A Coalitional Game-Based Mechanism for Forming Cloud Federations

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

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

***Abstract*—We model the cloud federation formation problem using concepts from coalitional game theory by considering the cooperation of the cloud providers in providing the requested VM instances. We design a mechanism that enables the cloud providers to dynamically form a cloud federation maximizing their profit. Furthermore, the mechanism guarantees that the cloud federation structure is stable, that is, the cloud providers do not have incentives to break away from the current federation and join some other federation.**

I. Introduction

In this paper, we consider the IaaS offering by a federation of cloud providers. A *cloud federation* is a collection of cloud providers that cooperate in order to provide the resources requested by users [1]. Cloud providers offer IaaS using virtualization of low level resources. Cloud providers provision their resources into different types of virtual machine (VM) instances. We model the cloud federation formation as a coalitional game where cloud providers decide to form a coalition (cloud federation) to allocate VMs dynamically based on users' requests.

We focus on designing a mechanism for solving the cloud federation formation problem. The mechanism allows the cloud providers to make their own decisions to form a federation yielding the highest total profit. In this mechanism, coalitions of cloud providers decide to merge and split in order to form a federation providing requested resources as a service to the user. The mechanism also determines the individual profit of each participating cloud provider in the federation. Each cloud provider covers its incurred costs, and receives its individual profit based on its market power. The mechanism provides a stable federation structure, that is, none of the cloud providers has incentives to merge to another federation or split from a federation to form another federation. We analyze the properties of our proposed cloud federation mechanism and perform extensive simulation experiments to investigate its properties.

Related Work. The primary requirements for forming federations of cloud providers are discussed by Rochwerger *et al.* [1]. Goiri *et al.* [2] provided models that assist the cloud providers in making decisions on forming cloud federations. A game theoretic solution for dynamic resource allocation in a cloud federation was proposed by Hassan *et al.* [3]. The authors defined a price function for a cloud provider that gives incentives to other clouds to contribute resources and to form a federation. A revenue sharing mechanism for multiple cloud providers using stochastic linear programming games was proposed by Niyato *et al.* [4]. Coalitional games have been used in many fields where cooperation is important. Saad *et al.* [5] proposed a merge-and-split coalition formation mechanism in wireless networks. Their proposed mechanism partitions the network of antenna devices into coalitions maximizing their utilities. A mechanism for dynamic virtual organization formation in grids based on coalitional game theory was proposed by Mashayekhy and Grosu [6]. The mechanism considers the incentives of the grid service providers while providing the required capabilities to execute the user application. In this paper, we target VM allocation in federated clouds and not the allocation of jobs to grid service providers which was the focus of [6]. We also employ a new method for profit division among cloud providers instead of the equal share method used in [6].

II. Cloud Federation Formation Framework

System Model. We first describe the system model which considers a set of cloud providers, a set of brokers as mediators, and several cloud customers. We assume that a set of cloud providers $\mathcal{I} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m\}$ is available to provide resources in the form of VM instances to cloud users. The cloud providers offer n types of VM instances: $\mathcal{VM} = \{VM_1, \ldots, VM_n\}$, where each instance provides a specific number of cores, amount of memory, and amount of storage. The VM instance of type VM_j $(j = 1, \ldots, n)$ is characterized by w_j^c, the number of cores, and by w_j^s, the amount of storage provided. The amount of memory is proportional to the number of cores. We assume that all cloud providers offer the same types of VM instances.

Each cloud provider $\mathcal{C}_i \in \mathcal{I}$ has a specific number of cores and storage available. We denote by N_i, the number of available cores of cloud provider $\mathcal{C}_i$, and by S_i, the amount of available storage of cloud provider $\mathcal{C}_i$. Each provider $\mathcal{C}_i$ incurs cost when providing resources. For a cloud provider $\mathcal{C}_i$, we denote by c_{ij}^c, the cost associated with each core of VM instance of type VM_j, and by c_{ij}^s, the cost associated with each GB of storage of each VM of type VM_j, where $j = 1, \ldots, n$.

The cost of memory is included in the cost of the cores. We use different costs for one core in different VM instances since it is the most general case and prices employed by the current cloud providers reflect that [7]. However, a cloud provider bills a user based on the allocated VM instances. To do so, all cloud providers set a price p_j^c on the cores, and p_j^s on the storage of each type of VM instance VM_j, where $j = 1, \ldots, n$. As a result, from the user's point of view the way the cloud providers provide the requested VM instances does not affect the final price that she pays for her request.

A user sends a request consisting of the number of VM instances of each type needed to a broker. A request is denoted by $\mathcal{R} = \{r_1, \ldots, r_n\}$, where r_j is the number of requested VM instances of type VM_j, $j = 1, \ldots, n$. The final price paid by the user for each of the r_j VM instances is $r_j(p_j^c + p_j^s)$, where $p_j^c + p_j^s$ is a fixed price for an instance of type VM_j. A broker has all the information about cloud providers such as their available resources and associated cost, and it is responsible for forming the federation.

Cloud Federation Formation as a Coalitional Game. We model the cloud federation formation problem as a coalitional game. A *coalitional game* [8] is defined by the pair $(\mathcal{I}, v)$, where $\mathcal{I}$ is the set of players (cloud providers) and v is the *characteristic function*, defined on $\mathcal{F} \subseteq \mathcal{I}$. The characteristic function is a real-valued function such that $v : \mathcal{F} \to \mathbb{R}^+$ and $v(\emptyset) = 0$.

Each subset $\mathcal{F} \subseteq \mathcal{I}$ is a *coalition* (in our case we will call it *federation*). If all the available cloud providers form a federation, it is called the *grand federation*. A federation $\mathcal{F}$ has a *value* given by the characteristic function $v(\mathcal{F})$. Here $v(\mathcal{F})$ represents the profit obtained when the cloud providers of federation $\mathcal{F}$ cooperate as a group and is given by:

$$v(\mathcal{F}) = \sum_{\mathcal{C}_i \in \mathcal{F}} \sum_{j=1}^{n} x_{ij}^c (p_j^c - w_j^c c_{ij}^c) + x_{ij}^s (p_j^s - w_j^s c_{ij}^s), \quad (1)$$

where x_{ij}^c represents the number of VM instances of type VM_j from $\mathcal{C}_i$ providing the cores, and x_{ij}^s represents the number of VM instances of type VM_j from $\mathcal{C}_i$ providing the storage.

Since a given federation $\mathcal{F}$'s goal is to maximize its profit, we can formulate the cloud federation profit maximization problem as an integer program (IP) as follows:

$$\text{Maximize} \sum_{\mathcal{C}_i \in \mathcal{F}} \sum_{j=1}^{n} x_{ij}^c (p_j^c - w_j^c c_{ij}^c) + x_{ij}^s (p_j^s - w_j^s c_{ij}^s) \quad (2)$$

Subject to:

$$\sum_{j=1}^{n} w_j^c x_{ij}^c \le N_i, \ (\forall \mathcal{C}_i \in \mathcal{F}), \quad (3)$$

$$\sum_{j=1}^{n} w_j^s x_{ij}^s \le S_i, \ (\forall \mathcal{C}_i \in \mathcal{F}), \quad (4)$$

$$\sum_{\mathcal{C}_i \in \mathcal{F}} x_{ij}^c = r_j, \ (\forall j = 1, \ldots, n), \quad (5)$$

$$\sum_{\mathcal{C}_i \in \mathcal{F}} x_{ij}^s = r_j, \ (\forall j = 1, \ldots, n), \quad (6)$$

$$\sum_{j=1}^{n} (x_{ij}^c + x_{ij}^s) \ge 1, \ (\forall \mathcal{C}_i \in \mathcal{F}), \quad (7)$$

$$x_{ij}^c \ge 0, x_{ij}^s \ge 0, \text{ and are integers} \quad (\forall \mathcal{C}_i \in \mathcal{F} \text{ and } \forall j = 1, \ldots, n), \quad (8)$$

The objective function (2) represents $v(\mathcal{F})$, the total profit the participating cloud providers in federation $\mathcal{F}$ receive, which is equal to the revenue received from the user minus the cost incurred by the cloud providers. Constraints (3) ensure that the number of cores a cloud provider assigns to a user is less than the available number of cores provided by that cloud provider. Constraints (4) guarantee that the amount of storage assigned to the user is less than the amount of available storage at each cloud provider. Constraints (5) guarantee that the number of cores assigned to the user for each type of VM by all cloud providers is exactly the number of cores requested by the user for that type of VM. Constraints (6) guarantee that the storage assigned to the user for each type of VM by all cloud providers is exactly the amount of storage requested by the user for that type of VM. Constraints (7) ensure that each cloud provider in the federation contributes at least one type of resource. These constraints force the cloud providers to contribute resources to the federation. Constraint (8) represents the integrality requirement for the decision variables.

The *payoff* or the *share* of cloud provider $\mathcal{C}_i$ part of federation $\mathcal{F}$, denoted by $\psi_{\mathcal{C}_i}(\mathcal{F})$ is given by the normalized Banzhaf value [9]. The *Banzhaf value* is a division of payoffs for the grand federation that takes into account the power of the players. In this study, the power is defined as the market share of the cloud providers. A cloud provider that contributes more resources in all the possible federations in which it participates should receive higher profit regardless of its resource allocation in the selected federation.

III. Cloud Federation Formation Mechanism

Federation Formation Framework. The core of the cloud federation game can be empty. If the grand coalition does not form, independent and disjoint federations would form. Coalition formation theory investigates the coalitional structures in games where the grand coalition does not form. *Coalition formation* [10] is the partitioning of the players into disjoint sets. A federation structure $\mathcal{FS} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_h\}$ forms a partition of the set of cloud providers $\mathcal{I}$ such that each provider is a member of exactly one federation, *i.e.*, $\mathcal{F}_i \cap \mathcal{F}_j = \emptyset$ for all i and j, where $i \ne j$ and $\bigcup_{\mathcal{F}_i \in \mathcal{CF}} \mathcal{F}_i = \mathcal{I}$. The set of all federation structures is denoted by Π. The problem of finding the optimal federation structure is NP-complete [11].

In the cloud federation formation game defined in the previous section only one of the federations in the federation structure is selected to provide the resources requested by users. As a result, the formation of other federations with cloud providers outside of the selected federation is not important.

We model the cloud federation formation problem as a hedonic game [12] considering that cloud providers have preferences over the federations.

Definition 1 (Hedonic game): A hedonic game is a pair $(\mathcal{I}, \succeq)$, where $\succeq_i$ is a reflexive, complete, and transitive binary relation on Π_i, where Π_i is the set of coalitions in $\mathcal{I}$ containing $\mathcal{C}_i$.

We define the *federation preference relation* $\succeq_i$ for each $\mathcal{C}_i$. This allows $\mathcal{C}_i$ to compare two federations and to indicate its preference to be a part of one of them. $A \succeq_i B$ implies that $\mathcal{C}_i$ prefers to be a member of federation A than to be a member of federation B, or at least it prefers both federations equally. In addition, $A \succ_i B$ indicates that $\mathcal{C}_i$ strictly prefers to be a member of A than a member of B.

To model the cloud federation formation as a hedonic game, we need to define the federation preference relation. For all $\mathcal{C}_i \in \mathcal{I}$ and for all $\mathcal{F}, \mathcal{F}' \in \Pi_i$, we define $\succeq_i$ as

$$\mathcal{F} \succeq_i \mathcal{F}' \iff v(\mathcal{F}) \geq v(\mathcal{F}'). \tag{9}$$

That means a cloud provider prefers the federation that gives the higher profit. Using this preference relation, every cloud provider can evaluate its preferences over the set of possible federations that the cloud provider can be a member of.

We define two comparison relations in order to find a federation that is more preferred than other federations, the *merge comparison* $\triangleright_m$ and the *split comparison* $\triangleright_s$, as follows:

$$\begin{aligned}\{\mathcal{F} \cup \mathcal{F}'\} \quad & \triangleright_m \{\mathcal{F}, \mathcal{F}'\} \iff \\ & \{\forall \mathcal{C}_i \in \mathcal{F}; \{\mathcal{F} \cup \mathcal{F}'\} \succ_i \mathcal{F} \text{ and} \\ & \forall \mathcal{C}_j \in \mathcal{F}'; \{\mathcal{F} \cup \mathcal{F}'\} \succ_j \mathcal{F}'\}\end{aligned} \tag{10}$$

$$\begin{aligned}\{\mathcal{F}, \mathcal{F}'\} \quad & \triangleright_s \{\mathcal{F} \cup \mathcal{F}'\} \iff \\ & \{\exists \mathcal{C}_i \in \mathcal{F}; \mathcal{F} \succeq_i \{\mathcal{F} \cup \mathcal{F}'\} \text{ or} \\ & \exists \mathcal{C}_j \in \mathcal{F}'; \mathcal{F}' \succeq_j \{\mathcal{F} \cup \mathcal{F}'\}\}\end{aligned} \tag{11}$$

Equation (10) implies that federation $\{\mathcal{F} \cup \mathcal{F}'\}$ is preferred over two disjoint federations $\{\mathcal{F}, \mathcal{F}'\}$, if the profit obtained by federation $\{\mathcal{F} \cup \mathcal{F}'\}$ is greater than the profit obtained by the providers in $\mathcal{F}$, and it is greater than the profit obtained by the providers in $\mathcal{F}'$. As a result, all providers are able to improve the total profit. Equation (11) implies that $\{\mathcal{F}, \mathcal{F}'\}$ is preferred over $\{\mathcal{F} \cup \mathcal{F}'\}$, if at least one federation is able to keep the same amount of profit or to increase the profit of its members regardless of the effect on the other players outside that federation.

Using the defined comparison relations, we propose a cloud federation formation mechanism involving two types of rules as follows [10]:

Merge Rule: Merge any set of federations $\{\mathcal{F}, \mathcal{F}'\}$, where $\{\mathcal{F} \cup \mathcal{F}'\} \triangleright_m \{\mathcal{F}, \mathcal{F}'\}$.

Split Rule: Split any federation $\{\mathcal{F} \cup \mathcal{F}'\}$, where $\{\mathcal{F}, \mathcal{F}'\} \triangleright_s \{\mathcal{F} \cup \mathcal{F}'\}$.

Federations decide to merge only if all cloud providers are able to strictly improve the total profit through the merge rule. Therefore, the merge rule is an agreement among the cloud providers to operate together if it is beneficial for them.

As we mentioned before, one of the formed federations, the final federation, provides the requested VM instances, thus, the formation of the rest of the federations is not important. The reason for this is that the rest of the cloud providers which are not in the final federation can participate again in another federation formation process for allocating resources to another request. Therefore, a federation decides to split only if there is at least one sub-federation that strictly improves the total profit of its constituent cloud providers. Under the split rule, the profit of the other sub-federations may decrease. The split rule can be seen as the implementation of a *selfish* decision by a federation, which does not take into account the effect of the split on the other federations.

Through the merge-and-split process some of the possible federations are visited and their values are calculated. Based on those values, we define the *estimated Banzhaf value* of $\mathcal{C}_i$ as follows:

$$E_{\mathcal{C}_i}(\mathcal{I}) = \frac{1}{\lambda} \sum_{\substack{\mathcal{F} \subseteq \mathcal{I} \setminus \{\mathcal{C}_i\} \\ \mathcal{F} \in \mathcal{V} \\ \mathcal{F} \cup \mathcal{C}_i \in \mathcal{V}}} [v(\mathcal{F} \cup \{\mathcal{C}_i\}) - v(\mathcal{F})]. \tag{12}$$

where $\mathcal{V}$ is the set of all visited federations, and λ is the total number of visited federations containing $\mathcal{C}_i$. That means, $\lambda = 2^{m-1} - \alpha$, where α is the number of non-visited federations. The estimated Banzhaf value is based only on the value of federations that are visited during the merge and split process. The normalized estimated Banzhaf value is defined as

$$\mathcal{E}_{\mathcal{C}_i}(\mathcal{I}) = \frac{E_{\mathcal{C}_i}(\mathcal{I})}{\sum_{\mathcal{C}_j \in \mathcal{I}} E_{\mathcal{C}_j}(\mathcal{I})}. \tag{13}$$

The profit that each member $\mathcal{C}_i$ receives in the grand federation is calculated as follows:

$$\psi_{\mathcal{C}_i}(\mathcal{I}) = \mathcal{E}_{\mathcal{C}_i}(\mathcal{I}) v(\mathcal{I}). \tag{14}$$

The payoff vector $\Psi(\mathcal{I}) = (\psi_{\mathcal{C}_1}(\mathcal{I}), \cdots, \psi_{\mathcal{C}_m}(\mathcal{I}))$ gives the payoff divisions of the grand federation. We define $\psi_{\mathcal{C}_i}(\mathcal{F})$, the payoff of cloud provider $\mathcal{C}_i \in \mathcal{F}$, as follows:

$$\psi_{\mathcal{C}_i}(\mathcal{F}) = \frac{\psi_{\mathcal{C}_i}(\mathcal{I})}{\sum_{\forall \mathcal{C}_j \in \mathcal{F}} \psi_{\mathcal{C}_j}(\mathcal{I})} v(\mathcal{F}). \tag{15}$$

During the merge-an-split we estimate the Banzhaf value for each provider based only on the federations that were already explored. The profit obtained by the federation is divided among participating cloud providers in proportion to their power in the federation.

Cloud Federation Formation Mechanism (CFFM). The proposed cloud federation formation mechanism (CFFM) is given in Algorithm 1. A broker executes the mechanism. CFFM uses a branch-and-bound method to solve the IP problem for each federation to find the allocation and the profit of the federation. We denote by B&B-VM-ALLOCATION($\mathcal{F}_i$) the function that implements the branch-and-bound method for solving the IP problem for a federation $\mathcal{F}_i$.

CFFM starts with a request from a user. A federation structure $\mathcal{FS}$ consisting of every singleton $\mathcal{C}_i \in \mathcal{I}$ as a federation $\mathcal{F}_i$ is formed. Then, CFFM calculates $v(\mathcal{F}_i)$. CFFM uses a matrix $visited$ to keep track of all pairs of federations in $\mathcal{FS}$ that are visited for merging. By using this matrix, all

Algorithm 1 Cloud Federation Formation Mechanism (CFFM)

```
1:  Receive request $\mathcal{R}$
2:  $\mathcal{FS} = \{\{\mathcal{C}_1\}, \cdots, \{\mathcal{C}_m\}\}$
3:  Calculate $v(\mathcal{F}_i)$ for each $\mathcal{F}_i \in \mathcal{FS}$
4:  repeat
5:      $stop \leftarrow$ TRUE
6:      for all $\mathcal{F}_i, \mathcal{F}_j \in \mathcal{FS}, i \neq j$ do
7:          $visited[\mathcal{F}_i][\mathcal{F}_j] \leftarrow$ FALSE
8:      end for
9:      {Merge process starts:}
10:     repeat
11:         $flag \leftarrow$ TRUE
12:         Randomly select $\mathcal{F}_i, \mathcal{F}_j \in \mathcal{FS}$ for which
                $visited[\mathcal{F}_i][\mathcal{F}_j] =$ FALSE$, i \neq j$
13:         $visited[\mathcal{F}_i][\mathcal{F}_j] \leftarrow$ TRUE
14:         B&B-VM-ALLOCATION($\mathcal{F}_i \cup \mathcal{F}_j$)
                {Allocate VMs using $\mathcal{F}_i \cup \mathcal{F}_j$}
15:         if $\mathcal{F}_i \cup \mathcal{F}_j \triangleright_m \{\mathcal{F}_i, \mathcal{F}_j\}$ then
16:             $\mathcal{F}_i \leftarrow \mathcal{F}_i \cup \mathcal{F}_j$ {merge $\mathcal{F}_i$ and $\mathcal{F}_j$}
17:             $\mathcal{F}_j \leftarrow \emptyset$ {$\mathcal{F}_j$ is removed from $\mathcal{FS}$}
18:             for all $\mathcal{F}_k \in \mathcal{FS}, k \neq i$ do
19:                 $visited[\mathcal{F}_i][\mathcal{F}_k] \leftarrow$ FALSE
20:             end for
21:         end if
22:         for all $\mathcal{F}_i, \mathcal{F}_j \in \mathcal{FS}, i \neq j$ do
23:             if not $visited[\mathcal{F}_i][\mathcal{F}_j]$ then
24:                 $flag \leftarrow$ FALSE
25:             end if
26:         end for
27:     until $(|\mathcal{FS}| = 1)$ or $(flag =$ TRUE$)$
28:     {Split process starts:}
29:     for all $\mathcal{F}_i \in \mathcal{FS}$ where $|\mathcal{F}_i| > 1$ do
30:         for all partitions $\{\mathcal{F}_j, \mathcal{F}_k\}$ of $\mathcal{F}_i$,
                where $\mathcal{F}_i = \mathcal{F}_j \cup \mathcal{F}_k, \mathcal{F}_j \cap \mathcal{F}_k = \emptyset$ do
31:             B&B-VM-ALLOCATION($\mathcal{F}_j$)
                    {Allocate VMs using $\mathcal{F}_j$}
32:             B&B-VM-ALLOCATION($\mathcal{F}_k$)
                    {Allocate VMs using $\mathcal{F}_k$}
33:             if $\{\mathcal{F}_j, \mathcal{F}_k\} \triangleright_s \mathcal{F}_i$ then
34:                 $\mathcal{F}_i \leftarrow \mathcal{F}_j$ {that is $\mathcal{FS} = \mathcal{FS} \setminus \mathcal{F}_i$}
35:                 $\mathcal{FS} = \mathcal{FS} \bigcup \mathcal{F}_k$
36:                 $stop \leftarrow$ FALSE
37:                 Break (one split occurs;
                        no need to check other splits)
38:             end if
39:         end for
40:     end for
41: until $stop =$ TRUE
42: Find $\mathcal{F}_k = \arg\max_{\mathcal{F}_i \in \mathcal{FS}} \{v(\mathcal{F}_i)\}$
43: Calculate $\psi_{\mathcal{C}_i}(\mathcal{F}_k)$, $\forall \mathcal{C}_i \in \mathcal{F}_k$
44: $\mathcal{F}_k$ allocates and provides the requested VM instances.
```

possible combinations of two federations in $\mathcal{FS}$ are visited during the merge process. The merge process starts every time by choosing two non-visited federations in $\mathcal{FS}$ randomly, e.g., $\mathcal{F}_i$ and $\mathcal{F}_j$. B&B-VM-ALLOCATION is called to find an optimal VM allocation on $\mathcal{F}_i \cup \mathcal{F}_j$. If $\mathcal{F}_i \cup \mathcal{F}_j \triangleright_m \{\mathcal{F}_i, \mathcal{F}_j\}$, then federations $\mathcal{F}_i$ and $\mathcal{F}_j$ decide to merge. $\mathcal{F}_i \cup \mathcal{F}_j$ is saved in $\mathcal{F}_i$, and $\mathcal{F}_j$ is removed from $\mathcal{FS}$. Since $\mathcal{F}_i$ is changed, it can be selected in the next merge steps. Thus, $visited[\mathcal{F}_i][\mathcal{F}_k]$ for all $\mathcal{F}_k \in \mathcal{FS}$, $k \neq i$ is set to false. The merge process tries to find another pair of non-visited federations suitable for merging. If all the federations are tested and a merge does not occur, or the grand federation forms, the merge process ends.

The federation structure $\mathcal{FS}$ obtained by the merge process is then subject to splits. In the split process, all federations that have more than one member are subject to splitting. The mechanism tries to split $\mathcal{F}_i$ that has more than one member into two disjoint federations $\mathcal{F}_j$ and $\mathcal{F}_k$ where $\mathcal{F}_j \cup \mathcal{F}_k = \mathcal{F}_i$. B&B-VM-ALLOCATION is called twice to find an optimal allocation on $\mathcal{F}_j$ and an optimal allocation on $\mathcal{F}_k$. Since the split is a selfish decision, the splitting occurs even if only one of the members of federation $\mathcal{F}_j$ or $\mathcal{F}_k$ can improve its individual value. As a result, the federation with the higher individual payoff is the decision maker for the split.

If one or more federations split, then the merging process starts again. To do so, the *stop* flag is set to false. Multiple successive merge-and-split operations are repeated until the mechanism terminates. That means that there are no choices for merge or split for all existing federations in $\mathcal{FS}$. Let's consider $\mathcal{FS}_{final}$ as the final federation structure. The mechanism selects one of the federations in the $\mathcal{FS}_{final}$ that yields the highest total profit. The mechanism calculates the individual profit of the participating clouds in the federation using the normalized estimated Banzhaf value. The selected federation will allocate and provide the requested VM instances to the user.

In the following, we characterize the properties and the stability of the cloud federation obtained by CFFM. We define the *individual federation stability* as follows. A federation $\mathcal{F}$ is individual federation stable if there is no cloud provider $\mathcal{C}_i \in \mathcal{F}$ that can leave $\mathcal{F}$ without making at least one cloud provider $\mathcal{C}_j \in \mathcal{F}$ unhappy. We showed that CFFM produces an individually stable federation. This result and its proof will be presented in an extended version of this paper.

TABLE I: The properties of available VM instances.

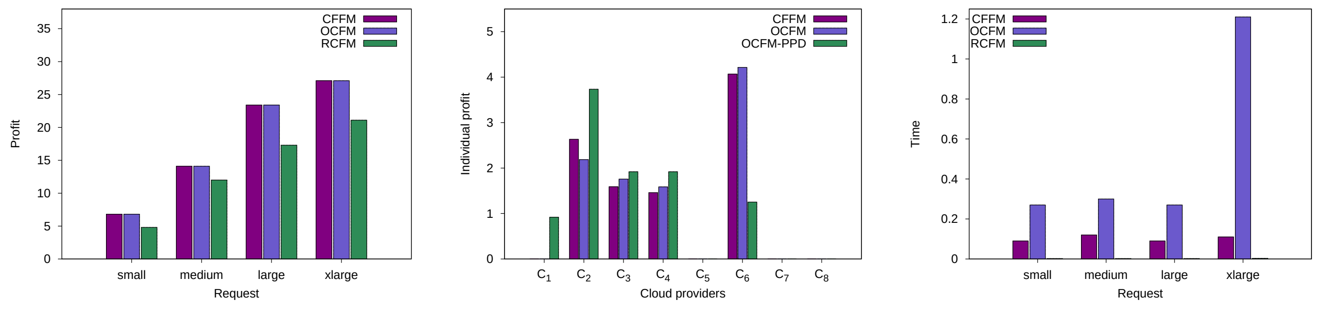
	VM_1	VM_2	VM_3	VM_4
w_j^c (1.6GHz CPU)	1	2	4	8
w_j^s (TB Storage)	0.22	0.48	0.98	1.99

IV. Experimental Results

Experimental Setup. We consider eight cloud provides offering four types of VM instances. We considered only eight cloud providers since it is a reasonable estimation of the number of cloud providers that could potentially form a federation in practice. We consider four types of VM instances $\mathcal{VM} = \{VM_1, VM_2, VM_3, VM_4\}$ representing small, medium, large and extra large VM instances, respectively. The description of the VM instances is provided in Table I. The instance types and pricing are similar to the ones used by Microsoft Azure [7].

The parameters used in our experiments and their values are listed as follows: N_i, the number of cores, is a random number between [100, 1000], and S_i, the amount of storage (TB), is a random number in [1, 100] for each cloud provider i. c_j^c, core cost matrix, p_j^c, core price vector, c_j^s, storage cost matrix, and p_j^s, storage price vector are a random number in [0,1] for each VM instance j. We use the ILOG Concert Technology APIs in C++ to solve the IP problem by CPLEX solver [13].

Analysis of Results. We compare the performance of our cloud federation formation mechanism, CFFM, with that of



(a) (b) (c)

Fig. 1: (a) Total Profit of the Cloud Federation; (b) Profit of Individual Cloud Providers; (c) Execution Time of the Mechanisms.

two other mechanisms: Optimal Cloud Federation (OCFM), Random Cloud Federation (RCFM). The OCFM mechanism finds the optimal allocation on all cloud providers by solving a relaxed problem in which the constraints (7) in the proposed IP are not considered. As a result, in OCFM it is not necessary that all the cloud providers provide resources to fulfill the user request. The RCFM mechanism selects several cloud providers randomly and forms a federation. All the mechanisms use the branch-and-bound method for solving the proposed IP. We consider four different customer requests, $(10, 0, 0, 0)$, $(10, 10, 0, 0)$, $(10, 10, 10, 0)$, and $(10, 10, 10, 10)$ representing small, medium, large and extra-large, respectively. All requests cannot be served by only one cloud provider and they need to form a federation in order to serve the user. We perform a series of ten experiments in each case, and we represent the average of the obtained results.

In Fig. 1a, we compare the total profit obtained by CFFM with that obtained by the other two mechanisms. In all cases CFFM provides the highest profit which is the same as the optimal profit obtained by OCFM. These results show that using a RCFM is not efficient in terms of the total profit of the federation. For example, for a small request, CFFM and OCFM obtain a total profit of \$6.81, while RCFM obtains a total profit of \$4.8.

In Fig. 1b, we show the individual value of each participating cloud provider in the federation for a medium request. We present three different profit divisions: CFFM, OCFM and OCFM-PPD. CFFM uses the estimated normalized Banzhaf value while the OCFM uses the normalized Banzhaf value. OCFM-PPD is a variant of OCFM that uses proportional profit division instead of the Banzhaf value. Here, the proportional payoff division means that each cloud provider participating in the federation and providing resources receives a profit equal to the price that the user pays for that resource minus the cost the cloud provider incurs to provide the resource. CFFM and OCFM obtain a total profit \$9.75, where both mechanisms find a federation of size 4, $\{C_2, C_3, C_4, C_6\}$. CFFM explores 53 federations until it finds the final federation. As it is shown in the figure, the individual profit of the participating cloud providers are very close in CFFM and OCFM.

Fig. 1c shows the execution time of the three mechanisms. These results were obtained on a 3.00GHz Intel quad-core PC with 8GB of memory. From 255 federations that 8 cloud providers could form, CFFM only considers some of them in the merge-and-split process based on the merge and split rules. On average, CFFM explores 48 federations until it finds the final federation. As a result, the execution time of CFFM is a lot less than that of OCFM which goes through all the federations. For each federation, both mechanisms run the IP solver once. In cases that the IP solver requires more time (i.e., for larger request), the execution time of both mechanisms increases. RCFM execution time is close to zero since the the IP solver is executed for only one federation taking about 3.3 milliseconds.

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

References

[1] B. Rochwerger, D. Breitgand, E. Levy, A. Galis *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM J. of Res. and Dev.*, vol. 53, no. 4, pp. 4–1, 2009.

[2] I. Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," in *Proc. IEEE Intl. Conf. on Cloud Comp.*, 2010, pp. 123–130.

[3] M. Hassan, B. Song, and E. Huh, "Distributed resource allocation games in horizontal dynamic cloud federation platform," in *Proc. IEEE Intl. Conf. on High Perf. Comp. and Comm.*, 2011, pp. 822–827.

[4] D. Niyato, A. Vasilakos, and Z. Kun, "Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach," in *Proc. IEEE/ACM Intl. Symp. on Cluster, Cloud and Grid Comp.*, 2011, pp. 215–224.

[5] W. Saad, Z. Han, M. Debbah, and A. Hjorungnes, "A distributed coalition formation framework for fair user cooperation in wireless networks," *IEEE Trans. on Wireless Comm.*, vol. 8, no. 9, pp. 4580–4593, 2009.

[6] L. Mashayekhy and D. Grosu, "A merge-and-split mechanism for dynamic virtual organization formation in grids," in *Proc. IEEE Intl. Perf. Comp. and Comm. Conf.*, 2011, pp. 1–8.

[7] WindowsAzure. [Online]. Available: http://www.windowsazure.com/en-us/pricing/calculator/

[8] G. Owen, *Game Theory*, 3rd ed. New York, NY, USA: Academic Press, 1995.

[9] ——, "Multilinear extensions and the banzhaf value," *Naval Research Logistics Quarterly*, vol. 22, no. 4, pp. 741–750, 1975.

[10] K. Apt and A. Witzel, "A generic approach to coalition formation," *International Game Theory Review*, vol. 11, no. 3, pp. 347–367, 2009.

[11] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme, "Coalition structure generation with worst case guarantees," *Artificial Intelligence*, vol. 111, pp. 209–238, 1999.

[12] A. Bogomolnaia and M. Jackson, "The stability of hedonic coalition structures," *Games & Econ. Behavior*, vol. 38, no. 2, pp. 201–230, 2002.

[13] "IBM ILOG CPLEX Optimization Studio for Academics Initiative." [Online]. Available: http://www01.ibm.com/software/websphere/products/optimization/academic-initiative/