# Federated Private Clouds via Broker's Marketplace: A Stackelberg-Game Perspective

Xuanjia Qiu*, Chuan Wu*, Hongxing Li†, Zongpeng Li‡ and Francis C.M. Lau*

*Department of Computer Science, The University of Hong Kong, Hong Kong, {xjqiu,cwu,fcmlau}@cs.hku.hk
†Department of Computer Science, University of California, Davis, U.S., hxlihku@gmail.com
‡Department of Computer Science, University of Calgary, Canada, zongpeng@ucalgary.ca

***Abstract*—More and more enterprises have set up their own private clouds by applying virtualization to their data centers; the benefit is flexible resource supply to different internal demands. Aiming to meet the peak demand in their resource provisioning, private clouds are often under-utilized. A new paradigm has emerged that advocates leasing the spare resources to external users, when and if adequate rental prices are offered. A broker is typically employed which pools the spare resources of multiple private clouds together and leases them to serve external users' jobs. Good mechanisms have yet to be derived for the broker to set the offered prices to buy spare resources from the private clouds, and to schedule jobs on the available resources, such that the economic benefits of both the broker and the private clouds are maximized. The design of the mechanism is especially challenging when we consider the dynamic arrival of users' jobs and volatile availability of spare resources at the private clouds, while aiming at long-term profit optimality. In this paper, we model the interaction between the broker and the private clouds as a two-stage Stackelberg game. As the leader in the game, the broker decides and offers prices for renting VMs of different types from each private cloud. As a follower, each private cloud responds with the number of VMs of each type that it is willing to lease. By combining with the Stackelberg game model we design online algorithms for the broker to set the prices and schedule jobs on the private clouds, and for the private cloud to decide the numbers of VMs to lease, based on the Lyapunov optimization theory. We prove that the broker achieves a time-averaged profit that is close to the offline optimum with complete information on future job arrivals and resource availability, while each private cloud makes their best earning. The proposed online algorithm is carefully evaluated based on usage traces of Google cluster and Amazon EC2.**

## I. Introduction

More and more enterprises have set up their own private clouds. They add virtualization to their data centers such that the available computing resources can be flexibly allocated to different internal jobs at different times, with minimal management efforts. The resources in a private cloud are often provisioned according to the peak demand, which could lead to low utilization (*e.g.*, 10%) at times [1]. A new paradigm is called for, which tries to lend out idle resources to external jobs, for monetary remuneration that can offset the investment and maintenance costs of the private cloud. Internet users on the other hand can enjoy low-price cloud computing [2]. A broker is commonly employed, who pools together spare resources from individual private clouds and publishes resource availability information to the Internet users. In the process, it tries to maintain the anonymity of the resource providers. A prominent example is *SpotCloud* [3], which is a marketplace where computing capacity is sold and bought; buyers can enjoy a lower rate for computing services most of the time, as compared to the spot instance market of Amazon EC2 [4].

Although a promising paradigm for federating private clouds, the important broker's mechanisms need to be carefully designed. There have been studies [5][2] advocating double auctions, where each private cloud decides the resource prices, the users do their bidding based on their willingness-to-pay, and the broker as the auctioneer tries to match the sell and buy prices. With such a mechanism, the private clouds can fully express their valuation of the resources by setting up the prices themselves. Nevertheless, we argue that in reality, a private cloud that wishes to sell its idle resource in a "best-effort" manner, may not like to spend the effort in devising a pricing mechanism, and would passively decide whether to accept an offered price and the amount of resources to lease at this price. To cater for such a practical scenario, we consider a federated cloud marketplace where the broker offers prices and purchases idle resources from private clouds on the fly, and re-sells them to the users.

The broker not only judiciously decides different prices to elicit contribution of computing resources from private clouds in different geographical locations, but also exploits temporal and spatial availability of resources when scheduling users' jobs. Due to the volatility of spare resources from the private clouds, resources from this marketplace are suitable for delay-tolerant workloads, *i.e.*, compute-intensive jobs with relaxed completion deadlines [6], such as scientific computing jobs, weekly batch jobs, background transcoding tasks, etc.

We are interested in the following questions: Given dynamical arrival of jobs and volatile availability of resources, how can the broker efficiently decide resource prices and schedule jobs at each time instant, in order to achieve long-term profit maximization while guaranteeing completion of the jobs within the respective deadlines? Given prices offered by the broker, how should each private cloud determine the types and amounts of resources (*e.g.*, virtual machines) to supply out of its idle pool, in order to maximize its own profit?

The research was supported in part by a grant from Hong Kong RGC under the contract HKU 718513.

To answer these questions, we model the interaction between the broker and the private clouds as a two-stage Stackelberg game [7]. As the leader in the game, the broker decides and offers the best prices for renting VMs of different types from each private cloud that can maximize its profit. As a follower, each private cloud responds with the best number of VMs of each type that it is willing to lease, in order to maximize its own profit. Closely combined with the Stackelberg game model are online algorithms that we design for the broker to decide its prices and to schedule the jobs on the private cloud and for each private cloud to decide the numbers of VMs to lease, which is based on the Lyapunov optimization theory. We prove that the broker achieves a time-averaged profit that is close to the offline optimum with complete information on future job arrivals and resource availability. Putting selfishness of the broker and the private clouds aside, we also design an online social welfare maximization algorithm, and compare the social welfare achieved by our Stackelberg game based algorithm with the optimal social welfare. The proposed online algorithms are carefully evaluated based on usage traces of Google cluster and Amazon EC2.

The remainder of this paper is organized as follows. We discuss related work in Sec. II, introduce the system model and formulate the problem in Sec. III and IV respectively. In Sec. V we present online algorithms for the broker and each private cloud. We carry out a theoretical analysis in Sec. VI and report on the performance evaluation in Sec. VII. We conclude the paper in Sec. VIII.

## II. Related Work

*SpotCloud* [3][2] hosts a cloud capacity market where consumer-provided cloud computing resources are leased to a wide range of buyers. Pricing guidelines are provided to the sellers, and the sellers decide resource prices by themselves, with some unclear mechanisms. Cloud federation has been proposed to achieve resource scalability and to increase individual cloud providers' profits [8] [9], where the focus has been on mutual resource leasing among the clouds. Our federated cloud model is different, where private clouds are not directly connected but pooled by a broker to serve users. In the context of grid computing, broker mechanisms have been investigated for pooling individual grids' capacities to run large tasks [10]. Their emphasis has been on fairness of resource usage among different tasks, instead of individual grids' profit maximization, since the autonomy of grids is less important in grid computing.

Stackelberg game has been used to model interactions among participants in a variety of scenarios, including cloud computing [11][12]. Valerio *et al.* [11] apply the Stackelberg game to model interactions between an IaaS provider and a number of SaaS providers, which rent resources from the IaaS cloud. Hadji *et al.* [12] decide on optimal prices by the cloud provider and the resource demands from the users using a Stackelberg game. Both works consider either fixed or predicted user demands, and achieve utility maximization of the participants in one shot. In contrast, we novelly combine a Stackelberg game with the Lyapunov optimization framework, and design some algorithms that can achieve long-term optimality without information into the future.

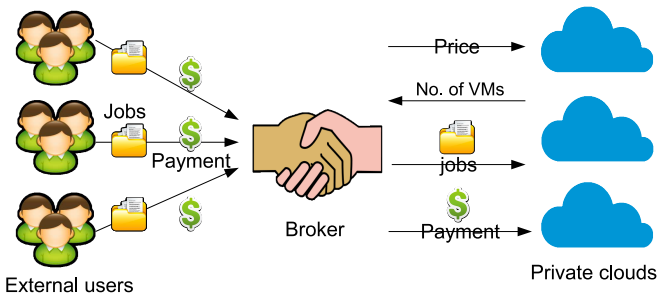


Fig. 1. System model.

Lyapunov optimization [13] has been frequently applied in resource scheduling in wireless networks [14], mobile computing [15] and data center power supply [16]. Li *et al.* [5] integrate Lyapunov optimization with double auctions for joint pricing and scheduling problem in federated clouds. But they investigate a market where the clouds need to make effort to offer prices, and actual trading prices are cleared by double auction, while here we study a market where clouds just want to sell their idle resources in a "best-effort" manner and simply choose to accept prices offered by the broker or not.

## III. System Model

### A. Participants in the Marketplace

We consider a broker-leading market with three types of participants:

- A broker that accepts jobs from the external users and delegates them to the private clouds. It also charges the external users and pays to private clouds. The broker wishes to maximize its profit.
- A set of private clouds $\mathcal{F}$ that are owned by different enterprises and managed separately. The owners are willing to lease their spare resources via the broker. Their aim is to maximize their individual profits.
- A set of external users who submit computation jobs to the broker for processing and are willing to pay for the service they receive.

An illustration of the system model is given in Fig. 1. All important notations in the paper are summarized in Table I.

### B. Computing Resources

The system operates in a time-slotted fashion. In a private cloud $i \in \mathcal{F}$, $S_i(t)$ homogeneous servers are available at time $t$, which is the total number of servers in cloud $i$ minus the number of servers running internal jobs at $t$.[1] $S_i(t)$ renders an ergodic random process with upper bound $S_i^{(max)}$. Each server

[1]We consider whole servers for leasing to external users and exclude the possibility of VMs running internal and external jobs sharing the same servers, for better performance isolation between internal and external jobs.

TABLE I
IMPORTANT NOTATION

| | |
|---|---|
| $a_{mk}$ | The amount of resource $k$ that a VM of type $m$ demands |
| $A_{ik}$ | The amount of resource $k$ that a server at cloud $i$ offers |
| $\mathcal{C}_i$ | Feasible configuration set at cloud $i$ |
| $d^j(t)$ | No. of jobs dropped at the job queue $Q^j$ at $t$ |
| $d^{j(max)}$ | Max. value of $d^j(t)$ |
| $D_j$ | Scheduling deadlines for job of type $j$ |
| $\mathcal{F}$ | Set of private clouds |
| $E_i^l(t)$ | The cost of turning a server at cloud $i$ off and restarting it later at $t$ |
| $E_i^h(t)$ | Extra cost per server per time slot when a server at cloud $i$ running external workload at $t$ |
| $G_b(t)$ | The profit of the broker at $t$ |
| $G_i(t)$ | The profit of cloud $i$ at $t$ |
| $G_s(t)$ | Social welfare achieved by the social planner at $t$ |
| $g_j$ | No. of VMs that a job of type $j$ needs simultaneously |
| $h^j(t)$ | The charge by the broker for processing a job of type $j$, at $t$ |
| $\mathcal{J}$ | Set of job types |
| $L_i^j$ | No. of jobs of type $j$ a server in cloud $i$ can process in a slot |
| $\mathcal{M}$ | Set of VM types |
| $o^j(t)$ | No. of external jobs of type $j$ arriving at the broker at $t$. |
| $o^{j(max)}$ | Upper bound of $o^j(t)$ |
| $p_i^m(t)$ | The price of using a VM of type $m$ at cloud $i$ |
| $Q^j(t)$ | Backlog of the queue buffering jobs of type $j$ at $t$ |
| $S_i(t)$ | No. of available servers in cloud $i$ at $t$ |
| $s_c^m$ | No. of VMs of type $m$ at configuration $c$ |
| $v_i^m(t)$ | No. of VMs of type $m$ that cloud $i$ is willing to lease |
| $W^j(t)$ | Backlog of virtual queue bounding scheduling delays of jobs in $Q^j$ |
| $x_i^c(t)$ | No. of servers running at configuration $c$ in cloud $i$ at $t$ |
| $\mu_i^j(t)$ | No. of jobs of type $j$ dispatched to cloud $i$ at $t$ |
| $\xi_j$ | Penalty of dropping a job of type $j$. |
| $\epsilon^j$ | Constant for controlling scheduling delays for jobs in $Q^j$ |

in cloud $i$ has $\mathcal{K}$ types of resources (such as CPU, RAM, disk, *etc.*), with the amount of $A_{ik}$ for type-$k$ resource, $\forall k \in \mathcal{K}$.

Suppose there are $\mathcal{M}$ types of virtual machines (VMs) in the system. Each type-$m$ VM requires the amount $a_{mk}$ of resource $k$, $\forall k \in \mathcal{K}$. At each time slot, a server can be configured to provision a number of VMs of different types, which we refer to as a *configuration*. A configuration $c$ is described by a vector $\vec{s_c} =< s_c^1, ..., s_c^m, ..., s_c^{|\mathcal{M}|} >$, where $s_c^m$ is the number of VMs of type $m \in \mathcal{M}$ coexisting on the server. A feasible configuration should satisfy the following constraints:

$$\sum_{m \in \mathcal{M}} s_c^m a_{mk} \leq A_{ik}, \forall k \in \mathcal{K} \quad (1)$$

We denote the set of feasible configurations of servers in cloud $i$ as $\mathcal{C}_i$.

### C. Workload

We consider delay-tolerant workloads. Let $\mathcal{J}$ denote the set of jobs submitted by users to the broker. A job of type $j \in \mathcal{J}$ is specified by a three-tuple $< m_j, g_j, D_j >$, where $m_j \in \mathcal{M}$ is the type of required VMs, $g_j$ specifies the number of VMs of type $m_j$ that the job needs simultaneously, and $D_j$ is the maximally allowed time for scheduling the job. Each job can be processed within one time slot, *i.e.*, $D_j + 1$ is the deadline for completing a type-$j$ job. Let $o^j(t)$ (with upper bound $o^{j(max)}$) be the total number of type-$j$ jobs that the external users submit at the beginning of time slot $t$, to the broker for processing, $\forall j \in \mathcal{J}$.

### D. Job Scheduling

**Job queues at the broker:** The broker buffers jobs from the external users, by placing jobs of the same type $j$ in a queue $Q^j$. The broker decides how to dispatch the jobs from the queues to the private clouds. At time slot $t$, the number of type-$j$ jobs dispatched from the broker to private cloud $i$ is $\mu_i^j(t)$, $\forall i \in \mathcal{F}$, $j \in \mathcal{J}$, *i.e.*,

$$\mu_i^j(t) \in \mathbb{Z}^+ \cup \{0\}, \forall i \in \mathcal{F}, j \in \mathcal{J} \quad (2)$$

Let $\vec{\mu_i}(t) =< \mu_i^1(t), ..., \mu_i^{|\mathcal{J}|}(t) >$ be the vector of the numbers of different types of jobs distributed to cloud $i$. When a job cannot be scheduled within the specified deadline, it is dropped from the queue and delegated to an outside public cloud for immediate processing. Let $d^j(t)$ be the number of type-$j$ jobs dropped by the broker at $t$, the maximum value of which is $d^{j(max)}$, *i.e.*,

$$d^j(t) \in \mathbb{Z}^+ \cup \{0\}, d^j(t) \leq d^{j(max)}, \forall j \in \mathcal{J} \quad (3)$$

The update equation of job queue $Q^j(t), \forall j \in \mathcal{J}$, is

$$Q^j(t+1) = \max\{Q^j(t) - \sum_{i \in \mathcal{F}} \mu_i^j(t) - d^j(t), 0\} + o^j(t) \quad (4)$$

**Handling scheduling delay constraints:** To guarantee that all non-dropped jobs are scheduled before the respective deadlines $D_j$, we apply the $\epsilon$-persistence queue technique [17], by associating a virtual queue $W_j$ with each job queue $Q^j$. The update equations of these virtual queues are as follows:

$$W^j(t+1) = \max\{W^j(t) + 1_{\{Q^j(t)>0\}}[\epsilon^j - \sum_{i \in \mathcal{F}} \mu_i^j(t)] - d^j(t) - 1_{\{Q^j(t)=0\}} \sum_{i \in \mathcal{F}} S_i(t) L_i^j / g_j, 0\}, \forall j \in \mathcal{J}, \quad (5)$$

where $\epsilon^j$'s are constants for controlling the delays $D_j$'s, for which the values are set by our online algorithms (to be discussed later), and $L_i^j$ is a constant representing the maximum number of jobs of type $j$ a server in private cloud $i$ can process in one time slot, *i.e.*, $L_i^j = \min_{k \in \mathcal{K}}\{\lfloor \frac{A_{ik}}{a_{m_j k}} \rfloor\}$. $\mathbf{1}_{\{\mathbf{X}\}}$ is an indicator function, the value of which is 1 if $X$ is true, or 0 otherwise. To avoid overwhelming the system when the overall workload is large, the maximum number of dropped jobs should be sufficiently large, *i.e.*, we assume $d^{j(max)} \geq \max\{o^{j(max)}, \epsilon^j\}$.

### E. VM Provisioning

At time slot $t$, in private cloud $i$, each server can be configured into a configuration $c \in \mathcal{C}_i$, or turned off to save power [18]. Let $x_i^c(t)$ be the number of servers configured to configuration $c$ in cloud $i$, and $v_i^m(t)$ be the total number of type-$m$ VMs that private cloud $i$ can supply. Let $\vec{v_i}(t) =< v_i^1(t), ..., v_i^{|\mathcal{M}|}(t) >$ be the vector of the total numbers of VMs of different types that private cloud $i$ can supply. They should satisfy the following constraints:

$$v_i^m(t) \leq \sum_{c \in \mathcal{C}_i} x_i^c(t) s_c^m, \forall m \in \mathcal{M} \quad (6)$$

$$\sum_{c \in \mathcal{C}_i} x_i^c(t) \leq S_i(t) \quad (7)$$

$$v_i^m(t) \in \mathbb{Z}^+ \cup \{0\}, \forall m \in \mathcal{M} \quad (8)$$

$$x_i^c(t) \in \mathbb{Z}^+ \cup \{0\}, \forall c \in \mathcal{C}_i \quad (9)$$

Since the workload dispatched from the broker to a private cloud should not exceed the total number of VMs that the private cloud is willing to supply, we further have this constraint:

$$\sum_{j:m_j=m, j\in\mathcal{J}} g_j \mu_i^j(t) \le v_i^m(t), \forall m \in \mathcal{M}, \forall i \in \mathcal{F}. \quad (10)$$

### F. Prices and Costs

**Charges to the external users:** The broker charges an external user $h^j(t)$ for running a job of type $j \in \mathcal{J}$ at $t$. We suppose that $h^j(t)$ is determined by the broker at a level which offers a discount off the rate of instances in public clouds [19], in order to make this platform attractive to users.

**Prices offered to the private clouds:** Let $p_i^m(t)$ be the price that the broker offers to private cloud $i$ for leasing a VM of type $m$ for one time slot, *i.e.*,

$$p_i^m(t) \ge 0, \forall i \in \mathcal{F}, m \in \mathcal{M} \quad (11)$$

Let $\vec{p_i}(t) =< p_i^1(t), ..., p_i^{|\mathcal{M}|}(t) >$ be the vector of prices offered to cloud $i$, for VMs of different types.

**Operational costs at the private clouds:** Operational cost is mainly related to the power consumption [1]. We assume that the cost of turning a server off and restarting it later is $E_i^l(t)$ in cloud $i$, while keeping a server running incurs a cost of $E_i^l(t)+E_i^h(t)$. The rationale behind is that even with a very low load, such as $10\%$ CPU utilization, the power consumption of a server is over $66\%$ of its peak power usage [18].

**Penalty for dropping jobs:** When the dropped jobs are delegated to outside public clouds for immediate processing, the broker needs to pay the public clouds, constituting a penalty to the broker. Let $\xi_j$ be the penalty for dropping a job of type $j$, which is larger than any charge of a single job $h^j(t)$.

## IV. Problem Formulation

Next, we first formulate decision making at the broker and the private clouds based on the Stackelberg game, and then present a cooperative, social-welfare maximization problem.

### A. The Stackelberg Game Formulation

The objective of the broker is to maximize its long-term-average profit, by deciding prices for renting VMs from the private clouds, and scheduling the jobs from its job queues to the private clouds at each time, subject to scheduling delay guarantee of each job. The objective of a private cloud is to maximize its long-term-average individual profit, by determining the amount of resource supply to serve external workload in each time slot, subject to its limit of available servers.

We consider a two-stage Stackelberg game where the broker is the leader and the private clouds are the followers. In Stage I, the broker announces the price $\vec{p_i}(t)$ (per VM of type $m$ per time slot) to each private cloud $i$, maximizing its profit. In Stage II, each private cloud $i$ decides the number of VMs $\vec{v_i}$ to lease to the broker, also maximizing their individual profits. We introduce the two stages in a backward fashion.

*1) Stage II: Each Private Cloud Optimizing Its Own Profit:* In time slot $t$, the profit $G_i(t)$ for private cloud $i \in \mathcal{F}$ is defined as its proceeds from the broker, minus its operational costs, *i.e.*, $G_i(t) = \sum_{m\in\mathcal{M}} p_i^m(t) v_i^m(t) - E_i^h(t) \sum_{c\in\mathcal{C}_i} x_i^c(t) - E_i^l(t) S_i(t)$.

Let $\overline{X}$ represent the time-averaged value of a time-varying process $X(t)$, *i.e.*, $\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\{X(t)\}$. The optimization problem for private cloud $i$ is:

$$\max \ \overline{G_i} \ \text{ s.t. } (6)(7)(8)(9), \forall t \quad (12)$$

In the above formulation, the private cloud $i$ is given the offered prices $\vec{p_i}(t), \forall t$, from the broker, and computes the numbers of VMs $\vec{v_i}(t)$ to supply to the broker at each time $t$.

*2) Stage I: The Broker Maximizing its Profit:* In time slot $t$, the profit of the broker $G_b(t)$ is defined as the proceeds from the external users minus the payment to the private clouds, and minus the job dropping penalties, *i.e.*,

$$G_b(t) = \sum_{j\in\mathcal{J}} h^j(t) o^j(t) - \sum_{i\in\mathcal{F}} \sum_{m\in\mathcal{M}} p_i^m(t) \sum_{j\in\mathcal{J}:m_j=m} \mu_i^j(t) g_j - \sum_{j\in\mathcal{J}} d^j(t) \xi_j.$$

The broker's long-term optimization problem is

$$\begin{aligned} \max \quad & \overline{G_b} && (13)\\ \text{s.t.} \quad & (2)(3)(10)(11), \forall t \\ & \overline{o^j} \le \sum_{i\in\mathcal{F}} \overline{\mu_i^j} + \overline{d^j}, \forall j \in \mathcal{J} && (14)\\ & \vec{v_i}(t) = \arg\max_{\vec{p_i}(t)} \overline{G_i}, \forall t && (15) \end{aligned}$$

(14) represents that each job queue is rate stable. (15) states that the amount of computing resources that the broker can rent from the private clouds is determined by the private clouds and affected by the rental prices offered by the broker.

### B. The Cooperative Social-Welfare Maximizing Problem

As the two parties (the broker and the private clouds) selfishly make decisions to maximize their own profits, the social welfare in the system, *i.e.*, the overall profit of both parties, may not be the largest, as compared to a cooperative scenario where a social planner globally optimally schedules job execution on the available resources for social-welfare maximization, without pricing/charges between the parties. To reveal the gap from the optimal social welfare, we further formulate a social welfare maximization problem, as follows.

The social welfare in time slot $t$ is

$$G_s(t) = \sum_{j\in\mathcal{J}} h^j(t) o^j(t) - \sum_{i\in\mathcal{F}} E_i^h(t) \sum_{c\in\mathcal{C}_i} x_i^c(t) - E_i^l(t) S_i(t) - \sum_{j\in\mathcal{J}} d^j \xi_j$$

The long-term social welfare maximization problem that the social planner pursues is:

$$\max \ \overline{G_s} \ \text{ s.t. } (2)(3)(6)(7)(8)(9)(10) \quad (16)$$

## V. Online Algorithms

When the input to the long-term optimization problems (*i.e*, job arrivals, operational costs and the number of available servers in the private clouds) varies over time with unknown statistics, neither the broker nor the private clouds can plan prices, VM provisioning or job scheduling in advance, let alone guaranteeing the scheduling deadline. We apply the Lyapunov optimization framework [13] to design online algorithms that allow the broker and the private clouds to interact on the fly. We first design dynamic algorithms for each private cloud and for the broker, respectively, and then design an online algorithm which solves the social welfare maximization problem (16).

### A. Dynamic Algorithm for Each Private Cloud

To maximize the long-term-average profit in (12), it suffices for a private cloud to maximize $G_i(t)$ subject to (6) (7) (8) (9) in each time slot $t$. We can derive an upper bound of $G(t)$ as follows:

$$\begin{aligned} G_i(t) &\leq \sum_{m\in\mathcal{M}} p_i^m(t)\sum_{c\in\mathcal{C}_i} x_i^c(t)s_c^m - E_i^h(t)\sum_{c\in\mathcal{C}_i} x_i^c(t) - E_i^l S_i(t) \\ &= \sum_{c\in\mathcal{C}_i} x_i^c(t)\Big(\sum_{m\in\mathcal{M}} p_i^m s_c^m - E_i^h(t)\Big) - E_i^l S_i(t) \\ &\leq \sum_{c\in\mathcal{C}_i} x_i^c\Big(\max_{c\in\mathcal{C}_i}\{\sum_{m\mathcal{M}} p_i^m s_c^m\} - E_i^h(t)\Big) - E_i^l S_i(t) \\ &\leq S_i(t)\Big(\max_{c\in\mathcal{C}_i}\{\sum_{m\mathcal{M}} p_i^m s_c^m\} - E_i^h(t)\Big) - E_i^l S_i(t) \end{aligned}$$

The equality can be established, *i.e.*, $G_i(t)$ achieves its largest value which is equal to the upper bound in the last row above, if there exists a $c_i^* \in \arg\max_{c\in\mathcal{C}_i}\{\sum_{m\in\mathcal{M}} p_i^m s_c^m\}$, such that $\sum_{m\in\mathcal{M}} p_i^m(t)s_{c_i^*}^m - E_i^h(t))x_i^{c_i^*}(t) \geq 0$. That is, $c_i^*$ is the most profitable configuration that private cloud $i$ can make, and the prerequisite for the private cloud to lease the server is that at least the operational cost can be covered if the private cloud provisions a server as the most profitable configuration. Hence, we derive the following optimal solution to (12):

$$x_i^c(t) = \begin{cases} S_i(t) & \text{if } c = c_i^* \text{ and } \sum_{m\in\mathcal{M}} p_i^m(t)s_{c_i^*}^m \geq E_i^h(t) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

and,

$$v_i^m(t) = \begin{cases} s_{c_i^*}^m S_i(t) & \text{if } \sum_{m\in\mathcal{M}} p_i^m(t)s_{c_i^*}^m - E_i^h(t) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

$\forall t, \forall i \in \mathcal{F}, c \in \mathcal{C}_i, m \in \mathcal{M}$. In each time slot $t$, private cloud $i$ looks for the most profitable configuration $c_i^*$, and see if it can cover the operational cost or not. If yes, the private cloud will offer all available servers as configuration $c_i^*$ to the broker. Otherwise, it decides not to lease any servers.

### B. Online Algorithm for the Broker

The broker solves its long-term profit maximization problem (13) by dynamically making decisions in each time slot. Applying the drift-plus-penalty framework in Lyapunov optimization [13], we derive the following optimization problems to be solved by the broker in each time slot (derivation is detailed in our technical report [20]), such that optimal solution to the long-term optimization problem (13) is pursued:

$$\min \quad \sum_{j\in\mathcal{J}} d^j(t)(V\xi_j - Q^j(t) - W^j(t)) \tag{19}$$
$$\text{s.t.} \quad \text{Constraint (3)}$$

and,

$$\min \quad \sum_{i\in\mathcal{F}}\sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{J}:m_j=m} \mu_i^j(t)(Vg_j p_i^m(t) - Q^j(t) - W^j(t)) \tag{20}$$
$$\text{s.t.} \quad \text{Constraints} \quad (2)(10)(11)(15).$$

where $V$ is a controlling constant that represents the trade-off between scheduling delays and profit. We will show its impact in Sec. VII.

(19) is a weight-minimizing problem. Its solution is readily available as follows:

$$d^j(t) = \begin{cases} d^{j(max)} & \text{if } V\xi_j < Q^j(t) + W^j(t) \\ 0 & \text{Otherwise} \end{cases}, \tag{21}$$

(21) means that when the sum of the backlogs of the job queue and the associated virtual queue goes beyond a threshold ($V_i\xi_j$), the jobs in queue $j$ should be maximally dropped (*i.e.*, sent to public cloud for immediate processing), in order to meet the job scheduling delay requirement. More analysis will come in Sec. VI.

In (20), the constraints are separated for different $i \in \mathcal{F}$, and thus we can equivalently decompose the problem into multiple disparate problems related to $\vec{\mu_i}(t)$ and $\vec{p_i}(t)$ of each cloud $i \in \mathcal{F}$ only. In addition, as the broker knows that each private cloud $i$ will respond with $\vec{v_i}(t)$ to given $\vec{p_i}(t)$ according to (18), constraint (15) can be replaced by (18). Therefore, (20) is equivalent to the following: for each $i \in \mathcal{F}$,

$$\min \sum_{m\in\mathcal{M}}\sum_{j\in\mathcal{J}:m_j=m} \mu_i^j(t)(Vg_j p_i^m(t) - Q^j(t) - W^j(t)) \tag{22}$$
$$s.t. \sum_{j\in\mathcal{J}:m_j=m} g_j\mu_i^j(t) \leq v_i^m(t), \forall m \in \mathcal{M}$$
$$\sum_{m\in\mathcal{M}} p_i^m(t)s_c^m = \max_{c\in\mathcal{C}_i}\{\sum_{m\in\mathcal{M}} p_i^m(t)s_c^m\} \tag{23}$$

$$v_i^m(t) = \begin{cases} s_c^m S_i(t) & \text{if } \sum_{m\in\mathcal{M}} p_i^m(t)s_c^m - E_i^h(t) \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad \forall m \in \mathcal{M}$$

The above is further equivalent to:

$$\min \quad (22)$$
$$s.t. \sum_{j\in\mathcal{J}:m_j=m} g_j\mu_i^j(t) \leq s_c^m S_i(t), \forall m \in \mathcal{M} \tag{24}$$
$$\sum_{m\in\mathcal{M}} p_i^m(t)s_c^m \geq E_i^h(t) \tag{25}$$
$$(23)$$

This is a Mixed Integer Bilevel Program, which is generally NP-hard [21]. Therefore we design a heuristic as follows, which is summarized in Alg. 1:

1. We relax the integer constraint of $\vec{s_c}$ in the second-level optimization problem (23), which then becomes a linear program. The KKT optimality conditions [22] of the relaxed liner program are as follows:

$$KKT_i \begin{cases} \sum_{m\in\mathcal{M}} p_i^m(t) + \sum_{k\in\mathcal{K}} \lambda_k \sum_{m\in\mathcal{M}} a_{mk} = 0 \\ (1) \\ \lambda_k \geq 0, \forall k \in \mathcal{K} \\ \lambda_k(\sum_{m\in\mathcal{M}} s_c^m a_{mk} - A_{ik}) = 0, \forall k \in \mathcal{K} \end{cases}$$

where $\lambda_i, \forall k \in \mathcal{K}$ are dual variables associated with constraints (1).

**Algorithm 1:** The Broker's Algorithm for each $i \in \mathcal{F}$ at $t$

**Input**: $V, g_j, Q^j(t), W^j(t), \forall j \in \mathcal{J}$, and, $E_i^h(t), S_i(t)$.
**Output**: $p_i^m(t), \forall m \in \mathcal{M}$, and, $\mu_i^j, \forall j \in \mathcal{J}$, and, $s_c^m, \forall c \in \mathcal{C}_i, \forall m \in \mathcal{M}$.
1. Solve the integral relaxation of
$$\max \quad (22) \quad \text{s.t.:} \quad \text{Constraints } (24)(25), KKT_i$$
and get $(\vec{p_i}^1(t), \vec{\mu_i}^1(t), \vec{s_c}^1)$.
2. Fix $\vec{p_i}^1(t)$ to solve (23), with the integral constraint kept, and get the integral optimal solution $\vec{s_c}^2$.
3. If $(\vec{p_i}^1(t), \vec{s_c}^2)$ does not satisfy (25), update $\vec{p_i}^1(t) = \vec{p_i}^1(t) \cdot \frac{E_i^h(t)}{\sum_{m \in \mathcal{M}} p_i^m(t) s_c^m}$.
4. Fix $\vec{p_i}^1(t)$ and $\vec{s_c}^2$ to solve
$\max$ (22) s.t. Constraint (24).
with integral constraint kept, and we can get $\vec{\mu_i}^3(t)$.
5. Return an approximated solution $(\vec{p_i}^1(t), \vec{\mu_i}^3(t), \vec{s_c}^2)$.

Now we are able to reformulate (22) as a one-level optimization problem by replacing (23) by $KKT_i$, *i.e.*,

$$\max \quad (22) \quad \text{s.t.:} \quad \text{Constraints } (24)(25), KKT_i.$$

It is still a mixed integer program. We further relax the integral constraint on $\vec{\mu_i}(t)$, and then solve the relaxed problem to derive solution $(\vec{p_i}^1(t), \vec{\mu_i}^1(t), \vec{s_c}^1)$.

2. We fix $\vec{p_i}(t)$ to $\vec{p_i}^1(t)$ to solve (23), with the integral constraint. It is a classical Multi-dimensional Knapsack Problem, which can be solved by many mature methods [23]. We can solve it and obtain the integral optimal solution $\vec{s_c}^2$.

3. If $(\vec{p_i}^1(t), \vec{s_c}^2)$ does not satisfy (25), we update $\vec{p_i}^1(t) = \vec{p_i}^1(t) \cdot \frac{E_i^h(t)}{\sum_{m \in \mathcal{M}} p_i^m(t) s_c^m}$. Doing this still honors (23).

4. We fix $\vec{p_i}(t)$ to $\vec{p_i}^1(t)$ and $\vec{s_c}$ to $\vec{s_c}^2$ to solve $\max$ (22) s.t.: Constraint (24), with integral constraint kept, and we can get $\vec{\mu_i}^3(t)$.

5. Combine $(\vec{p_i}^1(t), \vec{\mu_i}^3(t), \vec{s_c}^2)$ as the approximated solution to the original problem (22).

In summary, at the beginning of each time slot $t$, the broker accepts job submissions from the external users, runs Alg. 1 and then offers computed prices $\vec{p_i}^1(t)$ to private cloud $i \in \mathcal{F}$. Upon receiving the prices, each private cloud responds with the maximum number of VMs to supply ($\vec{v_i}(t)$) according to (18). The broker then schedules jobs according to $\vec{\mu_i}^3(t)$, and drops jobs according to (21). Finally the broker updates queue backlogs according to (4) and (5).

We will show in Sec. VI that this online algorithm combined with the optimal strategy of private clouds shown in Sec. V-A achieve a close-to-offline-optimum time-averaged aggregate profit for the broker and private clouds, while honoring the job scheduling deadlines of all jobs.

### C. Online Social-Welfare Maximizing Algorithm

Similarly, we apply the drift-plus-penalty framework of Lyapunov optimization to derive an online algorithm to solve the social welfare maximization problem (16). The social planner should solve the following optimization problem in each time slot (derivation is detailed in our technical report [20]):

**Algorithm 2:** Social-welfare Maximization Alg. at $t$

**Input**: $V, g_j, Q^j(t), W^j(t), \forall j \in \mathcal{J}$, and, $E_i^h(t), S_i(t), \forall i \in \mathcal{F}$.
**Output**: For each private cloud $i \in \mathcal{F}$: $\mu_i^j, \forall j \in \mathcal{J}$, and, $s_c^m, \forall c \in \mathcal{C}_i, \forall m \in \mathcal{M}$.
1. For each private cloud $i \in \mathcal{F}$:
1.1. Solve (28), so we pick a job queue $j_m^*$ for each VM type $m$.
1.2. Find the optimal server configuration $\vec{s_c}$, by solving (32).
1.3. If $\sum_{m \in \mathcal{M}} s_c^m (Q^{j_m^*}(t) + W^{j_m^*}(t)) > V E_i^h(t)$, we can configure each server in private cloud $i$ to that configuration, and schedule $\hat{\mu_i^j} = \lfloor \frac{S_i(t) s_{c^*}^{m_j}}{g_j} \rfloor$ jobs out of queue $Q^{j_m^*}$ to private cloud $i$. Otherwise, put all the servers in private cloud $i$ into low power state.
2. Drop jobs according to (21) and update queue backlogs according to (4) and (5).

$$\min \sum_{j \in \mathcal{J}} d_j(t)(V\xi_j - Q^j(t) - W^j(t)) \quad \text{s.t.} \quad \text{Constraint } (3) \tag{26}$$

and for each private cloud $i$,

$$\min - \sum_{j \in \mathcal{J}} \mu_i^j(t)(Q^j(t) + W^j(t)) + \sum_{c \in \mathcal{C}_i} x_i^c(t) V E_i^h(t) \tag{27}$$
$$s.t.: \sum_{j: m_j = m, j \in \mathcal{J}} g_j \mu_i^j(t) \le \sum_{c \in \mathcal{C}_i} x_i^c(t) s_c^m, \forall m \in \mathcal{M}$$
$$\text{Constraints } (2)(7)(9)$$

The solution of (26) is the same as (21).

(27) is an Integer Linear Program. We design a solution heuristic with reduced computation time and satisfactory sub-optimal results, as follows:

1. We assume $x_i^c, \forall c \in \mathcal{C}_i$, are known and solve (27) by relaxing the integral constraint of $\mu_i^j(t)$'s. We note that round-downs of the solution $\mu_i^j(t)$'s, *i.e.*, $\lfloor \mu_i^j(t) \rfloor$, are always feasible solutions of (27) with integral constraints satisfied and given $x_i^c$'s. The relaxation of (27) with given $x_i^c$'s is a weight-minimizing problem, with weight $-\frac{Q^j(t)+W^j(t)}{g_j}$. For each $m \in \mathcal{M}$, there exsits

$$j_m^* = \arg\max_{j'} \{ \frac{Q^{j'}(t) + W^{j'}(t)}{g_{j'}} | j' : m_{j'} = m \} \tag{28}$$

The solution of the relaxation of (27) is:

$$\hat{\mu_i^j} = \begin{cases} \frac{\sum_{c \in \mathcal{C}_i} x_i^c(t) s_c^{m_j}}{g_j} & \text{if } j = j_{m_j}^* \\ 0 & \text{Otherwise} \end{cases}. \tag{29}$$

This means that among all the job types that correspond to a VM type, we select a job type that is associated the minimal weight and assign all VMs of the corresponding type to serve jobs of this type.

2. By plugging (29) into (27), we get

$$\min \quad \sum_{c \in \mathcal{C}_i} x_i^c [V E_i^h(t) - \sum_{m \in \mathcal{M}} s_c^m (Q^{j_m^*}(t) + W^{j_m^*}(t))] \tag{30}$$
$$s.t. \quad \text{Constraints } (7)(9)$$

This is an integer program with special structure to be exploited. Its optimal solution is

$$\hat{x_i^c} = \begin{cases} S_i(t) & \text{if } c = c^* \text{ and} \\ & V E_i^h(t) < \sum_{m \in \mathcal{M}} s_c^m (Q^{j_m^*}(t) + W^{j_m^*}(t)) \\ 0 & \text{otherwise} \end{cases}, \tag{31}$$

where $c^* = \arg\max_{c' \in \mathcal{C}_i} \{\sum_{m \in \mathcal{M}} s_{c'}^m (Q^{j_m^*}(t) + W^{j_m^*}(t))\}$. This means that we configure all $S_i(t)$ servers to configuration $c^*$.

3. What remains to do is to find $c^* \in \mathcal{C}_i$ as follows:

$$\max \sum_{m \in \mathcal{M}} s_c^m (Q^{j_m^*}(t) + W^{j_m^*}(t)) \quad s.t. \text{ Constraint (1).} \tag{32}$$

(32) is a small-scale Multiple-dimensional Knapsack Problem [23]. Consider there are in general 3 types of resources with each VM (CPU, RAM, and disk storage). Then the number of VMs that can be packed into a server is a small number, and the problem can be solved with a classical method quickly.

4. By plugging (31) into (29) and rounding the numbers down, we have

$$\hat{\mu_i^j} = \begin{cases} \lfloor \frac{S_i(t) s_{c^*}^{m_j}}{g_j} \rfloor & \text{if } j = j_{m_j}^* \\ 0 & \text{otherwise} \end{cases}. \tag{33}$$

The social planner's algorithm is summarized in Alg. 2.

## VI. Performance Analysis

Detailed proofs of the following theorems can be found in our technical report [20].

### A. Guarantee of Job Scheduling Delays

*Theorem 1:* (Guarantee of Scheduling Delay) If we set $\epsilon^j = \frac{Q^{j(max)} + W^{j(max)}}{D_j}$ at the broker, each job of type $j \in \mathcal{J}$ on the broker is either processed in the federated private clouds or dropped for outsourcing before its maximum allowed scheduling delay $D_j$.

Theorem 1 is proved based on Lemma 1 in [20] and the $\epsilon$-persistence queue technique introduced in [17]. The condition on $\epsilon^j$ is to make the growth of the virtual queue fast enough to pose pressure on scheduling the corresponding jobs, or ensure triggering the drop mechanism before the deadline is passed. The broker only drops jobs when the resource capacity is lower than the workload demand, either at a time-average sense, or at a temporary sense due to the spike of job arrival curves and server availability curves.

### B. Optimality of Profit

*Theorem 2:* (Optimality of Profit of the Broker) Let $\Psi^*$ be the offline optimum of the time-averaged profit that the broker and the private clouds can obtain based on complete information on job arrivals, server availability, and operational costs of the private clouds for all the time. The solutions of (19) and (22) can achieve a time-averaged profit, $\Psi$, within a constant gap $\frac{B}{V}$ to $\Psi^*$, *i.e.*, $\Psi \geq \Psi^* - \frac{B}{V}$, where $V$ and $B$ are constants defined in our technical report [20].

Although the above theorem only shows the performance gap between the exact solutions of (19) and (22) and the offline optimum, we will show that our heuristic in Alg. 1 also performs well in Sec. VII.

Because our algorithm for each private cloud maximizes $G_i(t)$ and therefore maximizes $\overline{G_i}$ according to the discussion in Sec. V-A, we see that each private cloud already earns the best it can.

## VII. Performance Evaluation

### A. Simulation Configuration

Our experiments are driven by a set of synthetic traces based on data from Google cluster usage [24] and Amazon EC2 [4].

*1) Compute Resource:* There were approximately 12000 servers in the Google data center where the usage data was collected. We simulate 12 private clouds. The number of available servers in each private cloud varies over time following a Poisson process with mean 600. The resource configuration of each type of VMs is based on the configuration of on-demand VMs in Amazon EC2, as given in Table II. Because configuration of physical servers in Amazon EC2 is not publicly available, we set the resource of a server using the configuration of the most powerful on-demand compute optimized VM in EC2.

*2) Workload:* Jobs in Google cluster trace are of many different types, including delay-sensitive and delay-tolerant ones. We extract the workloads with the lowest priorities 2, 1, and 0 from the Google cluster trace, to represent delay-tolerant workloads. We follow the job arrivals from the Google traces, where each time slot corresponds to one hour (as is the case in the instance markets of *SpotCloud* [3] and Amazon EC2). We convert resource usage of jobs submitted to Google cluster to the number and type of VMs of jobs in our model. The lowest priorities of $2, 1, 0$ are converted to the maximally allowed scheduling delays of $24, 48, 72$ hours, respectively.

TABLE II
Server/VM resource configuration and charge

| | ECU | RAM(GB) | Storage (GB) | Charge |
|---|---|---|---|---|
| a server | 108 | 60 | 2 x 320 SSD | |
| medium VM | 3 | 3.75 | 1 x 4 SSD | $0.113\$/h$ |
| large VM | 6.5 | 7.5 | 1 x 32 SSD | $0.225\$/h$ |
| xlarge VM | 4 13 | 15 | 2 x 40 SSD | $0.450\$/h$ |
| 2xlarge VM | 8 | 26 | 30 2 x 80 SSD | $0.900\$/h$ |

*3) Costs and Prices:* The charge to the external users for a job equals to the charge for one VM of its required type, multiplied by the required number of VMs in that job. The charges for VMs are set to be $80\%$ of the time-varying prices of Amazon spot instances [4], because we believe that the levels of prices should be competitive enough to attract the external users to use the system. Penalties for dropping jobs equal to the charges for on-demand instances in Amazon EC2, as shown in Table II. The operational costs in different private clouds are set to be the electricity rates reported in [25].

*4) Other parameters:* By default $V$ is set to be $50000$.

### B. Individual Profit and Social Welfare

In Fig. 2 we compare the aggregate profit of the broker and the private clouds obtained by the online algorithms in Sec. V-A and Sec. V-B, and the social welfare obtained by the social planner's algorithm in Sec. V-C. It shows that the selfish individual profit maximization algorithms can achieve an aggregate profit close to the maximal social welfare. In Fig. 3 we further show the aggregate profit achieved at different values of $V$. We observe that when $V$ increases,

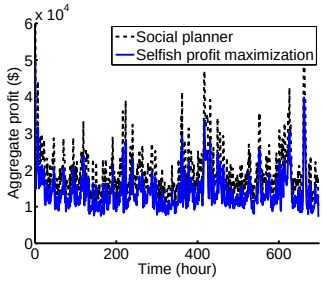


Fig. 2. Aggregate profit over time.

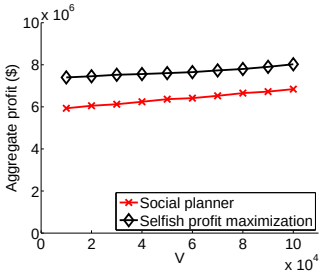


Fig. 3. Aggregate profit at different values of $V$.


Fig. 4. Comparison of average scheduling delays.


Fig. 5. Comparison of average job drop rates.

higher aggregate profits are achieved in both cases. This is because when $V$ grows, the gap between the aggregate profit achieved by our online algorithms and the offline optimum is getting smaller, which is given in Theorem 2.

### C. Scheduling Delays and Job Drops

Next we investigate the actual scheduling delays experienced by the jobs with the selfish individual profit maximization algorithms and the social planner's algorithm. Fig. 4 shows that the average scheduling delays with our online algorithms are smaller than the scheduling deadlines. In Fig. 5 we compare the job drop rates. When the overall compute capacity in the system is fixed and the scheduling deadline of jobs is larger, the system has more flexility to schedule the jobs in such a way that the deadlines of more jobs are met, so the average drop rates tend to be smaller. The average scheduling delays (and the job drop rates) experienced with the selfish profit maximization algorithms are slightly longer (and larger) than those with the social planner's algorithm, due to the selfishness of the players.

## VIII. Conclusion and Future Work

This paper presents the design of efficient mechanisms for a computing resource market led by a broker, where private clouds lease their spare computing resources to external users. We model the interaction between the broker and the private clouds as a two-stage Stackelberg game, and tailor the Lyapunov optimization framework to design online algorithms for the broker and the private clouds respectively, under time-varying job arrivals, operational costs, server availability and charges. We also develop an online algorithm for social welfare maximization as a benchmark to compare the individual profit maximization algorithms. Our proposed algorithms are evaluated based on real-life traces, and performance achieved with the individual profit maximization algorithms is shown to be close to that of the social welfare maximization algorithm. In our future work, we plan to extend our model to the case of compute capacity trading with multiple brokers.

## References


[1] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 68–73, 2009.

[2] H. Wang, F. Wang, J. Liu, and J. Groen, "Measurement and Utilization of Customer-Provided Resources for Cloud Computing," in *Proc. of IEEE INFOCOM*, Mar. 2012.

[3] *SpotCloud*, http://spotcloud.com/.

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

[5] H. Li, C. Wu, Z. Li, and F. C. Lau, "Profit-Maximizing Virtual Machine Trading in a Federation of Selfish Clouds," in *Proc. of IEEE INFOCOM (Mini-Conference)*, Apr. 2013.

[6] J. Luo, L. Rao, and X. Liu, "Temporal Load Balancing with Service Delay Guarantees for Data Center Energy Cost Optimization," *IEEE TPDS*, accepted on Feb. 15, 2013.

[7] D. fudenberg and J. Tirole, *Game Theory*. MIT Press, 1991.

[8] A. N. Toosi, R. N. Calheiros, R. K. Thulasiram, and R. Buyya, "Resource Provisioning Policies to Increase IaaS Provider's Profit in a Federated Cloud Environment," in *Proc. of IEEE HPCC*, Sep. 2011.

[9] B. Rochwerger et al., "The RESERVOIR Model and Architecture for Open Federated Cloud Computing," *IBM Journal of Research and Development*, vol. 53, no. 4, 2009.

[10] J. Altmann, C. Courcoubetis, G. D. Stamoulis, M. Dramitinos, T. Rayna, M. Risch, and C. Bannink, "GridEcon: A Market Place for Computing Resources," in *Proc. of International Workshop on GECON*, Aug. 2008.

[11] V. D. Valerio, V. Cardellini, and F. L. Presti, "Optimal Pricing and Service Provisioning Strategies in Cloud Systems: A Stackelberg Game Approach," in *Proc. of IEEE CLOUD*, Jun. 2013.

[12] M. Hadji, W. Louati, and D. Zeghlache, "Constrained Pricing for Cloud Resource Allocation," in *Proc. of IEEE International Symposium on Network Computing and Applications*, Aug. 2011.

[13] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.

[14] H. Li, W. Huang, C. W. abd Z. Li, and F. C. Lau, "Utility-Maximizing Data Dissemination in Socially Selfish Cognitive Radio Networks," in *Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2011)*, Oct 2011.

[15] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, Y. Qu, and B. Li, "eTime: Energy-Efficient Transmission between Cloud and Mobile Devices," in *Proc. of IEEE INFOCOM(Mini)*, Apr. 2013.

[16] W. Deng, F. Liu, H. Jin, and C. Wu, "SmartDPSS: Cost-Minimizing Multi-source Power Supply for Datacenters with Arbitrary Demand," in *Proc. of IEEE ICDCS*, Jul. 2013.

[17] M. J. Neely, "Opportunistic Scheduling with Worst Case Delay Guarantees in Single and Multi-Hop Networks," in *Proc. of IEEE INFOCOM*, Apr. 2011.

[18] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services," in *Proc. of NSDI*, Apr. 2008.

[19] B. Javadi, R. K. Thulasiram, and R. Buyya, "Statistical Modeling of Spot Instance Prices in Public Cloud Environments," in *Proc. of IEEE International Conference on Utility and Cloud Computing*, Dec. 2011.

[20] X. Qiu, C. Wu, H. Li, Z. Li, and F. C. M. Lau, "Federated Private Clouds via Brokers Marketplace: A Stackelberg-Game Perspective," `http://www.cs.hku.hk/~xjqiu/federation-broker.pdf`, The University of Hong Kong, Tech. Rep., Jan. 2014.

[21] B. Colson, P. Marcotte, and G. Savard, "An Overview of Bilevel Optimization," *Ann. of Oper. Res.*, vol. 153, pp. 235–256, 2007.

[22] S. Boyd, *Convex Optimization*. Cambridge University Press, 2004.

[23] M. J. Varnamkhasti, "Overview of the Algorithms for Solving the Multidimensional Knapsack Problems," *Advanced Studies in Biology*, vol. 4, no. 1, pp. 37–47, 2012.

[24] *Google Cluster Data*, http://code.google.com/p/googleclusterdata/.

[25] *United States Energy Information Administration, Dept. of Energy*, http://www.eia.doe.gov.