Decision Model for Provisioning Virtual Resources in Amazon EC2

Cheng Tian, Ying Wang, Feng Qi, Bo Yin
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
Beijing, China
tiancheng@bupt.edu.cn, wangy@bupt.edu.cn, qifeng@bupt.edu.cn, yinbo@bupt.edu.cn

Abstract—**Nowadays computing resources can be acquired from IaaS cloud providers in different purchasing options. Taking Amazon Elastic Compute Cloud (EC2) for instance, there are three purchasing models, and each option has different price and yields different benefit to clients. The issue that we address in this paper is how cloud users could make a provisioning plan for computing resources. We propose a model which is based on the characteristics of three purchasing options provided by Amazon EC2, which can be used for guiding the capacity planning activity.**

Keywords-cloud computing; capacity planning; spot pricing; cost optimization; Amazon EC2;

I. INTRODUCTION

Traditionally, the enterprise built its IT infrastructure in advance based on its peak or average demand. Thus the enterprise took the cost of construction, operation and maintenance of IT infrastructures. Nowadays the enterprise can directly purchase computing resources from cloud providers. Amazon Elastic Compute Cloud (EC2) can be described as the most successful IaaS cloud computing provider. It offers three purchasing options, each option having its unique purchasing model. Although cloud providers have taken the burden of the cost of IT infrastructures from the enterprise, it is left to the enterprise to make an optimization provisioning plan according to computing resources provided by cloud providers.

The resource provision in cloud computing has been studied by many researchers. An optimization formulation is proposed in [1] to determine an optimal schedule of on-demand instances. The studies in [2, 3] focus on the most effective way to buy spot instances in the short-term plan. Two long-term planning methods is proposed in [4, 5] respectively, both without considering the effect of using spot instances. To the best of our knowledge, using all three purchasing options in the long-term plan for cost optimization of resource provision in cloud computing was not studied before. The intention of this work is to provide a decision model for provisioning computing resources to obtain the optimal solution for purchasing three options provided by Amazon EC2, while meeting demands of clients.

The rest of this paper is organized as follows: Section II describes the purchasing models and analyzes the spot instance pricing. The proposed model is presented in Section III. Section IV presents the performance evaluation results. The paper is summarized in Section V.

II. PURCHASING MODELS AND SPOT INSTANCE PRICING

A. Amazon EC2 Purchasing Models

Amazon EC2 offers three purchasing models. ***On-demand Instances*** let clients pay for computing resources by the hour with no long-term commitments or upfront payments. ***Reserved Instances*** require clients to make a one-time, upfront payment for an instance, reserve it for a one or three year term, and pay a lower rate for each hour running that instance. Both reserved and on-demand instances remain active until terminated by the client. ***Spot Instances*** allow clients to specify the maximum hourly price that they are willing to pay to run a particular instance type. Amazon sets a Spot Price for each instance type, which is the price all clients will pay to run a spot instance for that given hour. Clients get the spot instance if the maximum hourly price specified is higher than the Spot Price; otherwise they wait. Spot instance requests can be one-time or persistent. A one-time request will only be satisfied once; a persistent request will remain in consideration after each instance termination. By using a persistent request, clients can launch instances any time the Spot Price is below the specified price.

B. Spot Instance Pricing

Amazon's description of "How Spot Instances Work" [6] gives the impression that spot prices are set through a uniform price, sealed-bid, market-driven auction. "Uniform price" means all bidders pay the same price. "Sealed-bid" means bids are unknown to other bidders. "Market-driven" means the spot price is set according to the clients' bids. But in [7], Ben-Yehuda et al. demonstrated that spot prices are usually not market-driven. Rather, they are typically generated at random from within a tight price interval via a dynamic hidden reserve price. In order to learn more about the way Amazon prices spot instances, we did statistical analysis on spot instance historical prices of 64 (=8×4×2) spot price trace files associated with the 8 instance types, the 4 regions, and 2 operating systems (Linux and Windows) from Jan 2010 to Dec 2011. The data was collected by Lossen [8].

Our analysis result support Ben-Yehuda et al.'s thesis. Due to space limitations, we only display analysis results of 8 instance types of Linux operating system in us-east region here. As shown in Fig. 1, for all instance types, the probability distributions of spot prices in different price intervals of 2010 are approximately equal to that of 2011. The possible reasons may lie in two aspects: one is the claim of Ben-Yehuda et al. is

This work was supported by 863 Program (2011AA01A102, 2012AA050801), National Key Technology R&D Program (2012BAH35F02) and the Fundamental Research Funds for the Central Universities BUPT 2011RC0501

978-3-901882-48-7

true; the other is the similarity of Amazon business in adjacent years. It is not our concern that which one is true. In any case, they all support that the probability distributions of spot prices in adjacent years are similar.

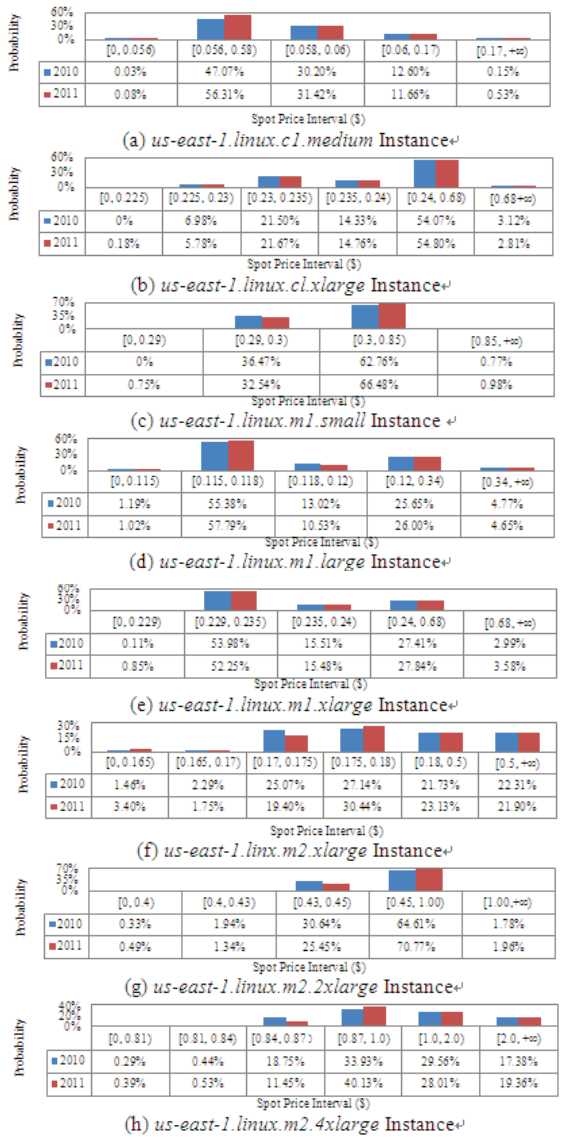

Figure 1. Spot prices probability distributions in different price intervals in 2010 and 2011of 8 instance types of Linux operatin system in us-east region

Spot Instance price is normally cheaper than that of On-demand Instance, even cheaper than Reserved Instance. The research [9] by Wee has shown that the spot price is 52.3% cheaper than the standard price of equivalent instance type on average. In Fig. 1, we can also see that the probability of the historical prices of *us-east-1.linux.cl.xlarge* Instance above its on-demand price (i.e. $0.68) in 2010 and 2011 is just about 3%, and the probability of the historical prices below its reserved price (i.e. $0.24) is almost 50%.

Many enterprise application workloads experience huge fluctuations over time. In order to achieve desired quality of service (QoS), a simple way is to reserve enough reserved instances according to the highest request arrival rate. This approach is uneconomical, since many resources will sit idle during normal or low operational periods. Another way which won't lead to many idle resources is to buy on-demand or spot instances. This approach may be more expensive because on-demand instance is more expensive than reserved instance. Although spot instance is cheaper than reserved instance, its price is fluctuant and it may be terminated by the cloud provider whenever the bid price is lower than the spot price. If a lot of spot instances are used, QoS could not be ensured. In this paper, we will propose a decision model used to make the provisioning plan which not only has low cost but also ensure the application QoS.

III. PROPOSED MODEL

We assume it is possible to approximately predict such demand in terms of mean normal arrival rate λ_0 and mean arrival rates during load surges λ_+ and periods of low demand λ_-. It has been demonstrated in [5] that it's more profitable to divide the request period into three different request periods when making a capacity plan. Let ΔT_0, ΔT_+ and ΔT_- denote the amount of time the application under normal load, high load and low load respectively, and ΔT_{total} denote the sum of three periods (typically a year).

Let Ω denote the set of all instance types. Let $n_i^{(r)}$ denote the reserved number of reserved instances type i. Let $p_i^{(r)}$ denote the one-time fee of reserved instance type i and $p_i^{(u)}$ denote the unit price to execute reserved instance type i. The total cost of the reserved instances in a year is:

$$C_{total}^{(r)} = \sum_j \sum_{i\in\Omega} p_i^{(u)} n_{ij}^{(u,0)} \Delta t_j^{(0)} + \sum_j \sum_{i\in\Omega} p_i^{(u)} n_{ij}^{(u,+)} \Delta t_j^{(+)} + \sum_j \sum_{i\in\Omega} p_i^{(u)} n_{ij}^{(u,-)} \Delta t_j^{(-)} + \sum_{i\in\Omega} p_i^{(r)} n_i^{(r)} \quad (1)$$

$$\text{where } \sum_j \Delta t_j^{(0)} = \Delta T_0, \sum_j \Delta t_j^{(+)} = \Delta T_+, \sum_j \Delta t_j^{(-)} = \Delta T_-$$

Let Δt_j denote the minimum time interval in which the number of reserved instances used doesn't change. And Let $n_{ij}^{(u)}$ denote the actual number of reserved instance type i used in the time interval Δt_j.

We can simplify (1) into

$$C_{total}^{(r)} = \sum_{i\in\Omega} p_i^{(u)} N_i^{(u,0)} + \sum_{i\in\Omega} p_i^{(u)} N_i^{(u,+)} + \sum_{i\in\Omega} p_i^{(u)} N_i^{(u,-)} + \sum_{i\in\Omega} p_i^{(r)} n_i^{(r)} \quad (2)$$

where $N_i^{(u,0)}$, $N_i^{(u,+)}$ and $N_i^{(u,-)}$ denote the total number of utilized server-hours of reserved instance type i in the period ΔT_0, ΔT_+ and ΔT_- respectively.

Let $p_i^{(o)}$ denote the unit price to execute on-demand instance i. The method of calculating the total cost of on-demand instances in a year is similar to the method of calculating the total cost of reserved instances. We can express the total cost of the on-demand instances in a year as follow:

$$C_{total}^{(o)} = \sum_{i\in\Omega} p_i^{(o)} N_i^{(o,0)} + \sum_{i\in\Omega} p_i^{(o)} N_i^{(o,+)} + \sum_{i\in\Omega} p_i^{(o)} N_i^{(o,-)} \quad (3)$$

where $N_i^{(o,0)}$, $N_i^{(o,+)}$ and $N_i^{(o,-)}$ denote the total number of server-hours of on-demand instance type i in the period ΔT_0, ΔT_+ and ΔT_- respectively.

In economics, there is a specialized term named "Investment Diversification". The purpose of investment diversification is to reduce the investment risk and ensure the stability of earnings. As the price of spot instance is fluctuant and it may be terminated by the cloud provider whenever the bid price is lower than the spot price, the concept of "Investment Diversification" also applies to purchasing spot instances. When the persistent request option is used to purchasing spot instances, and if only one specified price is given out, the client will lose all spot instances once the Spot Price increases above the specified price. But if more than one specified price are given out, the client may just lose part of spot instances when the Spot Price changes.

According to the analysis result in subsection B of section III, we can get the approximate probability of spot prices below a specified price of the next year according to the probability distribution of spot prices of the previous year. Let $\mathcal{L}_i$ denote the total number of specified prices made for spot instance type i. Let $p_{ij}^{(s)}$ denote the *j-th* specified price of spot instance type i, and $n_{ij}^{(s)}$ denote the number of spot instances type i provisioned under the specified price $p_{ij}^{(s)}$. Let $\pi_{ij}^{(s)}$ denote the approximate probability of spot prices in the next year below the specified price $p_{ij}^{(s)}$. The total cost of spot instances in a year is:

$$C_{total}^{(s)} = \Delta T_{total} \cdot \sum_{i\in\Omega} \sum_{j}^{\mathcal{L}_i} p_{ij}^{(s)} n_{ij}^{(s)} \pi_{ij}^{(s)} \quad (4)$$

At any point in time, a running spot instance can be terminated by Amazon. Nevertheless, it is assumed that the checkpointing mechanism is used to record the current state of application running on spot instances [3]. Therefore, whenever the spot instances are terminated by Amazon, on-demand or reserved instances will be created. Then, the application with checkpointed state from terminated spot instances will be resumed on the new on-demand or reserved instances. So there will not be any requests dropped.

Although we make several specified prices for an instance type, the probability of losing all spot instances of instance type i still exists. If the most of demands are planned to be satisfied by spot instances and only a few reserved instances are reserved in the plan, lots of on-demand instances have to be purchased to satisfy the demand planned to be satisfied by spot instances once these spot instances could not be acquired. This may lead to the result that the actual cost is far more than the expected cost. Thus, we need to control the amount of spot instances in the provisioning plan.

The Spot Instance control model is formulated as follows:

$$\frac{1}{3} \cdot n_{ij}^{(s)} \mu_i \cdot \sum_{\omega\in\{0,+,-\}} \frac{1}{\lambda_\omega} \le \beta_{ij} \cdot \pi_{ij}^{(s)} \quad (5)$$

where μ_i denotes the service rate of instance type i, and β_{ij} denotes the ability of the client to assume the risk of losing spot instances. The larger β_{ij} is, the more spot instances will be used in the plan. β_{ij} depends on both demand condition and the client's bias of obtaining a possible lower cost by taking the risk of paying more money.

From the above, the computing resources provisioning model proposed can be expressed as follows:

$$\min_{n_i^{(r)}, n_{ij}^{(s)}} \quad C_{total}^{(r)} + C_{total}^{(o)} + C_{total}^{(s)} \quad (6)$$

s.t.

$$\sum_{i\in\Omega} N_i^{(u,0)} \mu_i + \sum_{i\in\Omega} N_i^{(o,0)} \mu_i + \Delta T_0 \cdot \sum_{i\in\Omega} \sum_{j}^{\mathcal{L}_i} n_{ij}^{(s)} \pi_{ij}^{(s)} \mu_i \ge \lambda_0 \Delta T_0 \quad (7)$$

$$\sum_{i\in\Omega} N_i^{(u,+)} \mu_i + \sum_{i\in\Omega} N_i^{(o,+)} \mu_i + \Delta T_+ \cdot \sum_{i\in\Omega} \sum_{j}^{\mathcal{L}_i} n_{ij}^{(s)} \pi_{ij}^{(s)} \mu_i \ge \lambda_+ \Delta T_+ \quad (8)$$

$$\sum_{i\in\Omega} N_i^{(u,-)} \mu_i + \sum_{i\in\Omega} N_i^{(o,-)} \mu_i + \Delta T_- \cdot \sum_{i\in\Omega} \sum_{j}^{\mathcal{L}_i} n_{ij}^{(s)} \pi_{ij}^{(s)} \mu_i \ge \lambda_- \Delta T_- \quad (9)$$

$$N_i^{(u,\omega)} \le n_i^{(r)} \cdot \Delta T_\omega, \quad \forall i \in \Omega, \forall \omega \in \{0,+,-\} \quad (10)$$

$$\frac{1}{3} \cdot n_{ij}^{(s)} \mu_i \cdot \sum_{\omega\in\{0,+,-\}} \frac{1}{\lambda_\omega} \le \beta_{ij} \cdot \pi_{ij}^{(s)}, \quad \forall i \in \Omega, \forall j \in \mathcal{L}_i \quad (11)$$

$$N_i^{(u,\omega)}, N_i^{(o,\omega)} \in \{0,1,\cdots\}, \forall i \in \Omega, \ \forall \omega \in \{0,+,-\} \quad (12)$$

$$n_i^{(r)}, n_{ij}^{(s)} \in \{0,1,\cdots\}, \quad \forall i \in \Omega, \ \forall j \in \mathcal{L}_i \quad (13)$$

The objective function in (6) is to minimize the expected total provisioning cost. The decision variables of the model are $n_i^{(r)}$, $n_{ij}^{(s)}$, $N_i^{(u,\omega)}$ and $N_i^{(o,\omega)}$, among which only $n_i^{(r)}$ and $n_{ij}^{(s)}$ are objective decision variables of our model. We use $n_i^{(r)}$ to decide the number of reserved instances and $n_{ij}^{(s)}$ to decide the number of spot instances with the specified price $p_{ij}^{(s)}$. The constraint in (7), (8) and (9) ensures that the total amount of instances must satisfy demand of three different periods.

IV. Performance Evaluation

The model is implemented and solved by the GAMS/CPLEX optimization solver [10]. Furthermore, the real historical data of demand and spot prices are used in the evaluation. The workloads obtained from a medium-sized e-commerce website represent the demand. And the historical spot prices are gathered from Amazon EC2.

A. Basic Parameter Setting for Evaluation

First, we describe the scenario studied in the evaluation: *medium-sized site with small workload surges*. The mean request rate of this site is around 300 requests per second (rps) and peak load is twice the normal expected load. We estimate its demands in 2011 according to its workloads in 2010. The predicted demands are detailed in TABLE I. One instance type in US-East region Linux/UNIX *c1.xlarge* is considered. The practical prices of this instance type of three options are used in the evaluation. The basic input parameters of the experiment are summarized in TABLE I.

TABLE I. BASIC INPUT PARAMETERS FOR THE MODEL

	Parameter	Value		Parameter	Value
1	λ_0	250rps	6	ΔT_-	1348h
2	λ_+	520rps	7	μ	10rps
3	λ_-	80rps	8	$p^{(r)}$	$1820
4	ΔT_0	5369h	9	$p^{(u)}$	$0.24
5	ΔT_+	2043h	10	$p^{(o)}$	$0.68

B. *Adjustment of Specified Prices of Spot Instances*

For the first study, we discuss the impact of the way to set specified prices on the total cost. We make provisioning plans for the application demand in 2011 by using the real historical spot prices in 2010 of Linux/UNIX *c1.xlarge*. Then we run plans under the real historical prices and application demands in 2011. In TABLE II, the "*Expected Cost*" means the expected cost of the provisioning plan, and the "*Actual Cost*" means the cost generated by running the plan under the real historical prices and application demands in 2011.The related parameters and the results of six different ways to set specified prices are shown in TABLE II.

In Case 1, we just set one low specified price for spot instances. We can see that both the expected cost and the actual cost are high. In Case 2, we set one high specified price. Although the actual cost is much lower than the cost of Case 1, it is still 2.9% higher than the cost of Case 3， which uses two specified prices. According to the probability distribution of spot prices in 2010, there is no need to set three specified prices, which could be found from the result of Case 6. We need to analyze the probability distribution of the spot prices of the previous year when we make the decision of specified prices.

From the expected costs and the actual costs of Case 3, 4 and 5, we can see that the decision of β is very important. β_1 and β_2 of Case 3 are both larger than the relative β of Case 4. The expected cost and the actual cost of Case 3 are smaller than the relative cost of Case 4 respectively. However, although β_1 and β_2 of Case 5 are also larger than the relative β of Case 3 and the expected cost of Case 5 is lower than the expected cost of Case 3, its actual cost are higher than the actual cost of Case 3. The decision of β must be based on the demand condition and the ability to assuming a risk.

TABLE II. EXPECTED COSTS AND ACTUAL COSTS OF DIFFENRENT SPECIFIED PRICES CASES

Parameter	Case					
	1	*2*	*3*	*4*	*5*	*6*
$p_1^{(s)}$	$0.228	$0.239	$0.239	$0.239	$0.239	$0.239
$p_2^{(s)}$	×	×	$0.228	$0.228	$0.228	$0.236
$p_3^{(s)}$	×	×	×	×	×	$0.228
$\pi_1^{(s)}$	5.41%	42.83%.	42.83%	42.83%	42.83%	42.83%
$\pi_2^{(s)}$	×	×	5.41%	5.41%	5.41%	34.97%
$\pi_3^{(s)}$	×	×	×	×	×	2.38%
β_1	2	3	3	2	4	3
β_2	×	×	2.5	1	3	0.4
β_2	×	×	×	×	×	2
EC ($)	129,969.8	113,917	113,849.4	119,029	109,327.4	113,055.
AC ($)	130,820	120,912	117,459	122,161	119,631	118,104

EC denotes "Expected Cost", AC denotes "Actual Cost"

C. *Comparison with Other Planning Models*

In this subsection, we compare results of plans generated by our model with other four capacity planning methods. The four planning methods are described as follows.

1) *Method 1* – Only use the on-demand instances to satisfy the application demand;

2) *Method 2* – Only use the reserved instances and reserve the amount of instances according to the peak load;

3) *Method 3* – Use both on-demand and reserved instances. The amount of reserved instances is decided by only considering prices of reserved and on-demand instances.

4) *Method 4* – Use the three kinds of instances. The amount of reserved instances is decided by only considering prices of reserved and on-demand instances. Use one-time request to buy spot instances. The spot price prediction method used was proposed in [11].

Method 5 is the model proposed in this paper. Plans of Case 1~6 shown in TABLE II are used in the comparison. The plans generated by these five methods and their costs are shown in TABLE III. From the experiment results, except for the cost of Case 1 with an unreasonable specified price, we can see that the costs of Case 2~6 are all lower than the costs of plans generated by the other four methods.

TABLE III. PLANS MADE BY FIVE METHODS

Method		Parameter				
		$n^{(r)}$	$n_1^{(s)}$	$n_2^{(s)}$	$n_3^{(s)}$	*Cost ($)*
1		×	×	×	×	170,846.6
2		52	×	×	×	154,938.8
3		25	×	×	×	130,070
4		25	×	×	×	127,461
5	*Case 1*	25	2	×	×	130,820
	Case 2	16	21	×	×	120,912
	Case 3	16	21	2	×	117,459
	Case 4	19	14	1	×	122,161
	Case 5	13	28	2	×	119,631
	Case 6	15	21	2	1	118,104

In Fig. 2, we present the cost (in percentage) to be saved when our model is used in comparison with the other four methods. We compare the costs of Case 1~6 with that of four methods separately. We can see that only using on-demand instances or reserved instances (Method 1 and Method 2) to satisfy the application demand is very uneconomical. The cost of Method 1 is about 45% higher than that of Method 5 and the cost of Method 2 are about 30% higher than that of Method 5. The cost of Method 3 is close to the cost of Method 4, which means only using spot instances in short-term planning has little effect on reducing the total cost. As to our method, we can see that costs of Method 3 and Method 4 are about 10% larger than the cost of our method. The effect on reducing the cost of Method 5 is obvious.

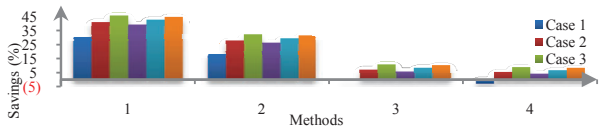

Figure 2. Savings obtained when our model is used in comparison with the other four methods

V. CONCLUSION AND FUTURE WORK DIRECTIONS

In this paper we proposed a decision model for provisioning computing resources of Amazon EC2. The experiment results show that the provisioning plan made by our model has the minimum cost compared with plans made by the other capacity planning methods. The experimental results also show only using spot instances in short-term planning has little effect on reducing the provisioning cost.

REFERENCES

[1] A. Stage, T. Setzer and M. Bicher, "Automated Capacity Management and Selection Infrastructure-as-a-service Providers," in *IFIP/IEEE Int. Symposium on Itegrated Network Management Workshops*, 2009.

[2] A. Andzrejak, D. Kondo, and S. Yi, "Decision Model for Cloud Computing under SLA Constraints," in *IEEE Int. Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOT)*, 2010.

[3] S. Yi, D. Kondo, and A. Andrejak, "Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud," in I*EEE 3rd Int. Conference on Cloud Computing (CLOUD)*, 2010.

[4] S. Chaisiri, R. Kaewpuang, B. Lee and D. Niyato, "Cost Minimization for Provisioning Virtual Servers in Amazon Elastic Compute Cloud," in *19th Annual IEEE Int. Symposium on Modelling,z Analysis, and Simulation of Computer and Telecommunication Systems*, 2011.

[5] R. Lopes, F. Brasileiro, and P. D. Maciel Jr., "Business-Driven Capacity Planning of a Cloud-based IT Infrastructure for the Execution of Web Application," in *IEEE Int. Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, 2010.

[6] "Amazon EC2 spot instances," http://aws.amazon.com/ec2/spot-instances/.

[7] O. A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster and D. Tsafrir, "Deconstructing Amazon EC2 Spot Instance Pricing," in *3rd IEEE Int. Conference on Cloud Computing Technology and Science*, 2011.

[8] T. Lossen, "Cloud exchange," http://cloudexchange.org/.

[9] O. A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster and D. Tsafrir, "Deconstructing Amazon EC2 Spot Instance Pricing," in *3rd IEEE Int. Conference on Cloud Computing Technology and Science*, 2011.

[10] GAMS Solvers, http://www.gams.com/solvers/index.htm.

[11] J. L. L. Simarro, R. Moreno-Voemediano, R. S. Montero and I. M. Llorente, "Dynamic Placement of Virtual Machines for Cost Optimization in Multi-Cloud Environment," in *IEEE Int. Conference on High Performance Computing and Simulation (HPCS)*, 2011.