Strategy-proof Mechanisms for Resource Management in Clouds

Lena Mashayekhy
Dept. Computer Science
Wayne State University
Detroit, MI 48202, USA
Email: mlena@wayne.edu

Daniel Grosu
Dept. Computer Science
Wayne State University
Detroit, MI 48202, USA
Email: dgrosu@wayne.edu

***Abstract*—The ever-growing demand for cloud resources places the resource management at the heart of the design and decision-making processes in cloud computing environments. Cloud providers offer heterogeneous resources such as CPUs, memory, and storage in the form of Virtual Machine (VM) instances. Recently, cloud providers have introduced auction-based models to sell their unutilized resources in an auction market which allow users to submit bids for their requested VMs. In this PhD dissertation, we address the problem of autonomic VM provisioning and allocation for the auction-based model considering multiple types of resources by designing exact and approximation mechanisms. The mechanisms also determine the payment the users have to pay for using the allocated resources. Furthermore, our proposed mechanisms drive the system into an equilibrium in which the users do not have incentives to manipulate the system by untruthfully reporting their VM bundle requests and valuations.**

***Keywords*-cloud computing; resource management; strategy-proof mechanism; virtual machine;**

I. Introduction

Cloud providers offer different types of services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [1]. IaaS provides CPUs, storage, networks and other low level resources, PaaS provides programming interfaces, and SaaS provides already created applications. In this paper, we focus on IaaS where cloud providers offer heterogeneous resources in the form of VM instances. The types of VM instances that a cloud provider offers are known to the users. For example, Microsoft Azure and Amazon Elastic Compute Cloud (Amazon EC2) are currently offering four types of VM instances: Small (S), Medium (M), Large (L), and Extra large (XL).

There are two approaches for VM provisioning and allocation that a cloud provider can employ: static provisioning and dynamic provisioning. In static provisioning, the cloud provider pre-provisions a set of VM instances without considering the actual demand from the users, while in dynamic provisioning the cloud provider provisions the resources taking into account the actual user demand. Due to a variable load demand, dynamically provisioning resources leads to a more efficient resource utilization and ultimately to higher revenues for the cloud provider. The aim of this paper is to design mechanisms that facilitate dynamic provisioning of cloud resources based on the user demand and the availability of resources.

Cloud providers can employ fixed-price and auction-based models to sell the VM instances to users. In the fixed-price model, the price of each type of VM instance is fixed and pre-determined by the cloud provider, while in the auction-based model, each user bids for a subset of available VM instances (bundle) and an auction mechanism decides the price and the allocation. In the auction-based mechanisms, users can obtain their requested VMs at lower prices than in the case of the fixed-price mechanisms. Also, the cloud providers can increase their profit by allowing users to bid on unutilized capacity. An example of such auction-based mechanism is the spot market introduced by Amazon.

In this paper, we focus on designing mechanisms for auction-based settings. However, our setup and mechanisms are different from the Amazon spot market, since Amazon allows only requests for individual VM instances and not for bundles of VM instances of different types. Therefore, this type of auctions do not guarantee that a user receives all requested VMs of different types together. In our setting, however, we allow users to request bundles of VM instances.

In our auction-based settings, each user can request a bundle of VM instances along with submitting a bid for it. This problem can be viewed as a multi-unit combinatorial auction. This is due to the fact that there are several VM instances of the same type available for allocation. Each user has a private value (private *type*) for her requested bundle. In our model, the users are single minded, that means each user is interested in a single bundle of VM instances, and bids only for that bundle. A single minded user obtains the specified value if she is allocated the whole bundle of VM instances (or any superset of it) and zero value, if she is allocated any other bundle. The users are also selfish in the sense that they want to maximize their own utility. It may be beneficial for them to manipulate the system by declaring a false type (i.e., different bundles or bids from their actual request).

The *objective of this Ph.D. dissertation* is to design, study and implement strategy-proof mechanisms for resource management in clouds in the presence of multiple types of

heterogeneous resources. Our goal is to design mechanisms that are efficient and lead to revenue maximization for the cloud providers. In addition, for users, a truthful mechanism eliminates the expensive overhead of strategizing about other bidders and prevents market manipulation. The *rationale* for our research is that, once efficient resource management mechanisms that consider the incentives of the users and cloud providers are developed and implemented, it will benefit both cloud providers and users.

II. Significance of our Research

The ever-growing demand for resources from businesses and individuals places the resource management at the heart of the design and decision-making processes in cloud environments. One of the major challenges faced by the cloud providers is to allocate and provision the resources such that their profit is maximized and the resources are utilized efficiently. The objective of this Ph.D. dissertation is to design, develop, and analyze mechanisms for resource management in cloud computing systems. The goal is to allocate resources efficiently while maximizing a global performance objective of the system (e.g., revenue, social welfare, or energy saving). We improve the state-of-the-art in both methodologies and applications. As for methodologies, we introduce novel resource management mechanisms based on mechanism design [2] and approximation algorithms. These mechanisms can be employed to provision and allocate VM instances in current and future cloud computing systems.

III. Research Accomplishments

In this section, we present our research accomplishments. We investigated the problem of federating resources in grids by employing coalitional game theory and designed grid federation formation mechanisms [3], [4], [5]. We also studied the problem of federating resources in grids considering the trust relationship among grid service providers [6]. We addressed the problem of federation formation in clouds and designed a coalitional game-based mechanism that enables the cloud providers to dynamically form a cloud federation maximizing their profit [7].

In this paper we focus only on two of our recent studies. In our paper presented at IEEE CLOUD 2013 [8], we formulated the problem of VM provisioning and allocation in clouds (VMPAC) and designed a family of strategy-proof greedy mechanisms, called GVMPAC-X to solve it. We proved that the approximation ratio of the mechanisms in the G-VMPAC-X family is RC_{max}, where R is the number of types of resources, and $C_{max} = \max_{r \in \mathcal{R}} C_r$, where C_r is the restricted capacity on each resource $r = 1, \ldots, R$. Since the achieved approximation ratio of the greedy mechanisms is not tight enough, we resort to approximation mechanisms that can guarantee better bounds for their obtained solutions.

In a paper presented at ACM CAC 2013 [9], we designed an optimal strategy-proof mechanism, called VCG-VMPAC. In addition, we designed a PTAS (Polynomial-Time Approximation Scheme) strategy-proof mechanism, called PTAS-VMPAC, that guarantees near optimal solutions. We proved that the solution determined by PTAS-VMPAC is in a $(1-\epsilon)$ neighborhood of the optimal, and that the time complexity of the algorithm is polynomial in the number of requests, N.

Our proposed mechanisms consist of an *allocation algorithm* that selects the users, and a *payment function* that determines the amount that each selected user needs to pay to the cloud provider.

Strategy-proof mechanisms drive the system into an equilibrium [2]. We proved that our proposed mechanisms in the G-VMPAC-X family, VCG-VMPAC, and PTAS-VMPAC are strategy-proof mechanisms, that is, the users do not have incentives to lie about their requested bundles of VM instances and their valuations.

The proposed mechanisms allow dynamic provisioning of VMs, and do not require pre-provisioning the VMs. As a result, cloud providers can fulfill dynamic market demands efficiently. A key property of our proposed mechanisms is the consideration of multiple types of heterogeneous resources for VMs, which is the case in real cloud settings.

We consider a cloud provider offering R types of resources, $\mathcal{R} = \{1, \ldots, R\}$, to users in the form of VM instances. These types of resources include cores, memory, storage, etc. The cloud provider has restricted capacity, C_r, on each resource $r \in \mathcal{R}$ available for allocation. The cloud provider offers these resources in the form of M types of VMs, $\mathcal{VM} = \{1, \ldots, M\}$, where each VM of type $m \in \mathcal{VM}$ provides a specific amount of each type of resource $r \in \mathcal{R}$. The amount of resources of type r that one VM instance of type m provides is denoted by w_{mr}. As an example, we consider that CPU represents the type 1 resource, memory the type 2 resource, and storage the type 3 resource. We can characterize a possible VM instance (of type $m = 1$) by: $w_{11} = 1$ core, $w_{12} = 1.6$ GB, and $w_{13} = 150$ GB.

We consider a set $\mathcal{U}$ of N users requesting a set of VM instances. User i, $i = 1, \ldots, N$, requests a bundle $S_i =< k_{i1}, k_{i2}, \ldots, k_{iM} >$ of M types of VM instances, where k_{im} is the number of requested VM instances of type $m \in \mathcal{VM}$. In addition, she specifies a bid b_i for her requested bundle S_i. User i values her requested bundle S_i at $v_i(S_i)$, where $v_i(S_i)$ is called the *valuation* of user i for bundle S_i. The valuation represents the maximum price a user is willing to pay for using the requested bundle for a unit of time. Each user can submit her request as a vector specifying the number of VM instances, and her bid. For example, $(< 1, 3, 4, 2 >, \$10)$ represents a user requesting 1 small VM instance, 3 medium VM instances, 4 large VM instances, and 2 extra large VM instances, and her bid is \$10. We denote by V the *social*

welfare, which is defined as the sum of users' valuations:

$$V = \sum_{i \in \mathcal{U}} v_i(S_i) \cdot x_i \tag{1}$$

where x_i, $i = 1, \ldots, N$, are indicator variables defined as follows: $x_i = 1$ if bundle S_i is allocated to user i, and $x_i = 0$, otherwise. To design strategy-proof mechanisms, we consider the standard mechanism design objective, that is, maximizing the social welfare. Maximizing social welfare can help a cloud provider increase its revenue by allocating the VMs to the users who value them the most.

We formulate the problem of VM provisioning and allocation in clouds (VMPAC) as an Integer Program (called VMPAC-IP) as follows:

$$\text{Maximize } V \tag{2}$$

Subject to:

$$\sum_{i \in \mathcal{U}} \sum_{m \in \mathcal{VM}} k_{im} w_{mr} x_i \leq C_r, \forall r \in \mathcal{R} \tag{3}$$

$$x_i = \{0, 1\}, \forall i \in \mathcal{U} \tag{4}$$

The solution to this problem is a vector $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ maximizing the social welfare. Constraints (3) ensure that the allocation of each resource type does not exceed the available capacity of that resource. Constraints (4) represent the integrality requirements for the decision variables. These constraints force the cloud provider to provision the whole bundle of VM instances and to allocate bundles to the selected users. The VMPAC problem is equivalent to the multidimensional knapsack problem (MKP) [10], where the knapsack constraints are the resource capacity constraints and the bundles are the items. The objective is to select a subset of items for the multidimensional knapsack maximizing the total value. As a result, the VMPAC problem is strongly NP-hard.

Each user i has a *quasi-linear utility function* defined as the difference between her valuation and payment, $u_i = v_i(S_i) - \mathcal{P}_i$, where S_i is the allocated bundle to user i based on the allocation algorithm, and $\mathcal{P}_i$ is the payment for user i that the mechanism calculates based on the payment function.

The allocation algorithms of our proposed G-VMPAC-X mechanisms order the users according to a general *efficiency* metric defined as:

$$e_i = \frac{v_i}{\sum_{r=1}^{R} f_r a_{ir}}, \forall i \in \mathcal{U} \tag{5}$$

where $a_{ir} = \sum_{m \in \mathcal{VM}} k_{im} w_{mr}$ is the amount of each resource of type r requested by user i, and f_r is the *relevance factor* characterizing the scarcity of resources of type r. A higher f_r means a higher scarcity of resource r, thus, a lower efficiency. That means, a user that requests more resources of a scarce type is less likely to receive her requested bundle.

The choice of relevance values, f_r, defines the members of the G-VMPAC-X family of allocation algorithms. We consider two choices for f_r and obtain two allocation algorithms, G-VMPAC-I and G-VMPAC-II as follows:

1) G-VMPAC-I: obtained when $f_r = 1$, $\forall r \in \mathcal{R}$. This is a direct generalization of the one-dimensional case. This generalization does not take into account the scarcity of different resources and may not work well in situations in which the VM instances are highly heterogeneous in terms of the resources provided.

2) G-VMPAC-II: obtained when $f_r = \frac{1}{C_r}$, $\forall r \in \mathcal{R}$. This addresses the scarcity issues in G-VMPAC-I, by scaling the values of f_r with the inverse of capacity C_r.

The allocation algorithms of the VCG-VMPAC and the PTAS-VMPAC mechanisms are based on dynamic programming. The allocation algorithm of the proposed PTAS-VMPAC mechanism iterates over all subsets users with at most q users. For each such subset the algorithm finds a feasible partial allocation, determines the amount of partially allocated resources for each of the r types of resources and rounds the amount of requested resources by the unallocated users for each of the r resources. Then, it uses a dynamic programming approach to find an allocation of bundles based on the rounded requests, and the remaining unallocated capacities. The algorithm determines the maximum welfare and the corresponding VM instance allocation obtained over all iterations.

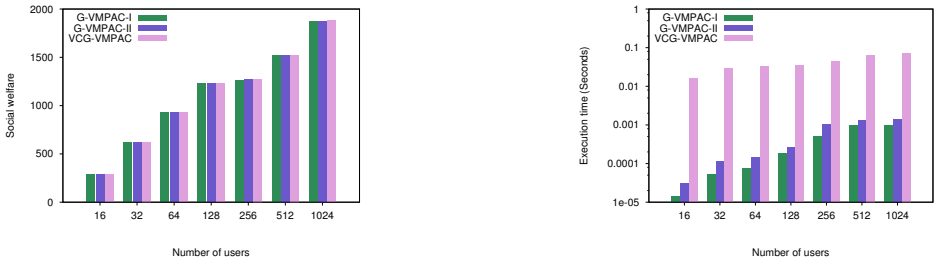
The payment functions are designed to find the critical payment [2] for each user, that is, the minimum value that if she reports, she can receive her requested bundle of VMs.

We investigate the strategy-proofness of our proposed mechanisms by analyzing the effects of untruthful declarations by a user. To show that our proposed mechanisms are robust against manipulation by a user, we consider three users requesting homogeneous VMs where their true types are $(< 5, 0, 0, 0 >, \$10)$, $(< 0, 4, 0, 0 >, \$24)$, and $(< 2, 0, 0, 2 >, \$20)$, respectively. The capacity of the three resources are as follows: 30 cores, 80 GB of memory, and 6000 GB of storage. The G-VMPAC-II calculates the efficiency of the users as 24.61, 32.98, and 11.59, respectively, then allocates resources to user 1 and 2 in the case that all users declare their true types. The payments of the winning users based on their critical payments are $\$4.71$ and $\$8.43$, respectively.

We assume that user 2 lies about her type. The consequence of such a declaration depends on her reported value v_2 and the bundle S_2. We consider different scenarios as shown in Table I, where user 2 does not reveal her true

Table I: Different scenarios for user 2's type declaration

Case	S_2	v_2	Scenario	Stat.	Pay.	Utility
I	$< 0, 4, 0, 0 >$	\$24	$\hat{v}_2 = v_2$, $\hat{S}_2 = S_2$	W	8.43	15.57
II	$< 0, 4, 0, 0 >$	\$30	$\hat{v}_2 > v_2$, $\hat{S}_2 = S_2$	W	8.43	15.57
III	$< 0, 4, 0, 0 >$	\$20	$\hat{v}_2 < v_2$, $\hat{S}_2 = S_2$	W	8.43	15.57
IV	$< 0, 4, 0, 0 >$	\$8	$\hat{v}_2 < v_2$, $\hat{S}_2 = S_2$	L	0	0
V	$< 1, 4, 0, 0 >$	\$24	$\hat{v}_2 = v_2$, $\hat{S}_2 > S_2$	W	9.38	14.61
VI	$< 0, 4, 0, 2 >$	\$24	$\hat{v}_2 = v_2$, $\hat{S}_2 > S_2$	L	0	0



(a) (b)

Figure 1: G-VMPAC-X performance: (a) Social welfare; (b) Execution time.

type. We present two of there scenarios here, and the reader is referred to [8] for a full discussion. Case I is when the user declares her true type. In case II, when user 2 reports a value greater than her true type, she will still win and the mechanism determines the same payment for her as in case I. In all cases, the user can not increase her utility by declaring a type other than her true type.

We performed extensive experiments in order to investigate the properties of the mechanisms in the G-VMPAC-X family, VCG-VMPAC and PTAS-VMPAC mechanisms. Due to space limitations, we will not present all the results here. The reader is referred to our papers [8], [9] for a full description of the experiments and results. We generate VM instance requests corresponding to systems with 16 to 1024 users. The number of VM instances and resource types offered by the cloud provider are the same in all the experiments. The generated requests are based on realistic data combining publicly available information provided by Amazon EC2 and Microsoft Azure.

Fig. 1a shows the social welfare for different number of users. The results show that different versions of G-VMPAC-X can obtain almost the same social welfare as the optimal social welfare (obtained by VCG-VMPAC). Fig. 1b shows the execution time for cases with different number of users on a logarithmic scale. From all the above results, we conclude that G-VMPAC-II finds near-optimal solutions to the VMPAC problem and requires small execution times.

IV. Future Work

In our previous studies, we proposed strategy-proof mechanisms for resource management in clouds in periodic-time settings. In order to complete the dissertation, we will finalize our on going research on designing online mechanisms. The online mechanisms make no assumptions about future demand and supply of VMs, which is the case in real cloud settings. They calculate the allocation and payment as users arrive at the system and place requests. Therefore, the cloud provider provisions and allocates VM instances as the resources become available. Another direction of our on going research is focused on designing energy-aware mechanisms for clouds. Such mechanisms decide how to provision VMs in as few physical machines as possible in order to save energy by powering them on or off.

There are several challenges that we need to address in order to complete the above mentioned research. First, in the above-mentioned settings, the problem becomes more challenging from the computational hardness standpoint, thus we will resort to approximate and heuristic approaches for determining the provisioning and allocation. Second, we must guarantee the strategy-proofness of the mechanisms, which is difficult to achieve when designing approximation algorithms.

Acknowledgment. This research was supported, in part, by NSF grants DGE-0654014 and CNS-1116787.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Comp. Sys.*, vol. 25, no. 6, pp. 599–616, 2009.

[2] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press, 2007.

[3] L. Mashayekhy and D. Grosu, "A merge-and-split mechanism for dynamic virtual organization formation in grids," in *Proc. 30th IEEE Intl. Conf. on Performance Computing and Communications Conference*, 2011, pp. 1–8.

[4] ——, "A merge-and-split mechanism for dynamic virtual organization formation in grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 540–549, 2014.

[5] ——, "A distributed merge-and-split mechanism for dynamic virtual organization formation in grids," in *Proc. 11th IEEE Intl. Conf. on Network Computing and Applications*, 2012, pp. 36–43.

[6] ——, "A reputation-based mechanism for dynamic virtual organization formation in grids," in *Proc. 41st IEEE Intl. Conf. on Parallel Processing*, 2012, pp. 108–117.

[7] ——, "A coalitional game-based mechanism for forming cloud federations," in *Proc. of the 5th IEEE Intl. Conf. on Utility and Cloud Computing*, 2012, pp. 223–227.

[8] M. M. Nejad, L. Mashayekhy, and D. Grosu, "A family of truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds," in *Proc. of the 6th IEEE Intl. Conf. on Cloud Computing*, 2013, pp. 188–195.

[9] L. Mashayekhy, M. M. Nejad, and D. Grosu, "A truthful approximation mechanism for autonomic virtual machine provisioning and allocation in clouds," in *Proc. of the ACM Cloud and Autonomic Computing Conf.*, 2013, pp. 1–10.

[10] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.