The Impact of User Rationality in Federated Clouds

Marian Mihailescu, Yong Meng Teo
Department of Computer Science
National University of Singapore
Computing 1, 13 Computing Drive, Singapore 117417
Email: [marianmi, teoym]@comp.nus.edu.sg

Abstract

With cloud computing, the long-envisioned dream of computing as utility is achieved. Many of the current standalone cloud providers offer resources and services using pay-per-use fixed pricing. Spot pricing, recently introduced in Amazon EC2, is more efficient by setting the resource price dynamically, based on demand. However, this pricing scheme, similar to the uniform price auction, is truthful only when supply can be adjusted, such as the case of a standalone cloud provider. In a federated cloud, where resources from multiple cloud providers are integrated to increase elasticity and reliability, rational users can have a greater impact. In this paper, we evaluate the impact of rationality in a federated cloud, by comparing the consumer welfare achieved by spot pricing, currently used in Amazon EC2, and a strategy-proof pricing scheme. We consider different EC2 regions as providers in a federated cloud, and use traces of spot prices to determine the consumer requests and the user welfare. Our results show that spot pricing is not suitable in a federated cloud, where rational users can increase their welfare by being untruthful.

1. Introduction

Cloud computing is an emerging platform that promises to achieve the long-envisioned dream of computing as utility. Currently, cloud consumers can use resources and services from public clouds over the Internet at pay-per-use price rates. On the other hand, private clouds are built to offer similar resources and services as public clouds, in an enterprise. Having the means for a private cloud to temporarily use resource from a public cloud as a part of an elastic resource capacity strategy can be critical in scaling enterprise cloud resources [5], [20]. This is achieved by hybrid or federated clouds, a recent paradigm that aims to integrate private and public clouds to increase elasticity and reliability [7], [14].

In contrast to other resource sharing systems used in research or academic communities, cloud computing is used commercially and its economic model is based on pricing. Currently, an active research topic in cloud computing is its economic aspect, including the problem of pricing cloud resources and services [8].

Traditionally, cloud providers specify a fixed price for each resource type or service they offer. However, this type of pricing is not efficient as it does not adapt to the changes in demand and supply [15]. For example, when demand becomes lower than supply, providers can decrease the price to incentivize consumers to use more resources. This issue is more severe in a federated cloud, where demand and supply is unpredictable with a larger number of providers and consumers. Moreover, having a market with multiple providers and consumers leads to participants making rational decisions in order to maximize their benefit [11]. To address some of these issues,

A version of this paper is published in the Proceedings of 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 620–627, Ottawa, Canada, May 13–16, 2012.

Amazon [1] introduced *spot pricing*, a market-driven resource allocation mechanism that sets the resource price according to consumer demand. Spot pricing is a scheme similar to the uniform price auction, where consumers bid the maximum price they are willing to pay for the resource, also called the reserved price, but pay the spot price, usually determined as the lowest winning bid. It was shown that if the supply can be adjusted, such as the case of a single provider, this type of auction is truthful and can prevent consumer collusion [10].

In this paper, we study the impact of user rationality in the context of a federated cloud marketplace. We measure user rationality using the cloud consumer welfare, determined as the difference between the consumer reserved price and the provider payment. Our study considers a federated cloud that consists of different Amazon EC2 regions. We analyze the spot pricing scheme used by Amazon EC2 to provide consumers with virtual machines called *instances*, and compare it with a strategy-proof dynamic pricing mechanism we propose to use in federated clouds [23]. Our main contributions are: i) a study of the rationality of cloud computing users, and ii) a model for determining EC2 workload using the spot pricing history.

The remainder of this paper is organized as follows. In Section 2, we describe our model for determining consumer spot requests and the user welfare in Amazon EC2. Next, in Section 3, we present our results and analysis for an example federated cloud consisting of four EC2 regions. Section 4 contains an overview of related works, and Section 5 concludes the paper.

2. Evaluation Model

Cloud infrastructure services, also known as *Infrastructure as a Service* (IaaS), deliver computer infrastructure using a virtualized environment to cloud consumers over the Internet. In IaaS, providers offer different instance types, typically using fixed pricing, such that strategic user behavior is limited. Each instance type approximates a different virtualized hardware configuration. For example, a large Amazon EC2 instance, *m1.large*, is the equivalent of a 64-bit system with 7.5 GB of memory, 4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units, each equivalent to a 1GHz CPU), and 850 GB of storage. In addition to fixed pricing, Amazon started using spot pricing to increase consumer demand for the EC2 service.

In spot pricing, cloud consumers specify bids that represent their reserved price, i.e. the maximum hourly price they are willing to pay for an instance type. A spot request may contain several instances, all of the same instance type. The spot price is determined by Amazon dynamically, based on the current queue of spot requests. When the current spot price is lower than the consumer bid, the respective spot request is dequeued and allocated, and the instances are started. After each hour, the spot price is updated, and, if the current spot price becomes higher than the consumer bid price, the instances allocated for the consumer spot request are terminated. The cloud consumer can specify a “persistent” flag for the spot request, such that after termination the request will be put back in the queue and restarted when the spot price becomes lower than the consumer bid price.

In our previous work [15], we have shown that fixed pricing does not achieve good economic efficiency in a system with multiple providers, or where users are both providers and consumers of resources, because it cannot adapt to the changes in demand and supply. Accordingly, we have proposed a strategy-proof dynamic pricing scheme, where payments for both consumers and providers contain incentives for them to be truthful. Our proposed scheme achieved a higher percentage of allocated requests and a higher user welfare, when compared to fixed pricing [23]. Its economic properties, such as the allocation of consumer requests for multiple resource types, make it suitable in a federated cloud, where cloud resources from different providers are integrated for increased elasticity and reliability.

2.1. Approach

In contrast to our strategy-proof scheme, spot pricing is a truthful mechanism only in a market with a single provider [10]. Accordingly, in a federated cloud with multiple providers, rational users may become an issue when spot pricing is used. Our evaluation determines the impact of rationality by comparing the consumer welfare

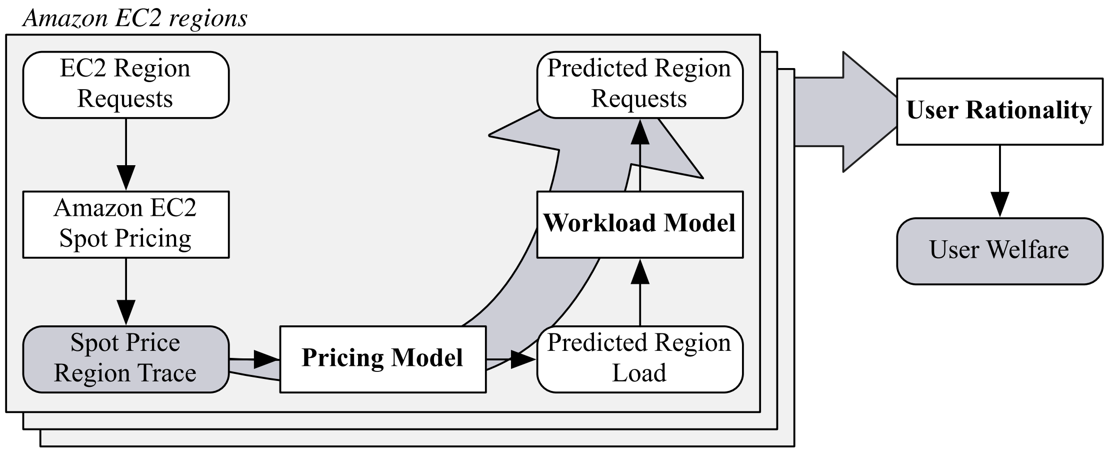


Figure 1: Proposed Approach

obtained using the two pricing schemes in a federated cloud containing different Amazon EC2 regions. We use historical data of EC2 spot prices to determine the welfare as the difference between the consumer reserved price, r, and the resource price, p:

$$w = r - p$$

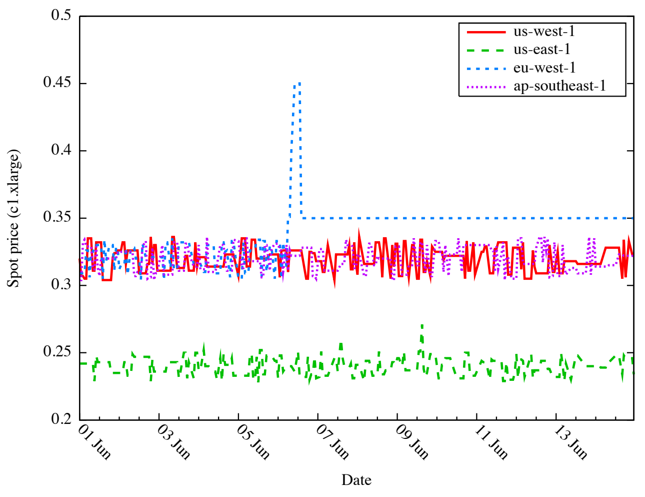
We use the EC2 spot price traces available at CloudExchange [2] for four EC2 regions: *us-east-1* (Northern Virginia), *us-west-1* (Northern California), *eu-west-1* (Ireland), and *ap-southeast-1* (Singapore). When using the spot pricing scheme, requests are allocated to the region geographically closest to the consumer, whereas in the case of our strategy-proof pricing scheme, requests are allocated dynamically to any EC2 region. We focus in our study on computational-intensive instance types (*c1.medium, c1.xlarge*), with the assumption that data availability and the latency of instances allocated in other regions are not an issue for the consumers.

According to Amazon, the spot price is determined periodically by the load of EC2 [1]. In the absence of any other public information provided by Amazon about the infrastructure or workload in EC2, we propose a bottom-up approach to predict the consumer welfare. Our approach, shown in Figure 1, contains three steps. In the first step, we use a *pricing model* to predict the hourly load of an EC2 region, using the spot price history. In the second step, we use a *workload model* to generate consumer spot requests that produce the load predicted in the first step. In the last step, we create a request queue for all the EC2 regions, and measure the consumer welfare obtained when allocating using the spot pricing scheme, and the strategy-proof scheme.

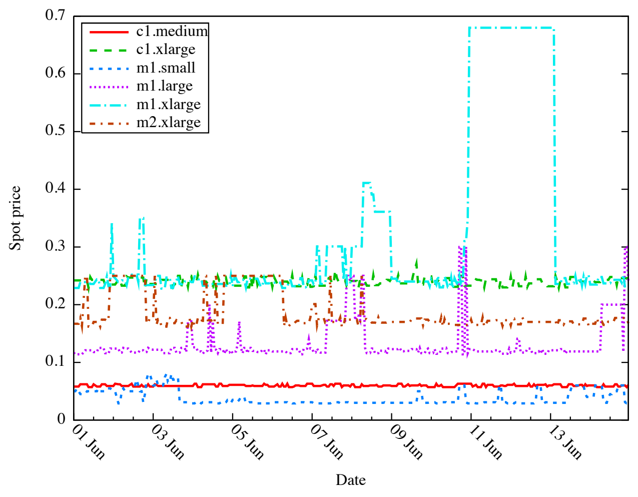
2.2. Pricing Model

In this section, we present our proposed pricing model used to predict the load of an Amazon EC2 region, using the spot price traces. Our model is based on three observations determined by the examination of the EC2 spot prices.

Figure 2(a) shows the spot prices of the same instance type, *c1.xlarge*, across different EC2 regions in the first weeks of June 2010. As can be seen from the figure, spot prices have spikes when the demand increases. However, these spikes are not correlated between different regions. For example, on June 6, there is a steep increase in the spot price in the *eu-west-1* region, but a decrease in the region *us-east-1*. Accordingly, our first observation is that *each region computes spot prices independently*, based on the respective region load. Thus, we can consider in our evaluation that each EC2 region is acting as a different resource provider in a federated cloud.



(a) in different EC2 regions for the same instance type



(b) in the same EC2 region for different instance types

Figure 2: Amazon EC2 Spot Prices

In Figure 2(b), we plot the spot prices for different resource types within the same EC2 region, *us-east-1*, for the same period of time as above. An examination of the data shows that, within the same region, there are spikes in the spot price for different instance types, according to the demand for the respective resource type. However, the spikes are not correlated among different resource types. For example, on June 4, there is a steep increase in the spot price of the *m1.large* instance type, and at the same time a steep decrease for the *m1.small* instance type. Accordingly, our second observation is that *each region has a different pool of resources for each instance type*. Based on this observation, we can analyze different instance types independently.

The plots in Figure 2 shows us that spot prices are usually within a small range, with spikes when the demand increases. Our last observation is that *each instance type in each region has a minimum reserved price (r), and a maximum spot price (p_f)*. Although spot prices can increase for short periods of time to values higher than the fixed instance price, we use, for simplicity, the fixed price as the maximum spot price.

Based on above observations and assumptions, we propose to use a power function to estimate the load of each EC2 region from the trace data. A power function can be used to explain the spikes in the spot price variations observed, and to help in determining the user rationality.
Accordingly, our model uses the following function for spot pricing:

$$spot\_price = r + (p_f - r) * load^k \tag{1}$$

where k is a parameter that determines the gradient of the increase in spot price relative to load.

2.3. Workload Model

Using the pricing model outlined above, we can use the spot prices to predict the load of the EC2 regions, for each instance type. In this section, we present our workload model, which determines the consumer requests that generate the respective load. There are two issues that we need to address. The first issue is determining the size of the EC2 region instance pool, for each instance type. The size of the instance pool is used to compute the total number of instances running in an EC2 region. The second issue is, given the total number of instances running, how to determine the number of consumer requests and the number of instances in a request.

Amazon does not provide any historical information about the number of instances running in EC2. However, it has been found that the unique identifier assigned to an EC2 instance at the time it starts running can be used to determine the total number of virtual machines started in the Amazon cloud until that point in time [3].

Accordingly, using the identifiers of instances started at different dates, it is possible to estimate the number of instances started in that interval of time. In our experiments, we look at the spot prices from June 2010, for which there are available the traces with the spot prices [2], the number of instances started daily, and the distribution of instance types in the *us-east* region [19]. We consider the distribution of instance types in the other EC2 regions similar to *us-east*, and the size of the regions proportional to the number of public IP addresses from each region, published by Amazon.

Using the number of instances started daily from each instance type, our workload model determines the number of consumer spot requests, the number of instances for each request, and the request arrival time, as follows. Firstly, we determine the number of instances in a request, which we assume it follows an exponential distribution. We are using the exponential distribution because it is memoryless and has the largest entropy. The distribution mean is computed by fitting the number of CPUs requested in parallel workload traces from production systems. Specifically, in our experiments, we use the traces of jobs that run on HPC clusters, available at the Parallel Workload Archive [18]. Secondly, we start to generate requests for each instance type in an EC2 region, containing a number of instances according to the exponential distribution determined above. We consider all the consumer requests with the total number of instances similar to the number of instances started hourly in the Amazon region from the EC2 trace. Lastly, we generate the request inter-arrival time by considering the arrival of requests a Poisson process and using a Poisson distribution with the mean of $(3600 - N)$ seconds, where N is the number of hourly requests determined in the previous step.

2.4. Validation

Spot prices are expressed by discrete values, in cents (¢), with maximum one decimal point. Accordingly, each discrete spot price corresponds to a range load values, with the same spot price for each range. For example, for the *m1.small* instance type in the *us-east-1* EC2 region, the minimum price is $r = 2.5$¢, and the fixed price is $p_f = 8.5$¢. Using the parameter $k = 4$, we determine the spot price $p = 2.5$¢ for the load range $0 - 36\%$, $p = 2.8$¢ for the range $47 - 51\%$, etc. As the load increases, the ranges become smaller and there is a steeper increase of the spot price, similar to the spikes observed in the trace data plotted in Figure 2. The value for the parameter k was determined experimentally, using trial and error. We have validated our model by comparing the spot prices from the EC2 trace with the spot prices generated using Equation 1. Figure 3 shows the spot price for the *c1.xlarge* instance type from the trace using a dotted line, and the spot price generated using our model using a continuous line. It can be seen that our model follows closely the trace values. As an example, we have

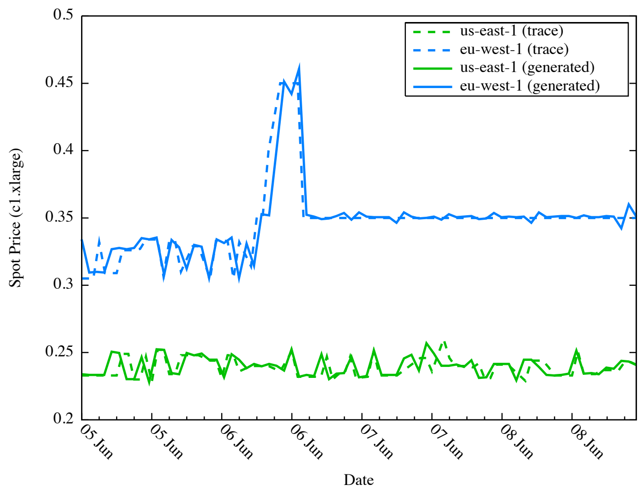


Figure 3: Spot Price Validation

plotted only the *us-east-1* and *eu-west-1* regions, between June 5–8; however a similar trend is observed for all regions in the observed time interval.

3. Results and Analysis

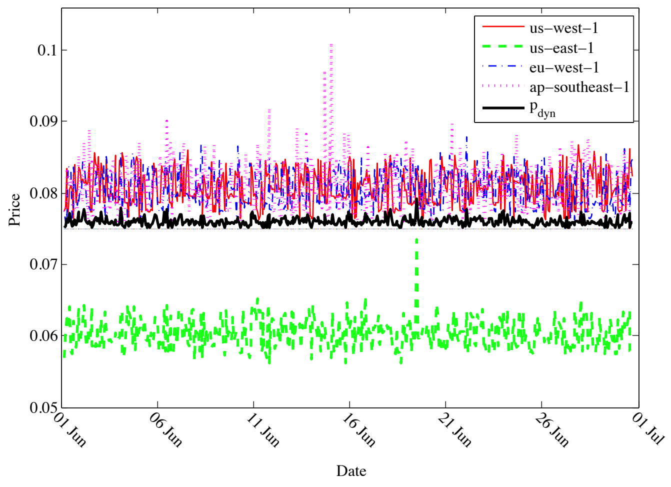
From the trace data, we can observe that the *us-west*, *eu-west* and *ap-southeast* regions are similar with respect to both the reserved price, r, and the fixed price, p_f. For example, for the instance type *c1.xlarge*, the reserved price is 30.4¢ and the fixed price is 76¢ in all three regions. However, for the *us-east* region, both the reserved price and the fixed price are different than in the other regions, for all instance types. For example, the reserved price for the *c1.xlarge* instance type is 22.8¢ and the fixed price is 60¢. Using r, p_f, and the spot prices from the traces, $spot\_price$, we determine the hourly load ranges using the function in Equation 1, as follows:

$$load_{min} = \sqrt[k]{\frac{spot\_price - r}{p_f - r}} * 100 \quad (2)$$

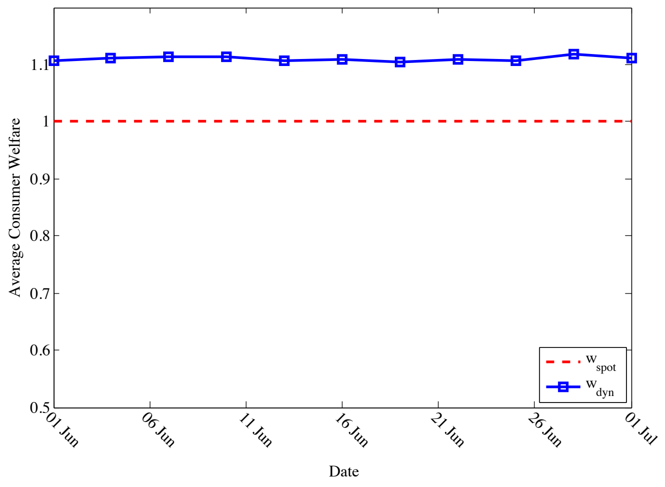
$$load_{max} = \sqrt[k]{\frac{spot\_price + 0.1 - p_0}{p_f - r}} * 100 \quad (3)$$

Next, we estimate the number of instances of each instance type running each hour, based on the total number of hourly running instances, the load range, $(load_{min} - load_{max})$, and the distribution between the instance types. According to [19], in June 2010, there were 9% *c1.medium* instances and 28% *c1.xlarge* instances from the total number of virtual machines started on EC2. Lastly, we generate consumer requests for a number of instances according to an exponential distribution with the mean 42.2, determined by the fitting function from the parallel workload trace. We have used the *LLNL-Thunder* log, which contains several months of records from the #2 machine in the 2004 Top500 list, installed at Lawrence Livermore National Lab [18].

Our implementation uses MATLAB scripts to predict the consumer requests as detailed above, and to order the requests in all the EC2 regions into one queue, sorted by the request arrival time. We allocate the consumer requests using the two pricing schemes using the First-Come-First-Serve policy. Consumer welfare when using spot pricing, w_{spot}, and the strategy-proof pricing scheme, w_{dyn}, is determined using the following functions:



(a) Spot and Strategy-proof Prices



(b) Average Consumer Welfare

Figure 4: *c1.medium* Results

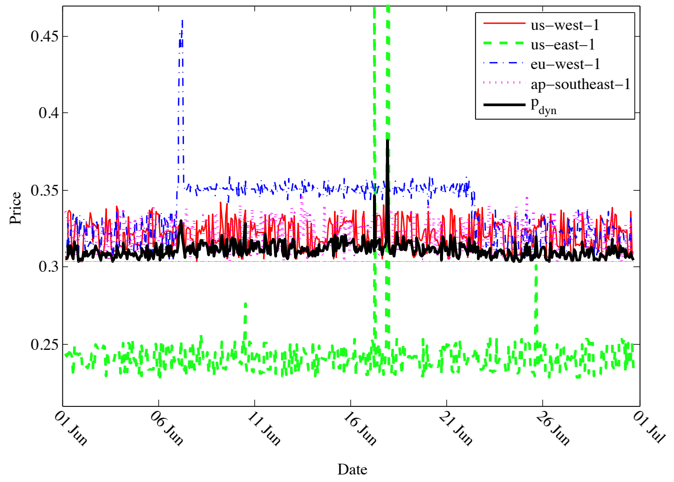
$$w_{spot} = \sum (p_f - p_{spot}) \quad (4)$$
$$w_{dyn} = \sum (p_f - p_{dyn}) \quad (5)$$

where p_{spot} is the instance price determined in a region when using spot pricing, and p_{dyn} is the price determined across regions by the strategy-proof scheme.

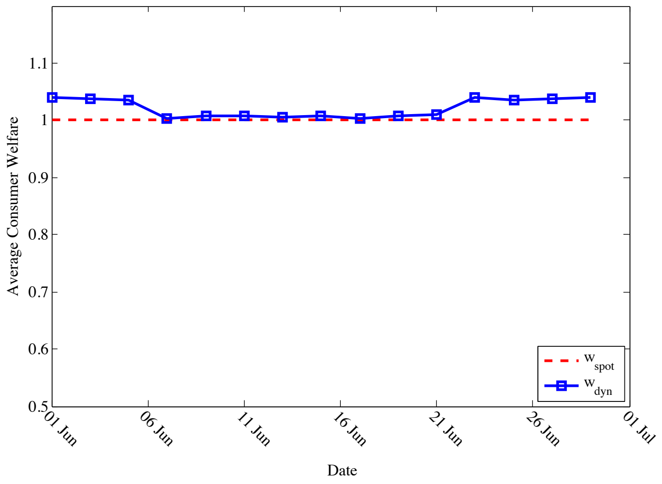
Figure 4 shows our results for the *c1.medium* instance type. In Figure 4(a), the black line represents the dynamic price determined using the strategy-proof scheme, with requests being allocated to any EC2 region. The thin lines represent the spot prices determined using the proposed spot pricing function, from Equation 1, in all of the EC2 regions, with a request being allocated only in the region where it was generated. It can be seen from the figure that the instance prices computed using the strategy-proof scheme are not lower than the reserved price of the three similar EC2 regions, due to the financial incentives required to achieve strategy-proof. When allocating consumer requests for one instance type, the strategy-proof scheme is similar to a Vickrey auction [25], where the payment is equal to the second provider price. In the case of Amazon EC2 regions, the lowest second price that can be found is the same reserved price for the three regions, *us-west-1*, *eu-west-1*, and *ap-southeast-1*.

A comparison of the normalized consumer welfare achieved by the two pricing schemes is shown in Figure 4(b). It can be seen that using the strategy-proof scheme results in a higher consumer welfare by more than 10% during the entire period. This is mostly because the strategy-proof scheme can allocate instances to different providers, according to the market price set by demand and supply. Thus, requests that generated in a region with high demand can be allocated by our scheme in a different region, such that consumer welfare is increased.

Similar to the above, we have evaluated the *c1.xlarge* instance type. Figure 5(a) shows the different prices obtained by the spot pricing scheme and the strategy-proof scheme. For the *c1.xlarge* instance type, the price determined by our strategy-proof scheme is not less than 0.075¢, the second-best reserved price. We have plotted the consumer welfare in Figure 5(b). Our results show that, for the *c1.xlarge* instance type, the average consumer welfare obtained using the strategy-proof scheme is higher by 4% in the first days and in the last week of the observed period. However, for the rest of the period, the average consumer welfare obtained using the strategy-proof scheme is only marginally higher, by 1%, to the welfare obtained using the spot pricing scheme. By correlating the welfare with the demand and prices from Figure 5(a), we can observe that the performance of the strategy-proof scheme starts to degrade beginning with day 6, when there is a high demand in the *eu-west-1*



(a) Spot and Strategy-proof Prices



(b) Average Consumer Welfare

Figure 5: *c1.xlarge* Results

region, until day 23, when the demand decreases. Moreover, in day 18, our strategy-proof scheme performs similar to spot pricing, due to a sudden spike in demand in the *us-east-1* region. Similar to our previous findings [15], the consumer welfare achieved by the strategy-proof scheme varies according to demand and supply. Thus, the consumer welfare increases when demand is lower than supply, and decreases when demand is higher.

By looking at the EC2 traces and computing the consumer welfare, we try to determine the rationality of cloud computing users. While users have found methods to decrease their costs when running instances on EC2, no strategy uses the pricing schemes available in EC2. In this context, rational behavior may include, for example, reselling reserved instances for profit, or bidding on instances in regions with lower demand and prices. Our evaluations shows that consumer welfare can be increased when users behave rational by more than 10%, even when having payments larger than the minimum spot price. We regard this as a sign that rationality is currently not a significant issue in cloud computing, with standalone providers that can adjust their supply to influence the resource market. For example, constraining supply to be lower than demand leads to a lower consumer welfare, and a better outcome for the provider. However, cloud computing usage is currently increasing both in *breadth*, i.e. the number of instance types and services offered, and in *depth*, i.e. the number of providers. With an increasing number of cloud consumers, it is expected that more providers will offer similar services. With interoperability among providers, consumers will have a choice of the same service across different clouds to improve reliability and elasticity, and rationality can become an important issue as consumers and providers strategize to improve their welfare.

4. Related Work

Users that are consuming and providing shared resources in large distributed systems are rational entities that try to maximize their own benefit, even if by doing so they abuse the system and reduce its performance [11]. There are many examples of rational user behavior in the literature, such as the free-riding phenomena in file-sharing peer-to-peer networks, where rational users affect the overall system performance by consuming more than their fair share [16]; increased waiting times in computational grids, when rational users compete for the same resources [26]; and rational users achieving higher rankings in the SETI@home project statistics, by modifying the project's software client to report false-negatives [16].

Many of the recent works in distributed systems consider the use of incentives to determine rational users to behave truthfully, according to a prescribed protocol [9], [17]. In a previous work, we proposed a strategy-proof pricing mechanism for allocating consumer requests for multiple resource types in federated clouds [15]. We based our payment scheme on Vickrey-Clarke-Groves, a well-known mechanism that incentivize rational users for being truthful, and used the mechanism design framework to prove its strategic and economic properties [23]. Other pricing schemes used in the allocation of shared resources that incentivize rational users for being truthful are based on combinatorial auctions [21], [28], or use the general equilibrium theory from economics to design market-based allocation schemes that provide mutual benefits to providers and consumers [12], [22]. In [10], the authors study uniform auctions, a mechanism similar to the spot pricing scheme used by Amazon, in the context of multiple spot markets. Their work considers the provider point of view, and focuses on maximizing provider revenue. A similar goal motivates the work presented in [24], where the authors propose new policies that increase utilization and profit for IaaS cloud providers. In contrast, our work considers the consumer perspective, and looks at the limitations of spot pricing in the context of a federated cloud, where users are rational.

Several authors have proposed leveraging spot pricing to increase the consumer welfare. This can be achieved by determining the optimal bid that can be used to achieve a budget or a deadline [4], and by making use of checkpointing to reduce overall cloud resource costs [27]. However, these solutions consider spot pricing with a standalone provider. In contrast, we study spot pricing when there are multiple providers, and we try to determine how much influence can the rationality of the consumers have in the allocation outcome. Understanding spot pricing is currently still an open problem. In [6], the authors fit EC2 spot prices using an auto-regressive process, and argue that Amazon does not consistently sets the spot price based on the consumer demand. In [13], the authors develop a statistical model that fits the spot price patterns to predict future price points for the use in stochastic scheduling algorithms. Our approach is different from the above works by trying to create a model for the spot pricing scheme in order to determine the workload behind it.

5. Conclusions

In this paper, we evaluate the rationality of Amazon EC2 users by comparing the welfare obtained using the spot pricing scheme currently used in EC2, with a strategy-proof pricing mechanism, that incentivizes truthful users. We consider four EC2 regions as different providers in a federated cloud. Our assumption is supported by the historical spot prices in different EC2 regions, which show that each region has different resource pools for each instance type, with the spot price computed dynamically based on the load in each resource pool. We propose a model to generate requests based on the traces of spot prices and the total number of instances started on EC2. In contrast to spot pricing, the strategy-proof mechanism can allocate consumer requests to any EC2 region. Accordingly, we have selected for comparison the instance types suited for compute-intensive applications, with the assumption that latency due to the geographical location of the allocated instance is not an issue for the consumer. Our study found that consumer welfare can be improved when using a strategy-proof scheme, based on the demand, by more than 10%. While rationality is currently not an issue in stand-alone clouds, our results show that in a federated cloud, users can increase their welfare by being untruthful.

Future work spans two main directions. Towards understanding rationality, we intend to create a model for rational behavior in the context of federated clouds. Currently, a rational user may represent either a provider or a consumer. However, there are new cloud brokering services available, where a broker is also rational in maximizing its interest. Towards understanding spot pricing, we intend to enhance our proposed model to be able to design consumer bidding strategies using existing spot price history.

Acknowledgments

We would like to thank Dr. Alexandru Iosup for the helpful feedback and comments in reviewing this paper.

References

[1] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/.

[2] Cloud Exchange. http://cloudexchange.org/.

[3] Anatomy of an Amazon EC2 Resource ID. http://www.jackofallclouds.com/2009/09/anatomy-of-an-amazon-ec2-resource-id/.

[4] A. Andrzejak, D. Kondo, and S. Yi. Decision Model for Cloud Computing under SLA Constraints. In *Proc. of the 18th Annual IEEE/ACM Intl. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2010)*, pages 257–266, Miami, USA, Aug. 2010.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb. 2009.

[6] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. Deconstructing Amazon EC2 spot instance pricing. In *IEEE 3rd Intl. Conf. on Cloud Computing Technology and Science, CloudCom 2011*, pages 304-311, Athens, Greece, Nov. 2011.

[7] R. Buyya, R. Ranjan, and R. N. Calheiros. InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services. In *Proc. of the 10th Intl. Conf. on Algorithms and Architectures for Parallel Processing (ICA3PP 2010)*, pages 13–31, Busan, Korea, May 2010.

[8] R. Buyya, C. S. Yeo, and S. Venugopal. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In *Proc. of the 10th IEEE Intl. Conf. on High Performance Computing and Communications (HPCC 2008)*, pages 5–13, Dalian, China, Sept. 2008.

[9] B.-G. Chun, R. Fonseca, I. Stoica, and J. Kubiatowicz. Characterizing Selfishly Constructed Overlay Routing Networks. In *Proc. of the 23rd Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2004)*, pages 1329–1339, Hong Kong, China, Mar. 2004.

[10] A. Danak and S. Mannor. Resource Allocation with Supply Adjustment in Distributed Computing Systems. In *Proc. of the 30th IEEE Intl. Conf. on Distributed Computing Systems (ICDCS'10)*, pages 498–506, Genova, Italy, Jun. 2010.

[11] R. K. Dash, N. R. Jennings, and D. C. Parkes. Computational-mechanism design: A call to arms. *IEEE Intelligent Systems*, 18(6):40–47, Nov. 2003.

[12] E. R. Gomes, Q. B. Vo, and R. Kowalczyk. Pure exchange markets for resource sharing in federated clouds. *Concurrency and Computation: Practice and Experience*, 2010.

[13] B. Javadi, R. K. Thulasiram, and R. Buyya. Statistical Modeling of Spot Instance Prices in Public Cloud Environments. In *Proc. of the 4th IEEE Intl. Conf. on Utility and Cloud Computing (UCC 2011)*, page (to appear), Melbourne, Australia, Dec. 2011.

[14] K. Keahey, M. Tsugawa, A. Matsunaga, and J. A. B. Fortes. Sky Computing. *IEEE Internet Computing*, 13(5):43–51, Sept. 2009.

[15] M. Mihailescu and Y. M. Teo. Dynamic Resource Pricing on Federated Clouds. In *Proc. of the 10th IEEE/ACM Intl. Symp. on Cluster, Cloud and Grid Computing (CCGRID 2010)*, pages 513–517, Melbourne, Australia, May 2010.

[16] S. J. Nielson, S. A. Crosby, and D. S. Wallach. A Taxonomy of Rational Attacks. In *4th Intl. Workshop on Peer-to-Peer Systems (IPTPS 2005)*, volume 3640 of *Lecture Notes in Computer Science (LNCS)*, pages 36–46, Ithaca, USA, Feb. 2005.

[17] N. Nisan and A. Ronen. Algorithmic Mechanism Design. In *Proceedings of the 31st Annual ACM Symp. on Theory of Computing (STOC'99)*, pages 129–140, Atlanta, USA, May 1999.

[18] Parallel Workload Archives. LLNL Thunder Log. http://www.cs.huji.ac.il/labs/parallel/workload/.

[19] RightScale. More servers, bigger servers, longer servers, and 10x of that! http://blog.rightscale.com/2010/08/04/more-bigger-longer-servers-10x/.

[20] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Cáceres, M. Ben-Yehuda, W. Emmerich, and F. Galán. The Reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):535–545, Jul. 2009.

[21] B. Song, M. M. Hassan, and E.-N. Huh. A Novel Cloud Market Infrastructure for Trading Service. In *Proc. of the 2009 Intl. Conf. on Computational Science and Its Applications (ICCSA '09)*, pages 44–50, Washington, DC, USA, Sept. 2009.

[22] F. Teng and F. Magoulès. Resource Pricing and Equilibrium Allocation Policy in Cloud Computing. In *Proc. of the 10th IEEE Intl. Conf. on Computer and Information Technology (CIT 2010)*, pages 195–202, Bradford, UK, Jun. 2010.

[23] Y. M. Teo and M. Mihailescu. A Strategy-proof Pricing Scheme for Multiple Resource Type Allocations. In *Proc. of the 38th Intl. Conf. on Parallel Processing (ICPP'09)*, pages 172–179, Vienna, Austria, Sept. 2009.

[24] A. N. Toosi, R. N. Calheiros, R. K. Thulasiram, and R. Buyya. Resource Provisioning Policies to Increase IaaS Provider's Profit in a Federated Cloud Environment. In *Proc. of the 13th IEEE Intl. Conf. on High Performance Computing & Communication (HPCC 2011)*, pages 279–287, Banff, Canada, Sept. 2011.

[25] W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, 16(1):8–37, Mar. 1961.

[26] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan. G-commerce: Market Formulations Controlling Resource Allocation on the Computational Grid. In *Proc. of the 15th Intl. Parallel & Distributed Processing Symp. (IPDPS'01)*, pages 46–54, San Francisco, USA, Apr. 2001.

[27] S. Yi, D. Kondo, and A. Andrzejak. Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud. In *Proc. of the IEEE Intl. Conf. on Cloud Computing (CLOUD 2010)*, pages 236–243, Miami, USA, Jul. 2010.

[28] S. Zaman and D. Grosu. Combinatorial Auction-Based Allocation of Virtual Machine Instances in Clouds. In *Proc. of the Second Intl. Conf. on Cloud Computing (CloudCom 2010)*, pages 127–134, Indianapolis, USA, Nov. 2010.