Performance Driven Resource Sharing Markets for the Small Cloud

Sung-Han Lin, Ranjan Pal, Marco Paolieri, Leana Golubchik
Department of Computer Science, University of Southern California
{*sunghan, rpal, paolieri, leana*}@usc.edu

***Abstract*—Small-scale clouds (SCs) often suffer from resource under-provisioning during peak demand, leading to inability to satisfy service level agreements (SLAs) and consequent loss of customers. One approach to address this problem is for a set of autonomous SCs to share resources among themselves in a cost-induced cooperative fashion, thereby increasing their individual capacities (when needed) without having to significantly invest in more resources. A central problem (in this context) is how to properly share resources (for a price) to achieve profitable service while maintaining customer SLAs. To address this problem, in this paper, we propose the *SC-Share* framework that utilizes two interacting models: (i) a *stochastic performance model* that estimates the achieved performance characteristics under given SLA requirements, and (ii) a *market-based game-theoretic model* that (as shown empirically) converges to efficient resource sharing decisions at market equilibrium. Our results include extensive evaluations that illustrate the utility of the proposed framework.**

***Index Terms*—data centers; small cloud; performance; markets**

I. Introduction

Infrastructure-as-a-Service is quickly becoming a ubiquitous model for providing elastic compute capacity to customers who can access resources in a pay-as-you-go manner without long-term commitments, with rapid scaling (up or down) as needed [1]. Cloud service providers (Amazon AWS [2], Google Compute Engine [3], and Microsoft Azure [4]) allow customers to quickly deploy their services without a large initial infrastructure investment.

***Proliferation of smaller-scale clouds*.** However, there are some non-trivial concerns in obtaining services from large-scale public clouds, including *cost* and *complexity*. Massive cloud environments can be costly and inefficient for some customers, e.g., Blippex [5], thus resulting in more and more customers building their own smaller-scale clouds (SCs) [6] for better control of resource usage; e.g., it is hard to guarantee network performance in large-scale public clouds due to their multi-tenant environments [7]. Moreover, smaller-scale providers exhibit a greater flexibility in customizing services for their users, while large-scale public providers minimize their management overhead by simplifying their services; e.g., Linode [8] distinguishes itself by providing clients with easier and more flexible service customization. The use of SCs is one approach to resolving cost and complexity issues.

Despite the potential of SCs, they are likely to suffer from resource under-provisioning during peak demand, which can lead to inability to satisfy service level agreements (SLAs) and consequent loss of customers. SLAs come in many forms, such as the average or maximum waiting time before being served, the probability of requests being rejected, and the amount of resources that each request can obtain. In order not to resort, similarly to large-scale providers, to resource over-provisioning, with all its disadvantages, one approach to realizing the benefits of SCs is to adopt hybrid architectures [9], [10], that allow private clouds (or small cloud providers) to outsource their requests to larger-scale public providers. However, the use of public clouds can potentially be costly for the small-scale provider.

Motivation. An emerging approach to solving the under-provisioning problem is for SCs to share their resources in a *federated cloud environment* [11]–[17], thus (effectively) increasing their individual capacities (when needed) without having to significantly invest in more resources, e.g., this can be helpful when the SCs do not experience peak workloads at the same time. Earlier efforts [16], [17] characterize the benefits of cloud federations, while [16] also studies the motivation for cloud providers to participate in the federation. Authors of [14] present SpotCloud [18], a cooperative system that helps providers selling idle resources to other providers or end-users at specified prices, and they analyze pricing models that incentivize providers to contribute their resources. Other efforts in cloud sharing domain demonstrate that the uncertainty in meeting SLAs can be an incentive enabling sharing of resources among clouds [13], and focus on designing efficient sharing mechanisms [12], [15].

However, many of these efforts assume the existence of the cloud federation and largely focus on designing sharing policies in order to maximize the profit of individual SCs. For example, [15] proposes a strategy to terminate less profitable spot instances, in order to accommodate more profitable on-demand VM requests. Moreover, most authors do not consider the trade-off between economical benefits (in terms of profit) and performance degradation for individual SCs, which is a significant factor to incentivizing SCs to participate in the cloud federation. Without the analysis of performance degradation due to resource sharing, the feasibility of a federation can be questioned. Thus, this work focuses on the fundamental, unanswered question of "*how each SCs should share resources to be profitable without violating customer SLAs, while also motivating other SCs to join the federation.*"

***Problem description*.** We consider an environment with multiple SCs; an example with 3 SCs is depicted in Fig. 1. In this work, we also refer to SCs sharing resources with each other as a *federation*. Each SC has its own SLAs with its customers: *the maximum waiting time before service of a request is initiated.* To satisfy SLAs, some SCs use public clouds as a "backup", i.e., buy needed resources on-demand from large-scale public clouds, when in danger of not being able to meet SLAs. If such SCs form a federation, in the event that an SC runs out of its resources, it can first use shared resources from

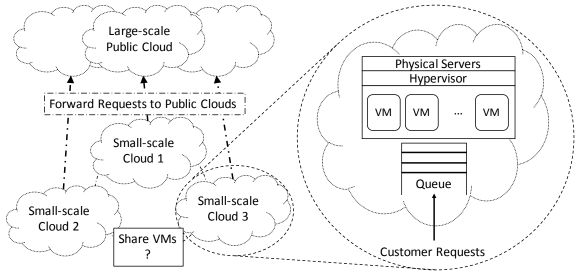


Fig. 1. System Overview

other SCs for a price lower than the price of using public clouds. The amount of shared resources directly affects how much workload the federation is able to handle, which in turn affects the profit each SC is able to achieve. In this sharing scenario, an important question is: *should SCs participate in the federation? If yes, how much should each SC share*? If an SC is too generous (i.e., shares too many of its resources), then it may be in danger of not being able to serve its own workload, resulting in more requests being forwarded to public clouds thereby reducing profit margins. As a result, an SC should determine the amount of resources shared based on the price of selling and buying resources, i.e., the net profit compared with the cost of using public clouds. However, if an SC is too selfish, i.e., shares few of its resources for higher profit, then either it may get removed from the federation for not being a useful contributor, or the federation may fall apart if most/all SCs tend towards selfish behavior. Thus, another critical question that needs to be addressed is: *what price can make each SC share a reasonable amount of resources so that other SCs will participate in the federation?*

Challenges and contributions. To answer these questions, we provide the following contributions:

1) ***Performance-dependent cost function:*** Operating costs of an SC depend on the SLA with its customers and on the performance achieved inside the federation; in particular, we need to compute how frequently the SC will need to allocate external resources to satisfy SLAs (e.g., maximum waiting time), and whether it will be able to access resources of other SCs, or only those of public clouds. In Sect. III-B, we develop a *detailed performance model* to compute such performance metrics for each SC. In turn, these metrics allow us to compute the operating cost of SCs (as defined in Sect. II-B). To address the high computational complexity of the detailed performance model (due to its large state space, which grows exponentially with the number of SCs), we develop an *approximate performance model* (Sect. III-C). This model provides accurate estimates of the measures of interest, with linear complexity in the number of SCs (which is crucial, given that SCs must take sharing decisions almost in real-time).
2) ***Sharing market design:*** The sharing mechanism should motivate SCs to participate, without significant oversight nor management, i.e., they should find an economic benefit in contributing resources to the federation. We design a market-based model to determine the price charged within the federation for the use of shared resources. The model is based on a non-cooperative, repeated game among SCs, each being selfish and trying to maximize its utility; as in real-world scenarios, SCs do not know the utility of other SCs, but they can compute (using our approximate performance model) the operating cost that they would incur for each possible sharing decision. We determine *market equilibrium* conditions under which the federation is successful and *market efficiency* is achieved (Sect. IV).
3) ***Experimental evaluation:*** In Sect. V, we perform an extensive experimental evaluation to validate the accuracy of our approximate performance model with respect to simulation, and to verify the existence of market equilibria. Results highlight errors lower than 10% for the performance metrics of interest; the proposed pricing model achieves market equilibria and good economic efficiency, successfully incentivizing SCs to stay in the federation.

To the best of our knowledge, ours is the first work that models small-cloud federations as a holistic *performance-driven market*, integrating engineering aspects (from a performance model) with economic ones (from a market model).

II. System Description

In this section, we first describe the architecture of the SC federation, illustrated in Fig. 1. We then introduce a definition of operating costs of SCs. Finally, we describe our sharing framework, which we call *SC-Share*.

A. Architecture Description

Each SC has a number of physical servers: through virtualization technology, physical resources (CPU, memory, storage) of SC i are packed into N_i homogeneous virtual machines (VMs), which are the resource unit adopted in this work. Customers request the allocation of individual VMs from SCs; the arrival process of VM requests at each SC i is modeled as a Poisson process with rate λ_i. The service time of each request at SC i (including the time elapsed from start of VM preparation until its release by the user) is modeled as an exponential random variable with rate μ_i. Each SC processes VM requests in FCFS order. If physical servers do not have sufficient resources for a new VM, the SC may need to reject the request, or queue it until more resources are available.

In a federation with K SCs (Fig. 1 depicts the case $K = 3$), we consider the following general scenario: when all VMs at an SC are fully occupied, its new VM requests are queued and can be served either by waiting for local resources to become available, or by purchasing resources from other SCs in the federation, or from a public cloud. In order to participate in the federation, SC i must determine the maximum number of VMs S_i to share with other SCs (at a given price) when idle VMs are available; i.e., at any time instant, the number of VMs shared by SC i is $I_i^{S_i} \leq S_i$. When all its VMs are occupied, SC i cannot terminate VMs serving requests of other SCs, but only stops accepting such requests until it is able to clear its own queue. Each SC i is required to maintain SLAs with its customers; we assume that this corresponds to a bound on the waiting time, i.e., a VM needs to be provided by SC i within Q_i time units from its request. If SC i determines that it is not able to satisfy this SLA using resources of the federation, it forwards the request to a public cloud (e.g., Amazon AWS).

In our performance model (Section III), we assume that SCs share the same type of VMs within the federation. This assumption is reasonable because many cloud providers sell similar VM configurations. For instance, Amazon LightSail [19], DigitalOcean, and Linode all have similar VM configurations with 2 CPU cores and 4 GB of RAM. However, due to the heterogeneous datacenters of different providers, the same VM configuration may exhibit different performance; for instance, DigitalOcean VMs can exhibit better I/O performance because SSDs are directly attached to physical machines, while Amazon LightSail VMs access SSDs remotely. Our framework does not account for such performance mismatch among VMs of different SCs, which is a future research direction.

B. Cost Metric Description

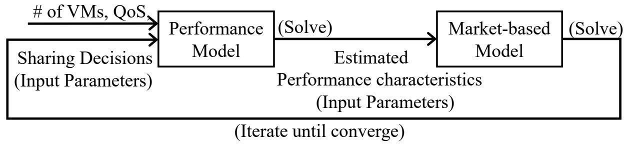
SCs usually make large up-front investments in infrastructure, and continue to pay for maintenance costs (e.g., power supply and cooling costs). In addition, SCs need to consider costs for forwarding requests to public clouds or for using resources of other SCs in the federation, in order to satisfy customer SLAs. We define a *cost metric* to combine these costs with the revenue generated by VM requests from other SCs in the federation, and compute the net operating cost.

Let $I_i^{S_i}$ be a random variable representing the number of SC i's VMs used by other SCs when SC i shares up to S_i VMs with the federation. Let $O_i^{S_i}$ and $P_i^{S_i}$ be random variables representing the number VMs used by SC i from the federation and from a public cloud, respectively, to satisfy its SLAs. The net cost for SC i is then

$$C_i^{S_i} = \overline{P_i^{S_i}} \cdot C_i^P + (\overline{O_i^{S_i}} - \overline{I_i^{S_i}}) \cdot C_i^G \quad \forall i, \tag{1}$$

where C_i^P and C_i^G represent the cost of using a single VM from a public cloud and from other SCs, respectively. $\overline{P_i^{S_i}}$, $\overline{O_i^{S_i}}$, and $\overline{I_i^{S_i}}$ are the mean number of VMs per second used by SC i from a public cloud, by SC i from other SCs in the federation, or by other SCs from SC i, respectively. Unlike [15], where cloud providers change VM prices based on system utilization, our model considers a fixed price C_i^G for every VM. Since VMs are homogeneous, we assume that $C_i^G = C_j^G \forall i, j = 1, ..., K$, but each SC can have different C_i^P depending on which public cloud it uses. These assumptions simplify our performance model, where SCs allocate available VMs in the federation without preferences due to prices. *To reduce cost, by making appropriate sharing decisions, i.e., determining the number of VMs to share with others, we need a performance model for each SC, in order to properly estimate* $\overline{P_i^{S_i}}$, $\overline{O_i^{S_i}}$, *and* $\overline{I_i^{S_i}}$ *(see Section III for details).*

Another incentive for participating in the federation is reducing power cost by forwarding VMs to other SCs when they offer VMs at cheaper prices than the cost of instantiating VMs in SC's own environment. For instance, previous efforts [17], [20] study the sharing mechanisms for cloud providers to minimize their costs. However, in this work, we only focus on the cost of additional resources required to satisfy customers' SLAs. Extending the cost function to incorporate power consumption of executing VMs is a future direction.



# of VMs, QoS
Performance Model
(Solve)
Market-based Model
(Solve)
Sharing Decisions (Input Parameters)
Estimated Performance characteristics (Input Parameters)
(Iterate until converge)

Fig. 2. Feedback between two models

C. Cost Metric Evaluation Framework

In order to help SCs determine whether it is beneficial for them to participate in the federation and share their resources, we design a framework *SC-Share* that allows each SC i to determine the best value of S_i, in order to maintain its SLA Q_i and minimize the expected operating cost $C_i^{S_i}$.

The essence of SC cooperation in such a federation is the mutual agreement among individual SCs to share their resources (if idle) with other SCs experiencing peak workloads.[1] However, the amount of resources S_i that each selfish but honest SC wants to share represents its strategic property that subsequently affects the cost metric $C_i^{S_i}$. Thus, in *SC-Share*, we develop a market-based model to capture SC interactions in the federation via a market consisting of K selfish SCs that interact strategically, and repeatedly over time, via a non-cooperative game to converge upon stable parameter values.

However, a feedback loop exists between the performance model and the market model: sharing decisions $S_i \; \forall i$ are used by the performance model to compute $\overline{P_i^{S_i}}$, $\overline{O_i^{S_i}}$, and $\overline{I_i^{S_i}}$ and evaluate the cost metrics $C_i^{S_i}$ in Eq. (1), which, in turn, determine the SC utility functions of the market model governing sharing decisions. Therefore, in *SC-Share*, we propose an iterative solution approach, as illustrated in Fig. 2, involving these two models and their mutual feedback, to converge upon stable sharing decisions.

III. Performance Model

In this section, we propose a performance model for *SC-Share* that is used to compute performance parameters required by the cost function of Eq. (1).

A. SC without Sharing Resources

We start with a degenerate case, where an SC does not participate in the federation and shares no VMs. Based on SLA requirements, the SC will forward a request to public clouds if service cannot be started within Q_i time units after its reception. To compute the cost, we need to estimate the mean number of requests forwarded per second by SC i, $\overline{P_i^0}$ (we denote it with 0 since no VMs are shared).

To compute $\overline{P_i^0}$, we use a Markovian model, where the state represents the number of requests at SC i, as illustrated in Fig. 3. In this example, we assume that SC i has N_i VMs and SLA Q_i with its customers. When at least one VM is idle, a new request can be served immediately. However, when all VMs are allocated, the probability that the new request is

[1]The issue of enforcing the agreement is beyond our scope here.

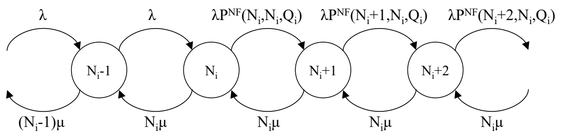


Fig. 3. A Markov model for forwarding

added to the queue of SC i (instead of forwarded to a public cloud) is equal to the probability that service will start in Q_i time units, based on the current number of queued requests. Let q_i be the number of customers in SC i (i.e., $\max(0, q_i - N_i)$ customers are waiting in its queue) at the time of the request arrival. Then, given exponential service times with rate μ and the FCFS service policy, the probability of queueing the request (instead of forwarding to a public cloud) is

$$P^{NF}(q_i, N_i, Q_i) = \begin{cases} 1 - \sum_{j=0}^{q_i - N_i} \frac{e^{-N_i \mu Q_i} \cdot (N_i \mu Q_i)^j}{j!} & \text{if } q_i \geq N_i, \\ 1 & \text{if } q_i < N_i. \end{cases} \tag{2}$$

In particular, $P^{NF}(q_i, V_i, Q_i)$ is less than one if the request cannot be served immediately upon arrival (i.e., $q_i \geq N_i$).

At the steady state, the expected probability of forwarding a new request to public clouds is then $P^F = \left(\sum_{k=N_i}^{\infty}(1 - P^{NF}(k, N_i, Q_i)) \cdot \pi_k\right)$, where π_k is the steady-state probability of having k requests in the system. So, the expected rate at which VM requests are forwarded to public clouds is $\overline{P_i^0} = \lambda \cdot P^F$, which can be used in Eq. (1) to compute the cost for $\overline{O_i^0} = \overline{I_i^0} = 0$.

B. Detailed Model for SC federation

The model of a federation (with sharing) is complex. Given a federation of K SCs, each of which will share a maximum of $S_i, i = 1, \dots, K$ VMs, our goal is to estimate the performance parameters $\overline{P_i^{S_i}}$, $\overline{O_i^{S_i}}$, and $\overline{I_i^{S_i}}$ for each SC i. To accurately estimate these parameters, we need to consider the interaction among SCs in the federation. One approach is to build a continuous-time Markov chain (CTMC), $\mathcal{M}$, with the following state space $\mathcal{S}$:

$$\mathcal{S} = \{(q_1, s_{1,1}, ..., s_{1,K}, ..., q_K, s_{K,1}, ..., s_{K,K}) \mid q_i \geq 0, \\ 0 \leq s_{i,j} \leq S_j,\ s_{i,i} = \sum_{j \neq i} s_{j,i} \leq S_i, i = 1, ..., K\},$$

where q_i is the number of requests from SC i's customers that are either queued or in service at SC i, $s_{i,i}$ is the number of VMs at SC i serving requests from other SCs, and $s_{i,j}, j \neq i$ is the number of VMs at SC j being used by SC i. Transition rates between states of $\mathcal{M}$ can be assigned so as to implement the probabilistic forwarding mechanism of the model for new arrivals, and service of queued requests. Due to lack of space, the transition structure for detailed model $\mathcal{M}$ is reported in Appendix A.

Although solving $\mathcal{M}$ could give us an accurate prediction of all performance characteristics required in Eq. (1), the corresponding state space $\mathcal{S}$ grows exponentially with K. Since re-computation of sharing decisions is needed when significant changes in workload or resource availability occur, a model with a more efficient solution is desirable.

C. Approximate Model for SC Federation

In this section, we focus on an approximate model that can be solved in (near) real-time (as system conditions, such as workload, change), but also yields sufficiently accurate results, in order to produce appropriate sharing decisions. Through analyzing the detailed model $\mathcal{M}$, we realize that using $\mathcal{M}$ allows estimation of performance parameters for all SCs in the federation simultaneously; however, in realistic scenarios, each SC computes its own performance parameters to estimate its cost assuming that other SCs' sharing decisions are fixed; thus, there is no need for the performance model to simultaneously output results for all SCs. Moreover, since we assume that the same cost is charged by all SCs for shared VMs, an SC does not need to distinguish the source or destination of shared VMs. Therefore, we propose a hierarchical approximate model that computes performance parameters iteratively.

Given a federation of K SCs, we consider each SC $i = 1, \dots, K$ in sequence, where SC K is the SC of interest, which we refer to as *target SC* in the rest of the paper. At each step, we build and analyze a Markovian model $\mathcal{M}^i$ where only SCs $\{1, \dots, i\}$ can access shared resources of the federation. The model $\mathcal{M}^i$ takes into account the solution of $\mathcal{M}^{i-1}$ and refines it to include also SC i. For example, in model $\mathcal{M}^1$, the first SC has exclusive access to all shared resources of the federation; in $\mathcal{M}^2$, only SC 1 and SC 2 utilize shared resources from all SCs, but VM allocations in $\mathcal{M}^1$ are taken into account. We repeat this process until reaching the target SC. In the following, we give a detailed description of $\mathcal{M}^i$, $1 \leq i \leq K$ and of its solution.

State space $\mathcal{S}_i$ for $\mathcal{M}^i$. The state space $\mathcal{S}_i$ of $\mathcal{M}^i$ is

$$\mathcal{S}^i = \{(q_i, s_i, o_i, a_i) \mid q_i \geq 0,\ 0 \leq s_i \leq S_i,\ 0 \leq o_i + a_i \leq B_i\},$$

where q_i is the total number of requests at SC i (queued or in service), s_i is the number of VMs of SC i currently used to serve requests from SCs $\{1, \dots, i-1\}$, o_i is the number of VMs from other SCs currently used by SC i, and a_i is the number of shared VMs used by SCs in $\mathcal{M}^{i-1}$. Given that there are at most N_i VMs in SC i, $\max(0, q_i - (N_i - s_i))$ requests are waiting at SC i; moreover, s_i is bounded by S_i, the maximum number of VMs shared by SC i. Since $\mathcal{M}^i$ includes SCs $\{1, \dots, i\}$ and SC i is the target in $\mathcal{M}^i$, we use o_i to record the number of shared VMs (not from itself) used by SC i, and we use a_i to record the number of shared VMs (not from SC i) used by SCs $\{1, \dots, i-1\}$; thus, $o_i + a_i$ is bounded by $B_i = \sum_{j \neq i} S_j$, the maximum number of VMs shared by SCs $\{1, \dots, K-1\}$.

State transitions. VM allocations in $\mathcal{M}^{i-1}$ affect the results of new states in $\mathcal{M}^i$ after state transitions. Each state transition happens in the period of time between two events (referred as *inter-event period* in the rest of paper), each of which can be a request arrival or a service completion instance. During an inter-event period, each state in $\mathcal{M}^i$ can increase the number of VMs shared by SC i due to SCs in $\mathcal{M}^{i-1}$ allocating VMs in SC i; similarly, the number of requests queued at SC i can decrease due to service completions in $\mathcal{M}^{i-1}$, which allow SC i to utilize the shared VMs. Thus, the probability of going to any destination state from any state in $\mathcal{M}^i$ depends on the probability of being at a specific state in $\mathcal{M}^{i-1}$. Here, we define three *interaction probability vectors* representing the probability of moving from each state (q_i, s_i, o_i, a_i) of $\mathcal{M}^i$ to any other state of $\mathcal{M}^i$, when an event happens, based on the probabilities computed for $\mathcal{M}^{i-1}$:

- $P^A(q_i, s_i, o_i, a_i)$ for an inter-event period preceding an arrival instance;
- $P^D_{loc}(q_i, s_i, o_i, a_i)$ for an inter-event period preceding a local departure instance;
- $P^D_{rem}(q_i, s_i, o_i, a_i)$ for an inter-event period preceding the remote departure instance of a VM allocated at other SCs by SC i.

The detailed computation of these three interaction probability vectors is described in the section "Interaction Probabilities."

Let a_{loc} represent the number of shared VMs from SC i allocated by SCs $\{1, \ldots, i-1\}$ in $\mathcal{M}^{i-1}$, and let a_{rem} represent the number of shared VMs from all others SCs (except SC i) allocated by SCs $\{1, \ldots, i-1\}$ in $\mathcal{M}^{i-1}$, respectively. Then, given a state in $\mathcal{M}^{i-1}$, which can produce the pair (a_{loc}, a_{rem}), $P^A(q_i, s_i, o_i, a_i)_{(a_{loc}, a_{rem})}$, $P^D_{loc}(q_i, s_i, o_i, a_i)_{(a_{loc}, a_{rem})}$, and $P^D_{rem}(q_i, s_i, o_i, a_i)_{(a_{loc}, a_{rem})}$ represent the probability of allocating VMs (a_{loc}, a_{rem}) in vectors $P^A(q_i, s_i, o_i, a_i)$, $P^D_{loc}(q_i, s_i, o_i, a_i)$, and $P^D_{rem}(q_i, s_i, o_i, a_i)$ respectively, after an event happens in the state (q_i, s_i, o_i, a_i) of $\mathcal{M}^i$. The legal combinations of the pair (a_{loc}, a_{loc}) are bounded by the state (q_i, s_i, o_i, a_i) in $\mathcal{M}^i$, which will be described in the section "Initial State Distribution". For simplicity, we use ${P^A}_{(a_{loc}, a_{rem})}$, $P^D_{loc(a_{loc}, a_{loc})}$, and $P^D_{rem(a_{loc}, a_{loc})}$ to represent the probability of VM allocations in $\mathcal{M}^{i-1}$ in the rest of paper.

Transitions for $\mathcal{M}^1$. In $\mathcal{M}^1$, there is only one SC, and no other model affecting the transitions; thus, $s_1 = a_1 = 0$, and

$$(q_1, 0, o_1, 0) \xrightarrow{\lambda} (q_1 + 1, 0, o_1, 0) \qquad \text{if } q_1 < N_1$$

$$(q_1, 0, o_1, 0) \xrightarrow{\lambda} (q_1, 0, o_1 + 1, 0) \qquad \text{if } q_1 \geq N_i \wedge o_1 < B_1$$

$$(q_1, 0, o_1, 0) \xrightarrow{\lambda \cdot P^{NF}(q_1, N_1, Q_1)} (q_1 + 1, 0, o_1, 0) \qquad \text{if } q_1 \geq N_i \wedge o_1 = B_1$$

$$(q_1, 0, o_1, 0) \xrightarrow{\min(q_1, N_1)\mu} (q_1 - 1, 0, o_1, 0) \qquad \text{if } q_1 > 0$$

$$(q_1, 0, o_1, 0) \xrightarrow{o_1 \mu} (q_1, 0, o_1 - 1, 0) \qquad \text{if } o_1 > 0$$

Transitions for $\mathcal{M}^i$. Any transition in $\mathcal{M}^i$ with $i > 1$ depends on interaction probability vectors from $\mathcal{M}^{i-1}$. Given any pair (a_{loc}, a_{rem}) from states in $\mathcal{M}^{i-1}$, the transitions corresponding to a request arrival instance at state (q_i, s_i, o_i, a_i) in $\mathcal{M}^i$ fall into one of the following cases:

C_1: The new request can use a VM at SC i when there is at least one free VM at SC i, even after considering a_{loc} and a_{rem} from $\mathcal{M}^{i-1}$ during the arrival period:

$$(q_i, s_i, o_i, a_i) \xrightarrow{\lambda \cdot {P^A}_{(a_{loc}, a_{rem})}} (q_i + 1, a_{loc}, o_i, a_{rem})$$

for all $q_i + a_{loc} < N_i$ such that $(a_{loc} \leq S_i) \wedge (o_i + a_{rem} \leq B_i)$.

C_2: The new request uses a VM from other SCs. This situation arises when SC i has no idle VMs prior to this arrival instance, but other SCs can provide at least one VM during the preceding inter-event period:

$$(q_i, s_i, o_i, a_i) \xrightarrow{\lambda \cdot {P^A}_{(a_{loc}, a_{rem})}} (q_i, a_{loc}, o_i + 1, a_{rem})$$

for all $q_i + a_{loc} \geq N_i$ and $o_i + a_{rem} + 1 \leq B_i$.

C_3: The new request must be queued or forwarded to a public cloud due to no available shared VMs in the federation, where all VMs have been occupied during the previous or current inter-event period by requests from other SCs:

$$(q_i, s_i, o_i, a_i) \xrightarrow[\cdot P^{NF}(q_i, V_i, Q_i)]{\lambda \cdot {P^A}_{(a_{loc}, a_{rem})}} (q_i + 1, a_{loc}, o_i, a_{rem})$$

for all $q_i + a_{loc} \geq N_i$ and $o_i + a_{rem} = B_i$. $V_i = N_i - s_i + o_i$ is the number of busy VMs currently used by SC i.

Given any pair (a_{loc}, a_{rem}) for states in $\mathcal{M}^{i-1}$, the transitions corresponding to a service completion instance at SC i for its own customers fall into one of the following cases:

C_4: The departure is from VMs of SC i allocating to SC i. If there is at least one job queued in SC i, the freed VM will be used by SC i directly:

$$(q_i, s_i, o_i, a_i) \xrightarrow{V_i \mu \cdot P^D_{loc(a_{loc}, a_{rem})}} (q_i - 1, a_{loc}, o_i, a_{rem}),$$

where $V_i = \min(q_i, N_i - s_i)$ is the number of busy VMs used by SC i, for all $q_i + a_{loc} > N_i$. However, if there are no queued requests in SC i, the freed VM will be allocated to other SCs, which have queued jobs:

$$(q_i, s_i, o_i, a_i) \xrightarrow{V_i \mu \cdot P^D_{loc(a_{loc}, a_{rem})}} (q_i - 1, a_{loc} + 1, o_i, a_{rem}),$$

for all $q_i + a_{loc} \leq N_i$. If none of other SCs have queued requests, the transition will be expressed like the one when SC i has queued requests.

C_5: The departure is from VMs of other SCs allocating to SC i. If there are no queued jobs in any SCs, the freed VM will be returned directly:

$$(q_i, s_i, o_i, a_i) \xrightarrow{o_i \mu \cdot P^D_{rem(a_{loc}, a_{rem})}} (q_i, a_{loc}, o_i - 1, a_{rem})$$

for all $q_i + a_{loc} \leq N_i$. If at least one requests queue in SCs $\{1, \ldots, i-1\}$, SC i has to return VMs:

$$(q_i, s_i, o_i, a_i) \xrightarrow{o_i \mu \cdot P^D_{rem(a_{loc}, a_{rem})}} (q_i, a_{loc}, o_i - 1, a_{rem} + 1).$$

However, if none of above conditions happen and there is at least one job queued in SC i, the VM will still be allocated to SC i, for all $q_i + a_{loc} > N_i$:

$$(q_i, s_i, o_i, a_i) \xrightarrow{o_i \mu \cdot P^D_{rem(a_{loc}, a_{rem})}} (q_i - 1, a_{loc}, o_i, a_{rem}).$$

Interaction Probabilities. As mentioned above, the interaction probability describes the probability of different VM allocations from SCs in $\mathcal{M}^{i-1}$ during an inter-event period when we construct $\mathcal{M}^i$. Steady-state and transient analysis (see Appendix B) of the total number of occupied VMs at

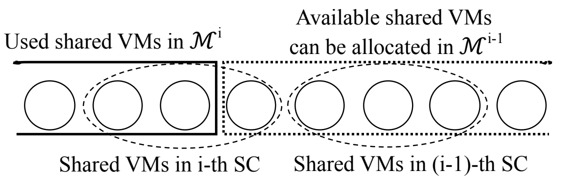

Used shared VMs in $\mathcal{M}^i$
Available shared VMs can be allocated in $\mathcal{M}^{i-1}$
Shared VMs in i-th SC
Shared VMs in (i-1)-th SC

Fig. 4. Possible VM usages are based on the state of $\mathcal{M}^i$

$\mathcal{M}^{i-1}$ can be leveraged for the computation of interaction probabilities between $\mathcal{M}^i$ and $\mathcal{M}^{i-1}$.

Initial State Distribution. The initial state distribution p_0 for $\mathcal{M}^{i-1}$ depends on the VM allocations in the current state of $\mathcal{M}^i$. For instance, as illustrated in Fig. 4, when state (q_i, s_i, o_i, a_i) in $\mathcal{M}^i$ allocates 2 shared VMs from SC i, the state $(q_{i-1}, s_{i-1}, o_{i-1}, a_{i-1})$ of initial state distribution p_0 for $\mathcal{M}^{i-1}$ can only allocate up to 1 shared VMs from SC i.

Given the current state (q_i, s_i, o_i, a_i) of $\mathcal{M}^i$, the initial state distribution of $\mathcal{S}^{i-1}$ for transient analysis is $\pi^X_{[(q_i,s_i,o_i,a_i)]}$, where $[(q_i, s_i, o_i, a_i)]$ is the subset of all states in $\mathcal{S}^{i-1}$, which satisfy the restriction from the VM usage of state (q_i, s_i, o_i, a_i) in $\mathcal{M}^i$. Then, interaction probability vectors are the outcome of initial state distribution times the transient state change during the average inter-arrival time or departure time:

$$P^A(q_i, s_i, o_i, a_i) = \left(\pi^X_{[(q_i,s_i,o_i,a_i)]} P(\tfrac{1}{\lambda_i})\right);$$
$$P^D_{loc}(q_i, s_i, o_i, a_i) = \left(\pi^X_{[(q_i,s_i,o_i,a_i]} P(\tfrac{1}{V_i\mu})\right);$$
$$P^D_{rem}(q_i, s_i, o_i, a_i) = \left(\pi^X_{[(q_i,s_i,o_i,a_i)]} P(\tfrac{1}{o_i\mu})\right),$$

where V_i is the number of used VMs in (q_i, s_i, o_i, a_i). The initial state distribution $\pi^X_{[(q_i,s_i,o_i,a_i)]}$ is computed through the concept of Conditional Probability Distribution [21] (see Appendix C).

Performance Parameters. Given that π^i represents steady-state probabilities of $\mathcal{M}^i$, the performance parameters can be computed as follows:

$$\overline{I^{S_i}_i} = \sum s_i * \pi^i_{(q_i,s_i,o_i,a_i)}; \qquad \overline{O^{S_i}_i} = \sum o_i * \pi^i_{(q_i,s_i,o_i,a_i)};$$
$$\overline{P^{S_i}_i} = \lambda_i \cdot \left(\sum (1 - P^{NF}(q'_i, V_i, Q_i)) \cdot \pi^i_{(q_i,s_i,o_i,a_i)}\right),$$

where $q'_i = q_i - (N_i - s_i)$ and $V_i = N_i - s_i + o_i$.

IV. Market-based Model

Next, we develop the empirical market-based model for *SC-Share* to determine appropriate sharing decisions for each SC. We first formulate SC utility functions that take performance characteristics (as computed above) into consideration. We then focus on the details of the game and on the notion of market efficiency.

A. SC Utilities

As discussed before, SCs participate in the federation in order to (a) obtain resources and satisfy SLAs at prices cheaper than public clouds, and (b) sell idle resources to other SCs for profit, similar to spot instances sold by Amazon AWS [22]. To this end, we define SC i utility $U^{S_i}_i$ (see Eq. (3) below) as the ratio of (a) the change in net cost of an SC when it participates in the federation versus when it does not to (b) the change in utilization of an SC when it participates in the federation, versus when it does not, to:

$$U^{S_i}_i = \frac{\left(max(C^0_i - C^{S_i}_i, 0)\right)^2}{\left(\rho^{S_i}_i - \rho^0_i\right)^\gamma} \qquad 0 \le \gamma \le 1, \tag{3}$$

where C^0_i is the cost for SC i when it does not participate in the federation, $C^{S_i}_i$ is the cost for SC i when it shares a maximum of S_i VMs, ρ^0_i is the system utilization of not participating in the federation, and $\rho^{S_i}_i$ is the utilization of SC i when it shares a maximum of S_i VMs. It is evident that an SC will try to minimize its cost for satisfying SLAs; thus, we consider the cost reduction as the numerator of Eq. 3. We consider the increment in SCs' utilizations (the denominator of Eq. (3)) because SCs always want to keep utilizing their resources in a certain level (the system utilization of SCs should always increase since all of them have to share resources to others in order to participate in the federation). For instance, an SC would want to increase the amount of shared VMs (i.e., increasing its system utilization) to obtain higher profit from the cooperation, but would like to decrease the amount of shared VMs whenever its high system utilization makes it forward more requests to a public cloud (i.e., the rate of cost reduction starts to decrease). Here, γ in Eq. (3) reflects the importance SC i places on utilization, where $\gamma = 0$ means SC i only considers cost reduction, referred as UF_0 in the rest of the paper, and $\gamma = 1$ means SC i considers the marginal cost reduction for utilization changes as the most important factor, referred as UF_1 in the rest of the paper (we make $\gamma = 1$ represent the most importance because $0 < \rho^{S_i}_i - \rho^0_i \le 1$). We choose such a structure for $U^{S_i}_i$ so that an SC will always pursue to reduce the cost, and the marginal utility is linear in $(C^0_i - C^{S_i}_i)$. In the experiments, we assume all SCs in the federation will follow the same γ setting since different values of γ produce different scales of utilities.

ALGORITHM 1: Proposed repeated game among SCs

```
Input: $C^P_i, C^G_i$, SC $\{1, ..., K\}$
Output: $\{S_1, ..., S_K\}$
SC $i$ has $N_i$ VMs, arrival rate $\lambda_i$ and SLA requirement $Q_i$;
In round $r = 0$, SC VM sharing vector is $\{S^{(0)}_1, ..., S^{(0)}_K\}$;
do
    $r = r + 1$;
    foreach $i \in 1, ..., K$ do
        $S^{(r)}_i \leftarrow$ the shared VM number which maximizes
        SC $i$'s utility based on $S^{(r-1)}_j, \forall j \ne i, C^P_i, C^G_i$;
    end
while $\exists i \in \{1, ..., K\}, S^{(r)}_i \ne S^{(r-1)}_i$;
$\{S^{(r)}_1, ..., S^{(r)}_K\}$ is the equilibrium point;
```

B. Non-Cooperative Game Among SCs

Game Setting. We implement a *finite repeated non-cooperative game*, where the strategy parameter of each SC i (S_i) is the maximum number of VMs shared with other SCs at any given time. Here, we adapt the concept of *fictitious* *play* [23], and assume that each SC does not need to know the utility functions of others. SC i determines S_i based on the performance characteristics achieved through sharing with others in the previous round of the game, resulting in a corresponding cost of maintaining the required SLAs. Algorithm 1 describes the details of our non-cooperative repeated game. In the initial round (without knowledge of other SCs' behavior), each SC makes an initial sharing decision arbitrarily, and begins sharing VMs with other SCs. Given the solution of the performance model (which takes $\{S_1^{(0)}, ..., S_K^{(0)}\}$ as input), each SC maximizes its utility, to determine $S_i^{(1)}$, its sharing decision for the next round. Using its new sharing decision and those from other SCs ($S_j^{(1)}, \forall j \neq i$) from the previous round, SC i maximizes its utility again, to determine a new sharing decision $S_i^{(2)}$. This continues until the game converges to an *equilibrium point*, as explained next.

Analyzing Market Equilibria. A Nash equilibrium point of our proposed repeated game represents the game state at which no SC has any incentive to improve its sharing decision [24]. In our work, we are primarily interested in *pure strategy* Nash Equilibria (NE) [24] as it is more practical to implement and realize (see Appendix **??** for a detailed reasoning). More importantly, we have designed utility functions for the SCs that take as arguments, parameters that are practically relevant to our problem, and are expressions that best reflect SC satisfaction levels. However, in the process, we could not strictly preserve salient mathematical properties related to the utility functions (see Appendix **??**) that allow us to derive closed form results about Market Equilibria (ME) from existing seminal works in micro-economic theory, *resulting us taking an experimental stance to characterize equilibria.* Below, we briefly rationalize our stance in the light of the inapplicability of seminal game-theory theorems in characterizing ME in our work. *A detailed explanation of our rationale (along with a description of the salient mathematical properties) is in the Appendix* **??**.

First, deriving closed form results for our work via the seminal result by *Nash* is not possible due to us (a) dealing with only pure strategy NE, and (b) the utility for an SC might not be *quasi-concave* [25] in general cases. *Second*, deriving closed form results for our work via the seminal result by *Debreu, Fan, and Glicksberg* (derived independently) [26]–[28] in relation to pure strategy NE is not possible due to (a') the quasi-concavity assumption might not always be satisfied (for the peer utility function), which in turn might not guarantee pure strategy NE (violating theorem assumptions), and (b') strategy sets in many applications (including specialized versions of our application setting, i.e., the number of shared VMs is discrete in nature) might not be *continuous* and infinite [24], in which case, we would have to go back to using Nash's theorem to guarantee mixed strategy NE (which we do not aim to achieve). Finally, deriving closed form results for our work via the strong seminal result by *Dasgupta and Maskin* [29] (that also accounts for discontinuous utility functions) is not possible due to the same reasons in (a) and (b) above.

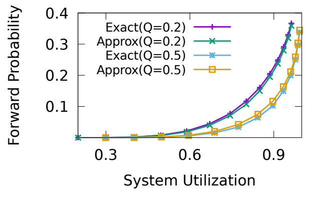
Despite barriers to closed form analysis, we observe through simulation results (see below) the existence of pure strategy NE for infinite strategy spaces (simulated in a discrete manner, thereby becoming a finite game in simulation), and for *non quasi-concave* SC utility functions. Thus, at least from the experimental results, we observe that for our work, (i) it is not necessary (via the theorem of *Nash*) for quasi concavity to hold for a pure strategy (also discounting the guarantee of only a mixed strategy via Nash's theorem) Nash equilibrium to exist, and (ii) it is not necessary (via the theorem of *Debreu et.al.,*) for quasi concavity to hold for a pure strategy (also discounting the infinite strategy space assumption via the theorem by *Debreu. et.al*, as the simulation is discrete in nature) Nash equilibrium to exist.

Reaching Market Equilibria. As addressed above, since we could not afford a mathematical proof, in this work, we simulate the game in Algorithm 1 and determine the equilibrium point empirically for a specific price setting (C_i^P and C_i^G). A traditional heuristic to search for one such equilibium point in the game is the numerical *Tâtonnement process* [30] that is based on the principle of gradient descent. In our work, due to the discrete nature of the SC strategy elements (e.g., # of VMs to share), we need a *discrete* version of a Tâtonnement process to arrive an equilibrium point. However, the design and analysis of such a process has been shown to be quite challenging [31]; moreover there is no existing discrete Tâtonnement process to the best of our knowledge. Thus, in our market-based model, we use the well known non-gradient based *Tabu Search* heuristic [32] to search for an equilibrium value of $S_i^{(r)}$, and reach the global optimum in most cases.

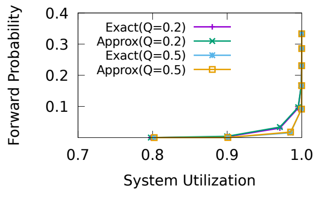
Fairness Among SCs. A joint social end goal, serving as a benchmark of how well selfish non-cooperative SCs participate in the federation w.r.t. their sharing behavior, is to (a) reach a certain level of fairness (see below for details) among SCs in terms of their utilities, and (b) maximize their individual utilities at ME. It is important to note here that if we only compare the fairness allocations among SCs, the scenario where all SCs share nothing with others can also be a most fair allocation, but it results in sub-optimal individual utilities (at times an individual utility of zero for the SCs) at ME (see V-B). To achieve our joint social end goal, we need to find a specific price setting (the ratio of C_i^G and C_i^P) that enables all SCs to maximize their utilities through sharing VMs while at the same time maintaining an appropriate level of fairness. In regard to adopting an appropriate fairness measure, we consider in our work the widely popular notion of weighted α-fairness [33] to combine individual SC utilities $U_i^{S_i}$ through the function

$$W(\alpha, \overrightarrow{S_i}, \overrightarrow{U_i^{S_i}}) = \begin{cases} \sum_{k=1}^{K} S_i \frac{(U_i^{S_i})^{1-\alpha}}{1-\alpha} & \alpha \geq 0, \alpha \neq 1; \\ \sum_{k=1}^{K} S_i \log U_i^{S_i} & \alpha = 1. \end{cases} \quad (4)$$

Here, S_i, the maximum number of shared VMs, is the weight used to combine the α-fairness metric of each SC i, while the parameter α controls the fairness of utility allocations among SCs. In this work, we evaluate three popular α-fairness utility functions, achieving different trade-offs between fairness and economic efficiency: (i) $\alpha = 0$, which gives the utilitarian function [34] (denoting minimum fairness), (ii) $\alpha = \infty$, which



(a) 10 VMs



(b) 100 VMs

Fig. 5. Comparing the result of forwarding estimation in 10 and 100 VMs with QoS = 0.2 and 0.5.

results in max-min fairness, and (iii) $\alpha = 1$, which gives proportional fairness. For each fairness function defined by α, our goal is to find the best price setting that motivates SCs, based on their system loads, to participate in the federation and share more of their VMs, i.e., thereby achieving higher values of α-fair functions.

V. Evaluation and Validation

We first validate the accuracy of our performance model, the results of which are needed as input parameters to the market-based model. To this end, we compute the solution of our approximate model (in Section III) numerically, and compare it to the solution of the exact model (computed through a C++-based simulator). We then use our market-based model to investigate *how the price of using shared VMs from other SCs affects achieving higher summation of weighted utilities*.

A. Performance model validation

SC without Sharing Resources. Here, we start with the accuracy evaluation of our forward probability estimation in Section III-A, since this is a measure used by all other models. Moreover, to demonstrate that SCs have more incentives to participate in the federation, we compare the results of two clouds, which have 10 and 100 VMs respectively, with the SLAs of $Q_i = 0.2$ and $Q_i = 0.5$ under various Poisson arrival rates; each request has an exponential service time with rate 1. In order to correctly compare the results among two SCs, in Fig. 5, we show the estimated forward probability under different system utilizations (by increasing the arrival rate). As shown in the figure, for both clouds, the probability of forwarding is higher for smaller QoS values, and our estimation properly predicts the forward probability under different settings. It is easy to see that the cloud with fewer VMs has higher forwarding probability under the same system utilization. Thus, if an SC does not want to increase its investments in infrastructure, it needs some mechanism to decrease its forwarding probability to reduce the cost of satisfying SLAs. In the following experiments, each SC in the federation has 10 VMs by default with exponential service time with rate $\mu = 1$ and QoS $Q_i = 0.2$.

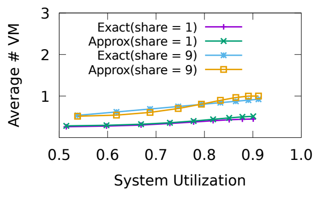
Approximate Model. In this section, we performed extensive experiments to validate the accuracy of the approximate model presented in Section III-C. Here, we want to investigate how well our approximate model performs as a function of the different number of shared VMs and system utilizations.

We begin with a 2-SC federation scenario. We fix the arrival rate of one SC to 7 and the number of shared VMs to 5 (total 10 VMs), and vary the number of shared VMs and system load (by changing the arrival rate) of another SC, referred to as target SC. Figures 6a and 6b illustrate the performance metrics of interest when the target SC shares 1 and 9 VM(s) under different system loads. (Due to lack of space, we omit $\overline{P_i^{S_i}}$ as its estimation remains accurate.) As shown in the figure, the exact and approximate $\overline{I_i^{S_i}}$ and $\overline{O_i^{S_i}}$ are nearly the same when the target SC shares very few VMs. The inaccuracy of our approximate model grows when the target SC shares more VMs (as compared to a scenario with 1 shared VM), but is still within 10%. Thus, the difference between $\overline{I_i^{S_i}}$ and $\overline{O_i^{S_i}}$ (see Eq. 1) remains accurate (within 10% of the exact solution).

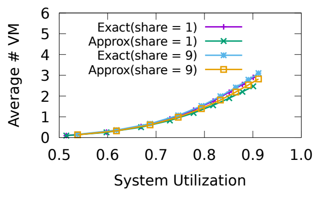
We now illustrate how the approximation error grows in larger systems. Firstly, we consider a 10-SC (each of which has a total of 10 VMs) federation scenario, and fix 9 SCs' settings, which have the following number of shared VMs $(3, 3, 3, 2, 2, 2, 1, 1, 1)$, and corresponding arrival rate $(7, 7, 7, 8, 8, 8, 9, 9, 9)$. Figures 6c and 6d illustrate the performance metrics of interest when the target SC shares 1 and 5 VM(s) under different system loads. We still observe that the difference between the exact and approximate $\overline{I_i^{S_i}}$ and $\overline{O_i^{S_i}}$ remains small (within 10% of the exact solution) when the system utilization is lower than 0.8 (within 20% when the system utilization is lower than 0.9). Generally, we can observe that the results of approximated $\overline{I_i^{S_i}}$ are under-estimated when the system has very high utilization because our approximate model breaks the direct relationship between the target SC and all other SCs (we only consider the connection between SC i and SC $i-1$); thus, the target SC might under-estimate the number of queued requests at all other SCs. For the same reason, the results of approximated $\overline{O_i^{S_i}}$ are over-estimated. However, the difference between $\overline{I_i^{S_i}}$ and $\overline{O_i^{S_i}}$ remains accurate (within 20% of the exact solution) when the system utilization is lower than 0.9. Second, we consider again a 2-SC federation scenario, with 100 VMs per SC. We fix the the number of shared VMs at 10 for both SCs, and vary system load for both of them. Figures 6e and 6f illustrate the performance metrics of interest when one SC has system utilization of 0.8 and 0.9 under different system loads of the target SC. We still observe that the difference between $\overline{I_i^{S_i}}$ and $\overline{O_i^{S_i}}$ remains accurate (within 20% of the exact solution) when the system utilization of the target SC is lower than 0.9.

Computational Complexity. Although our model has larger errors in some cases, our approximate model saves up to 10 times in computation time compared to the exact model when each SC shares a small number of VMs. However, the computation time of the exact model increases significantly (more than half of a day) when the system load of any SC reaches is greater than 0.9 or is lower than 0.6, particularly when SCs have a significant number of VMs, while our approximate model's computation time remains nearly the same since the parameters of our approximate model are tuned to achieve small error by default for high system utilizations (see Appendix B).

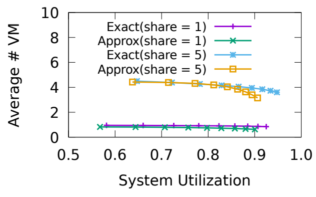
Summary. Our extensive experiments indicate that our ap-



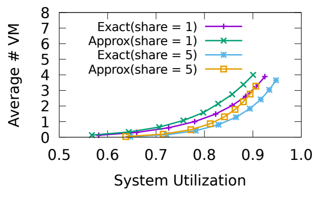
(a) $\overline{I_i^{S_i}}$, 2 SCs, 10 VMs



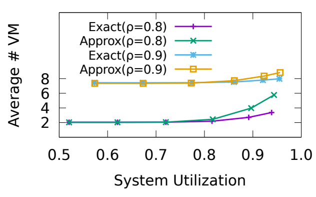
(b) $\overline{O_i^{S_i}}$, 2 SCs, 10 VMs



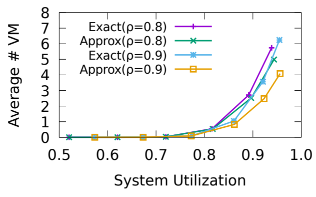
(c) $\overline{I_i^{S_i}}$, 10 SCs, 10 VMs



(d) $\overline{O_i^{S_i}}$, 10 SCs, 10 VMs



(e) $\overline{O_i^{S_i}}$, 2 SCs, 100 VMs



(f) $\overline{O_i^{S_i}}$, 2 SCs, 100 VMs

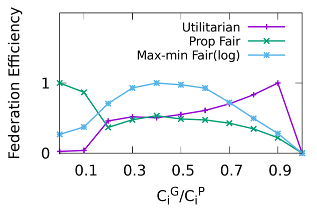
Fig. 6. Validating approx. perf. model (2 SCs and 10 SCs)

proximate model estimates $\overline{I_i^{S_i}}$ and $\overline{O_i^{S_i}}$ within 20% of the exact solution, under a variety of scenarios, while saving significant computation time. More importantly, the accuracy of the *difference* between $\overline{I_i^{S_i}}$ and $\overline{O_i^{S_i}}$, and $\overline{P_i^{S_i}}$, which are the parameters needed by the market-based model, are within 10% of the exact solution when the system utilization is reasonable. Overall, we believe that our approximate model is useful in estimating performance characteristics of the federation, as needed in the market-based model.

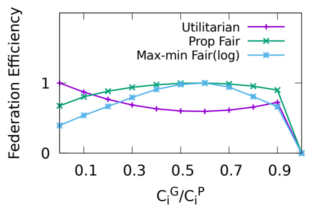
B. Market-based Model Evaluation

In this section, we perform extensive experiments to investigate how the value $\frac{C_i^G}{C_i^P}$ (Section II-B), the fairness level (Section IV-B) used by the federation, and utility functions (Section IV-A) chosen by SCs affect the criteria for SCs to participate in the federation through our proposed market-based model (Section IV). We focus on 3-SC scenarios (as a representative example) in the evaluation, as illustrated in Figure 7. Here, we display the ratio of the achieved value of the W metric (see subsection "Fairness Among SCs") to the (empirical) market efficient value of the W metric, as a measure of *federation efficiency*, for a given mixture of SC utility functions. To circumvent anomalies of scale due to representing results from different choices of the W metrics in the same figure, we take the logarithmic values of the max-min W metric. If no SCs are willing to participate in the federation, we display it as zero federation efficiency (since the value of the W metric is always greater than zero).

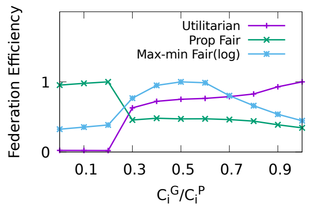
We first consider scenarios where the 3 SCs have significantly *different system loads* ($\rho_i = 0.58, 0.73, 0.84$). Fig. 7a illustrates the case where all SCs choose UF_0 ($\gamma = 0$) as their utility function; Fig. 7b illustrates the case where



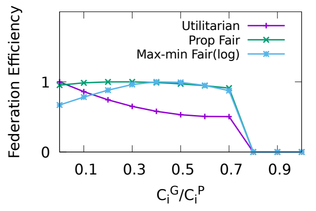
(a) All SCs with UF_0



(b) All SCs with UF_1



(c) All SCs with UF_0



(d) All SCs with UF_1

Fig. 7. Market results in 3-SC scenarios: (a)(b) are results where 3 SCs have $\rho_i = 0.58, 0.73, 0.84$, (c) is the result where 3 SCs have $\rho_i = 0.73, 0.79, 0.84$, (d) are the results where 3 SCs have $\rho_i = 0.49, 0.58, 0.66$

all SCs choose UF_1 ($\gamma = 1$) as their utility function. As shown in the figures, if all SCs chooses UF_0, the utilitarian W metric increases with increase in C_i^G/C_i^P (except when C_i^G/C_i^P is nearing 1), since the SCs choosing UF_0 as their utility are incentivized to share more VMs to reduce their net cost. When C_i^G/C_i^P is nearing 1, the federation cannot be formed because SCs with high utilizations do not reduce cost through using shared VMs, compared to that when resorting to a public cloud, and low utilization SCs do not generate enough demand to make high utilization SCs remain profitable. If all SCs use UF_1, they would only share 1 VM with others even when C_i^G/C_i^P increases because, in our setting, the increase in marginal cost reduction with increase in number of shared VMs is not sufficient to encourage SCs to contribute more VMs. Moreover, since all SCs only shared 1 VM when they use UF_1, both proportional W metric and max-min W metric achieve the same maximum state (due to the same weight for all SCs in Eq. (4)), as shown in Fig. 7b. In other cases, the results of the proportional W metric depend on the behavior of the lower utilization SCs. If these SCs choose UF_0, their cost reductions with increase in number of shared VMs are greater than high utilization SCs; thus, the maximum proportional W metric can only happen when all SCs share few VMs.

In Fig. 7c, we consider scenarios where 3 SCs have similarly *high system loads* ($\rho_i = 0.73, 0.79, 0.84$), where all of them consider UF_0. In this scenario, the results are similar to the cases in Fig. 7a; however, unlike the scenario where SCs having significant different utilizations *are not incentivized* to join the federation when $C_i^G/C_i^P = 1$, SCs in a scenario when they have similar high utilizations, are *incentivized* to cooperate when $C_i^G/C_i^P = 1$. This is because high utilization SCs share similar number of VMs with each other, resulting in canceling out the cost of using shared VMs. In Fig. 7d, we consider scenarios where 3 SCs have similarly *median system loads* ($\rho_i = 0.49, 0.58, 0.66$), where all of them consider UF_1. The results in these scenarios are similar to what we have discussed above, however, we observe the federation cannot be formed when C_i^G/C_i^P is beyond ≈ 0.8. This is because all low utilization SCs do not generate enough revenue from their incoming VM demand from other SCs to offset their costs of using shared VMs from other SCs.

Summary. Our extensive experimental evaluation indicates *three* C_i^G/C_i^P regions of operation to maximize various W metrics. When *maximizing proportional fairness based W metric* is the goal of the federation, the value of C_i^G/C_i^P should be set in the *lower range* of C_i^G/C_i^P (between 0 and 0.3 in our example setting). When *maximizing max-min fairness based W metric* is the goal of the federation, the value of C_i^G/C_i^P should be set in the *middle range* of C_i^G/C_i^P (between 0.3 and 0.7 in our example setting). Finally, when *maximizing utilitarian W metric* is the goal of the federation, the value of C_i^G/C_i^P should be set in the *high range* of C_i^G/C_i^P (between 0.7 and 1 in our example setting). However, the utilitarian setting also runs the risk of breaking the federation at a certain high value of C_i^G/C_i^P at which no SC would be willing to cooperate.

VI. Related Work

We give an overview of efforts related to our work and highlight the relevant differences. Works on hybrid cloud computing [9], [10] are related to ours in that they allow private clouds (or small-scale cloud providers) to outsource their requests to large-scale public providers. However, since that can potentially be costly for a small-scale provider, our work differs in that it focuses on the *cooperative framework*, while minimizing the use of public clouds.

Efforts that are closer to ours include incentivizing a federation among clouds [13], [14], [16], [17], [35] as well as some notion of resulting sharing mechanisms' efficiency [12], [15]. For instance, [14] proposes a decentralized cloud platform *SpotCloud* [18] - a real-world system allowing customers or SCs to sell idle compute resources to others at their specified prices - and presents a resource pricing scheme (resulting from a repeated seller game) plus an optimal resource provisioning algorithm. An earlier effort [17] characterizes the cloud federation to help cloud providers maximize their profits based on the system loads, while [16] adopts cooperative game theoretic approaches to model the cloud federation and study the motivation for cloud providers to participate in the federation. A model of federated cloud providers is presented in [13], which proposes to incorporate both, historical and expected future revenue into VM sharing decisions, in order to maximize an SC's profit. A decentralized cloud is proposed in [12] to study cooperation among SCs under varying workloads, employing various cooperation strategies, to reduce the request rejection rate (i.e., the efficiency metric in [12]). This work indicates that an SC is likely to collaborate with other SCs that have workload patterns that differ from its own. Another effort [15] studies the sharing policies for cloud providers to trade-off the approaches of outsourcing resources and rejecting less profitable in order to increase resource utilization and profit. Authors in [35] propose the a hierarchical cooperative game theoretic model for better resources integration and achieving a higher profit in the federation. Another approach [11] designs a decentralized cloud resource sharing platform for grouping resources of various SCs into computational units, in order to serve customers' requests.

Differences and Drawbacks. Our work differs from previous efforts in that we explicitly consider consequences of resource sharing on the resulting performance delivered to customers. In contrast, none of the above efforts explicitly model the system performance under the considered resource sharing environment. They either assume that resources can be reclaimed (when needed), thus resulting in lack of reliability of shared resources or they assume that an analytical performance characterization is possible (but do not propose a solution to estimate it). Such an analytical characterization is an important contribution of our work. To the best of our knowledge, this is the first work addressing the explicit interactions between performance model and economic model. Moreover, unlike previous efforts, that adopt the cooperative game theoretic approach, our work studying the non-cooperative game is more practical since likely no SC would be willing to share their utility specific information with others.

VII. Conclusion and Future Work

In this paper, we proposed *SC-Share* for small-scale clouds (SCs) to enable them to share their resources in a profitable manner while maintaining customer SLAs. Our framework is based on two (interacting) models: (i) an approximate performance model with an efficient solution that is able to produce sufficiently accurate estimates of performance characteristics of interest; and (ii) a market-based model that results in sharing policies which properly incentivize SCs to participate in the federation while achieving market success. *SC-Share* can suggest different price settings in different federations in order to achieve sufficient market efficiency. Moreover, *SC-Share* shows that even when the price of shared VMs is equal to the price of using a public cloud, a federation can still be formed under certain criteria.

SC-Share evaluates the resource sharing benefits among SCs by accounting only for the cost of using VMs. However, there are other parameters that *SC-Share* could account for in evaluating resource sharing benefits: (i) privacy concerns/risks of sharing/forwarding resources within cloud entities, (ii) data transmission costs for forwarding VM requests among cloud entities, and (iii) power consumption costs of running physical servers hosting VMs. We plan to incorporate these parameters into the *SC-Share* framework as part of future work.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, 2010.

[2] "Amazon AWS," http://aws.amazon.com.

[3] "Google Compute Engine," https://cloud.google.com/compute.

[4] "Microsoft Azure," https://azure.microsoft.com.

[5] "Blippex - Why we moved away from AWS," http://blippex.github.io/updates/2013/09/23/why-we-moved-away-from-aws.html.

[6] "As Private Cloud Grows, Rackspace Expands Options Inside Equinix," https://blog.equinix.com/blog/2016/05/16/as-private-cloud-grows-rackspace-expands-options-inside-equinix/.

[7] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *ACM SIGCOMM Computer Communication Review*, 2012.
[8] "It's clear to Linode: There's a market to bring cloud services to small companies," http://www.njbiz.com/article/20131104/NJBIZ01/311019994/It
[9] H. Zhang, G. Jiang, K. Yoshihira, and H. Chen, "Proactive workload management in hybrid cloud computing," *Network and Service Management, IEEE Transactions on*, 2014.
[10] M. Shifrin, R. Atar, and I. Cidon, "Optimal scheduling in the hybrid-cloud," in *Integrated Network Management (IM 2013)*. IEEE, 2013.
[11] O. Babaoglu, M. Marzolla, and M. Tamburini, "Design and implementation of a P2P Cloud system," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. ACM, 2012.
[12] H. Zhuang, R. Rahman, and K. Aberer, "Decentralizing the cloud: How can small data centers cooperate?" in *Peer-to-Peer Computing (P2P), 14-th IEEE International Conference on*, 2014.
[13] N. Samaan, "A novel economic sharing model in a federation of selfish cloud providers," *Parallel and Distributed Systems, IEEE Transactions on*, 2014.
[14] H. Wang, F. Wang, J. Liu, D. Wang, and J. Groen, "Enabling Customer-Provided Resources for Cloud Computing: Potentials, Challenges, and Implementation," *IEEE Transactions on Parallel & Distributed Systems*, 2014.
[15] 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 *High Performance Computing and Communications (HPCC), IEEE 13th International Conference on*. IEEE, 2011.
[16] M. M. Hassan, M. S. Hossain, A. J. Sarkar, and E.-N. Huh, "Cooperative game-based distributed resource allocation in horizontal dynamic cloud federation platform," *Information Systems Frontiers*, 2014.
[17] Í. Goiri, J. Guitart, and J. Torres, "Economic model of a cloud provider operating in a federated cloud," *Information Systems Frontiers*, 2012.
[18] "SpotCloud," http://www.spotcloud.com/.
[19] "Amazon lightsail," https://amazonlightsail.com/.
[20] Y. Kessaci, N. Melab, and E.-G. Talbi, "A pareto-based metaheuristic for scheduling hpc applications on a geographically distributed cloud federation," *Cluster Computing*, 2013.
[21] P. Billingsley, *Probability and measure*. John Wiley & Sons, 2008.
[22] "Amazon EC2 Spot Instances," https://aws.amazon.com/ec2/spot/.
[23] G. W. Brown, "Iterative solution of games by fictitious play," *Activity analysis of production and allocation*, 1951.
[24] D. Fudenberg and J. Tirole, "Game theory," 1991.
[25] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[26] G. Debreu, "A social equilibrium existence theorem," *Proceedings of the National Academy of Sciences*, 1952.
[27] I. L. Glicksberg, "A further generalization of the kakutani fixed point theorem, with application to nash equilibrium points," *Proceedings of the American Mathematical Society*, 1952.
[28] K. Fan, "Fixed-point and minimax theorems in locally convex topological linear spaces," *Proceedings of the National Academy of Sciences of the United States of America*, 1952.
[29] P. Dasgupta and E. Maskin, "The existence of equilibrium in discontinuous economic games, i: Theory," *The Review of economic studies*, 1986.
[30] H. R. Varian, *Intermediate Microeconomics: A Modern Approach: Ninth International Student Edition*. WW Norton & Company, 2014.
[31] T. Kaizoji, "Multiple equilibria and chaos in a discrete tâtonnement process," *Journal of Economic Behavior & Organization*, 2010.
[32] F. Glover, "Tabu search-part i," *ORSA Journal on computing*, 1989.
[33] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking (ToN)*, 2000.
[34] A. Mas-Colell, M. D. Whinston, J. R. Green *et al.*, *Microeconomic theory*. Oxford university press New York, 1995.
[35] D. Niyato, A. V. Vasilakos, and Z. Kun, "Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach," in *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2011, pp. 215–224.
[36] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1995.
[37] B. L. Fox and P. W. Glynn, "Computing Poisson Probabilities," *Commun. ACM*.

APPENDIX

A. Detailed Model for SC Cooperative $\mathcal{M}$

Table I reports the transition structure for the detailed model $\mathcal{M}$ introduced in Section III-B. Transitions are given for SC i from a generic state

$$(q_1, s_{1,1}, ..., s_{1,K}, ..., q_i, s_{i,1}, ..., s_{i,K}, ..., q_K, s_{K,1}, ..., s_{K,K}).$$

The transition rates include $P^F(V_i, n_i, Q_i)$, which is the probability that a request is forwarded to a public cloud when n_i requests are queued at SC i, all of its $V_i \leq N_i$ available VMs are currently busy, and the maximum allowed waiting time by the SLA is Q_i (see Section III-A for a detailed definition). We also assume a load balancing mechanism in the model: SC i determines with which SC j to share an idle VM by choosing (uniformly at random) among those SCs with the highest number of queued requests.

B. Transient Analysis

To compute transient probabilities, which describe transient changes in the number of VM allocations in CTMC $\mathcal{M}^{i-1}$ over inter-event periods at SC i, we use the method of *uniformization* [36], which decomposes the CTMC into a discrete-time Markov chain (DTMC) and a Poisson process as follows: given the infinitesimal generator Q^{i-1},

- the rate of the Poisson process is $\gamma \geq \max_j \left|q_{jj}^{i-1}\right|$,
- the transition matrix of the DTMC is $P^{i-1} = I + \frac{1}{\gamma} Q^{i-1}$.

Then, the transient probability vector $p^{i-1}(t)$ for $\mathcal{M}^{i-1}$ can be computed for all $t \geq 0$ as $p^{i-1}(t) = p_0 P^{i-1}(t)$, where $P^{i-1}(t) = \sum_{k=0}^{\infty} \frac{e^{-\gamma t}(\gamma t)^k}{k!} (P^{i-1})^k$ is the matrix of transition probabilities for the CTMC (for a given precision ϵ, the summation can be truncated using the Fox and Glynn method [37]). By letting p_0 be equal to the initial state distribution at any time instance, we can compute transient state changes of $\mathcal{M}^{i-1}$.

C. Conditional Probability Distribution

Given X, a discrete random variable over $\mathbb{N} = \{0, 1, \dots\}$, and π^X, its probability mass function, the conditional probability distribution for subset $Y \subseteq X$ is:

$$\pi^X(k) = \begin{cases} P_X(k|k \in Y) = \frac{P(X=k \wedge k \in Y)}{P(k \in Y)} & \text{if } k \in Y, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Next State	Rate	Condition for Transition
$(q_1, s_{1,1}, ..., s_{1,K}, ..., q_i + 1, s_{i,1}, ..., s_{i,K},$ $..., q_K, s_{K,1}, ..., s_{K,K})$	$P(q_i, s_{i,i})\lambda_i$	$(q_i + s_{i,i} < N_i) \vee (q_j + s_{j,j} \geq N_j, \forall j \neq i)$
$(q_1, s_{1,1}, ..., s_{1,K}, ...,$ $q_i, s_{i,1}, ..., s_{i,j} + 1, ..., s_{i,K}, ...,$ $q_j, s_{j,1}, ..., s_{j,j} + 1, ..., s_{j,K}, ...,$ $q_K, s_{K,1}, ..., s_{K,K})$	$\frac{\lambda_i}{\lvert K \rvert}$	$(q_i + s_{i,i} \geq N_i)\wedge$ $(L = \{(q_l, s_{l,l}) \mid q_l + s_{l,l} < N_l, s_{l,l} < S_l\}, \forall l \neq i)\wedge$ $(K = \{(q_k, s_{k,k}) \mid q_k + s_{k,k} = \min_L(q_l + s_{l,l})\})$ $\wedge(q_j, s_{j,j}) \in K$
$(q_1, s_{1,1}, ..., s_{1,K}, ..., q_i - 1, s_{i,1}, ..., s_{i,K}, ...,$ $q_j, s_{j,1}, ..., s_{j,j}, ..., s_{j,K}, ...q_K, s_{K,1}, ..., s_{K,K})$	$\min((N_i - s_{i,i}), q_i)\mu$	$(q_i + s_{i,i} > N_i) \vee (q_j + s_{j,j} \leq N_j, \forall j \neq i)$
$(q_1, s_{1,1}, ..., s_{1,K}, ...,$ $q_i - 1, s_{i,1}, ..., s_{i,j} + 1, ..., s_{i,K}, ...,$ $q_j - 1, s_{j,1}, ..., s_{j,j}, ..., s_{j,K}, ...,$ $q_K, s_{K,1}, ..., s_{K,K})$	$\frac{\min((N_i - s_{i,i}), q_i)\mu}{\lvert K \rvert}$	$(q_i + s_{i,i} \leq N_i) \wedge (s_{i,i} < S_i)\wedge$ $(L = \{(q_l, s_{l,l}) \mid q_l + s_{l,l} > N_l\}, \forall l \neq i)\wedge$ $(K = \{(q_k, s_{k,k}) \mid q_k + s_{k,k} = \max_L(q_l + s_{l,l})\})$ $\wedge(q_j, s_{j,j}) \in K$
$(q_1, s_{1,1}, ..., s_{1,K}, ..., q_i, s_{i,1}, ..., s_{i,j} - 1, ..., s_{i,K},$ $..., q_j, s_{j,1}, ..., s_{j,j} - 1, ..., s_{j,K}, ...q_K, s_{K,1}, ..., s_{K,K})$	$s_{i,j}\mu$	$(q_j + s_{j,j} > N_i) \vee (q_k + s_{k,k} \leq N_k, \forall k \neq j)$
$(q_1, s_{1,1}, ..., s_{1,K}, ..., q_i, s_{i,1}, ..., s_{i,j} - 1, ..., s_{i,K}, ...,$ $q_j, s_{j,1}, ..., s_{j,j}, ..., s_{j,K}, ...,$ $q_m, s_{m,1}, ..., s_{m,j} + 1, ..., s_{m,K}, ...,$ $q_K, s_{K,1}, ..., s_{K,K})$	$\frac{s_{i,j}\mu}{\lvert K \rvert}$	$(q_j + s_{j,j} \leq N_j)\wedge$ $(L = \{(q_l, s_{l,l}) \mid q_l + s_{l,l} > N_l\}, \forall l \neq i)\wedge$ $(K = \{(q_k, s_{k,k}) \mid q_k + s_{k,k} = \max_L(q_l + s_{l,l})\})$ $\wedge(q_m, s_{m,m}) \in K$

TABLE I

STATE TRANSITIONS IN $\mathcal{M}$ FROM STATE $(q_1, s_{1,1}, ..., s_{1,K}, ..., q_i, s_{i,1}, ..., s_{i,K}, ..., q_K, s_{K,1}, ..., s_{K,K})$