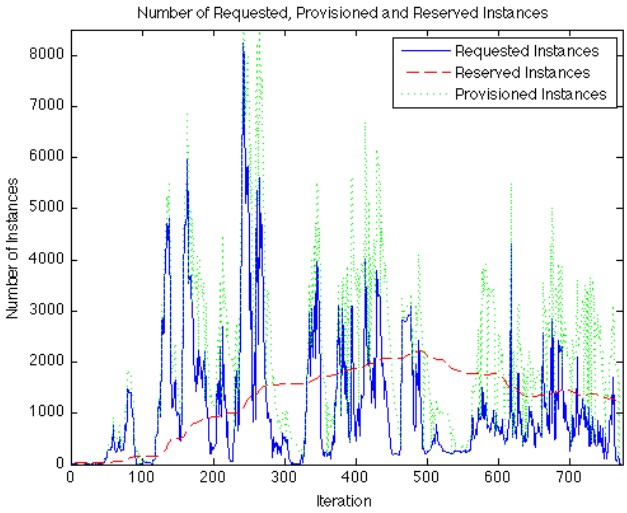

Fig. 3. Provisioning and reserved instances vs requested instances

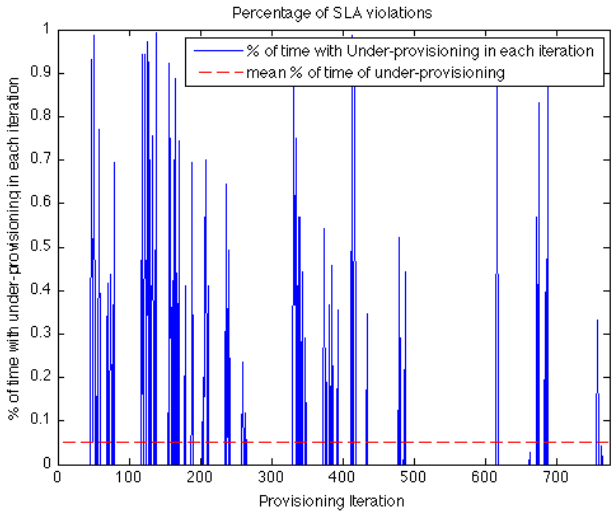

Fig. 4. Percentage of underprovisioning in each provisioning period

performance loss that occur in the system in a standardized way. The mean time of under-provisioning is 0,05%. This was the goal when defining the provisioning value and hence, the prediction model is accurate.

One of the most influential factors is the cost of the service for the user. This cost is the one that would determine if the service is competitive or if another solution is preferable.

To show this factor the cost for the user of using this algorithm has been recorded and compared with Amazons on-demand instances. Figure 5 shows the cost for the service user of using the broker.

At the end of the period the IT company have spent $5x10^5$ dollars or 32% less in cloud computing services.

VI. Improvements

In this section, we present the improvements of the basic prediction model and test the performances of each improve-

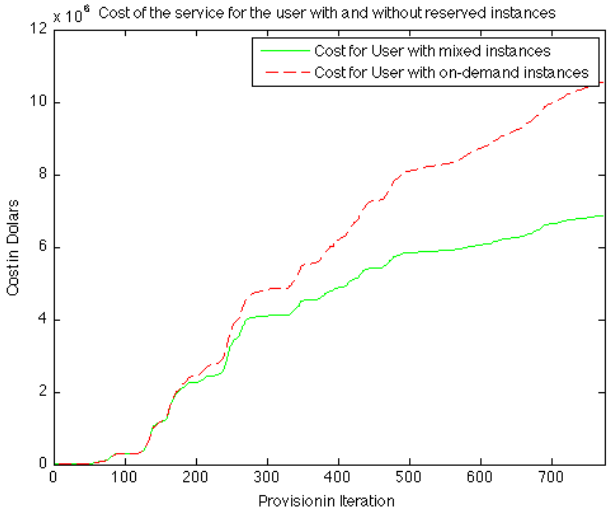

Fig. 5. Cost of the service for the user.

ment.

A. Reconfiguration based on under-provisioning

If an under-provisioning occur, the actual provisioned instances and requested instances should be analysed to see if there should be a change in the provisioned instances. This is extremely crucial to avoid performance degradation.

Even if the prediction model forecasts the necessary provisioned instances for a certain probability of under-provisioning there may be a change in the user patterns at a certain moment. If this change in the patterns creates continuous under-provisioning the performance of the service drops [23]. When under-provisioning occur, the algorithm recomputes the number of provisioned instances as the number of requested instances at the moment plus the number of expected new instances in the prediction interval.

B. Close control loop

The second improvement was focused in adjusting the statistic and prediction intervals to determine which was the one with the smaller prediction error in the expected under-provision value.

The statistic interval is the period in which the algorithm apply the Poisson distribution to predict the future values. The prediction interval is the period in which this future values are predicted.

The results presented in Table I represent the average error in the prediction of the algorithm at the end of the trace.

What we see here is that the error is not the same for different statistic and prediction ranges. If the predictions were perfect, they all should present the same error value and this value should be zero.

$$|\rho_{up}^{opt} - \rho_{alg}| = 0 \quad (16)$$

TABLE I
MEAN ERROR OF THE PREDICTOR AT THE END OF THE TRACE CONSIDERING DIFFERENT STATISTIC AND PREDICTION RANGES

Statistics / Prediction	1 day	2 days	3 days	1 week	2 weeks	3 weeks
3 hours	0.025	0.024	0.023	0.022	0.019	0.019
6 hours	0.023	0.024	0.02	0.02	0.019	0.018
9 hours	0.024	0.019	0.019	0.019	0.017	0.013
12 hours	0.02	0.013	0.014	0.012	0.014	0.012
18 hours	0.025	0.021	0.021	0.017	0.014	0.007
24 hours	0.013	0.02	0.011	0.02	0.012	0.002

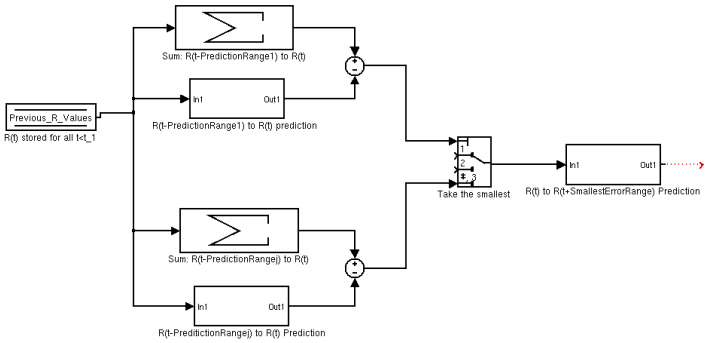

Fig. 6. Close control loop that sets optimum ranges

where ρ_{up}^{opt} is the probability of under-provisioning for which the algorithm is designed, and ρ_{alg} the real under-provisioning it achieves.

In this experiment, it is clear that the bigger the statistics and prediction ranges we take, the smaller the error is, but this can not be applied in all cases. To solve that situation, we present a tool that automatically set the optimum historic and prediction range.

The algorithm first select many different statistic and prediction ranges in a near past. With the results, it compares the errors, and it applies the statistic and prediction range that generates the smallest error for the next prediction .

With this method, we get the expected probability of under-provision with the best accuracy.

This is useful because the number of requested instances may exhibit a variance in the statistics with the time. Figure 6 shows a schematic view of the close control loop that the algorithm implements to select the best ranges.

The algorithm used to implement this control loop is explained in the following lines.

Algorithm 3 Close control loop algorithm

```
for i = 1 : Statistic ranges do
  for j = 1 : Prediction ranges do
    | ρ_opt − ρ_real |= error_ij
    if error_ij < errormin then
      errormin = error_ij → best_ij
    end if
  end for
end for
```

Applying this solution to the trace the average under-provisioning obtained at the end of the trace is remarkably close to 0,05%.

VII. Conclusion and future work

This work studies the reservation and provisioning values that minimize the cloud computing service cost with a controlled performance degradation. To reduce the cost, the algorithm uses mixed on-demand and reserved instances a single cloud provider.

We tested the algorithm that reserve and provision dynamically with real world traces obtained from the Grid Workload Archive, and compared the result after different improvements. The results show that the IT companies reduce their cost of service deployment by up to 32% with less than 0.05% performance degradation.

As future work, we will consider flexible reservation periods with different discounts. As well as, other kinds of workloads to determine, how the statistic models change and how are results altered.

VIII. Acknowledgment

The research leading to these results has received funding from the European Unions Seventh Framework Programme ([FP7/2007-2013]) under grant agreement n^o 261552 (StratusLab); from Consejería de Educación of Comunidad de Madrid, Fondo Europeo de Desarrollo Regional, and Fondo Social Europeo through MEDIANET Research Program S2009/TIC-1468; and from Ministerio de Ciencia e Innovación of Spain through research grant TIN2009-07146.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Tech. Rep., 2009.

[2] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, "Sky computing," *Internet Computing, IEEE*, pp. 43 –51, sept 2009.

[3] E. Walker, "The real cost of a cpu hour," University of Texas at Austin, Tech. Rep., April 2009.

[4] M. D. Assunção, A. D. Costanzo, and R. Buyya, "Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters," *Proceedings of the 18th ACM international symposium on High Performance Distributed Computing*, pp. 141–150, 2009.

[5] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Multi-cloud deployment of computing clusters for loosely-coupled mtc applications," *Transactions on Parallel and Distributed Systems*, 2010.

[6] M. de Assunção, A. di Costanzo, and R. Buyya, "A cost-benefit analysis of using cloud computing to extend the capacity of clusters," *Cluster Computing*, Jan 2010.

[7] R. Harms and M. Yamrtino, "EU Public Sector Cloud Economics," Microsoft, Tech. Rep., 2011.

[8] "Amazon pricing web page, http://aws.amazon.com/ec2/pricing," May 2011. [Online]. Available: http://aws.amazon.com/ec2/pricing

[9] "The grid workload archieve web page, http://gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Main.Home," May 2011. [Online]. Available: http://gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Main.Home

[10] "Gogrid pricing web page, http://www.gogrid.com/cloud-hosting/cloud-hosting-pricing.php," May 2011. [Online]. Available: http://www.gogrid.com/cloud-hosting/cloud-hosting-pricing.php

[11] "Rackspace pricing web page, http://www.rackspacecloud.com/cloud_hosting_products/servers/pricing/," May 2011. [Online]. Available: http://www.rackspacecloud.com/cloud_hosting_products/servers/pricing/

[12] "Cloud sigma pricing web page, http://www.cloudsigma.com/en/pricing/price-schedules," May 2011. [Online]. Available: http://www.cloudsigma.com/en/pricing/price-schedules

[13] "Elastichost pricing web page, http://www.elastichosts.com/cloud-hosting/pricing," May 2011. [Online]. Available: http://www.elastichosts.com/cloud-hosting/pricing

[14] M. Armbrust, A. Fox, R. Griffith, and A. Joseph, "mOSAIC," European Commission: Information Society and Media, Tech. Rep., May 2010. [Online]. Available: www.mosaic-cloud.eu

[15] "Optimis home page, http://www.optimis-project.eu/content/welcome-optimis," May 2011. [Online]. Available: http://www.optimis-project.eu/content/welcome-optimis

[16] B. Sotomayor, "A resource management model for vm-based virtual workspaces," Master's thesis, The University of Chicago, 2007.

[17] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Capacity leasing in cloud systems using the opennebula engine," *Workshop on Cloud Computing and its Applications*, October 2008.

[18] K. Konstanteli, D. Kyriazis, T. Varvarigou, T. Cucinotta, and G. Anastasi, "Real-Time Guarantees in Flexible Advance Reservations," in *Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International*, 2009, pp. 67–72.

[19] P. Gardfjäll, E. Elmroth, L. Johnsson, O. Mulmo, and T. Sandholm, "Scalable grid-wide capacity allocation with the swegrid accounting system (sgas)," *Concurr. Comput. : Pract. Exper.*, vol. 20, pp. 2089–2122, December 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1458640.1458641

[20] "Amazon instances web page, http://aws.amazon.com/ec2/instance-types/," May 2011. [Online]. Available: http://aws.amazon.com/ec2/instance-types/

[21] "Nordugrid trace web page, http://gwa.ewi.tudelft.nl/pmwiki/reports/gwa-t-3/trace_analysis_report.html," May 2011. [Online]. Available: http://gwa.ewi.tudelft.nl/pmwiki/reports/gwa-t-3/trace_analysis_report.html

[22] G. Zhao, J. Liu, Y. Tang, W. Sun, F. Zhang, X. Ye, and N. Tang, "Cloud computing: A statistics aspect of users," *Cloud Computing*, pp. 347–358, 2009.

[23] V. Machiraju, M. Sayal, A. V. Moorsel, and F. Casati, "Automated sla monitoring for web services," *IEEE International Symposium on Integrated Network Management*, pp. 28–41, 2002.

 ISBN: 978-1-61208-153-3