Profit-Maximizing Virtual Machine Trading in a Federation of Selfish Clouds

Hongxing Li*, Chuan Wu*, Zongpeng Li† and Francis C.M. Lau*
*Department of Computer Science, The University of Hong Kong, Hong Kong, Email: {hxli, cwu, fcmlau}@cs.hku.hk
†Department of Computer Science, University of Calgary, Canada, Email: zongpeng@ucalgary.ca

***Abstract*—The emerging federated cloud paradigm advocates sharing of resources among cloud providers, to exploit temporal availability of resources and diversity of operational costs for job serving. While extensive studies exist on enabling interoperability across different cloud platforms, a fundamental question on cloud economics remains unanswered: When and how should a cloud trade VMs with others, such that its net profit is maximized over the long run? In order to answer this question by the federation, a number of important, correlated decisions, including job scheduling, server provisioning and resource pricing, need to be dynamically made, with long-term profit optimality being a goal. In this work, we design efficient algorithms for inter-cloud resource trading and scheduling in a federation of geo-distributed clouds. For VM trading among clouds, we apply a double auction-based mechanism that is strategyproof, individual rational, and ex-post budget balanced. Coupling with the auction mechanism is an efficient, dynamic resource trading and scheduling algorithm, which carefully decides the true valuations of VMs in the auction, optimally schedules stochastic job arrivals with different SLAs onto the VMs, and judiciously turns on and off servers based on the current electricity prices. Through rigorous analysis, we show that each individual cloud, by carrying out our dynamic algorithm, can achieve a time-averaged profit arbitrarily close to the offline optimum.**

I. Introduction

The emerging federated cloud paradigm advocates sharing of disparate cloud services (in separate data centers) from different cloud providers, and interconnecting them based on common standards and policies to provide a universal environment for cloud computing. Such a cloud federation exploits temporal and spatial availability of resources (*e.g.*, virtual machines) and diversity of operational costs (*e.g.*, electricity prices): when a cloud experiences a burst of incoming jobs, it may resort to VMs of other clouds having idle resources; when the electricity price for running servers and VMs is high at one data center, the cloud can schedule jobs onto other data centers with lower electricity charges. In this way, the aggregate job processing capacity of the cloud federation can potentially be higher than the summation of the capacities of separate clouds operating alone, leading possibly to a larger overall profit.

To realize such a federated cloud paradigm, fundamental problems on cloud economics need to be resolved. Naturally, a cloud in the real world is selfish, and would try every mean to maximize its own profit—*i.e.*, its income from handling jobs and leasing VMs to other clouds, minus its operational costs and expenses in VM rental from other clouds. Only if its profit can be maximized and in any case not lower than when operating alone, can a cloud be incentivized to join a federation. This incentive is materialized if an efficient mechanism to carry out resource trading and scheduling among federated clouds, thus achieving profit maximization for individual clouds, can be put in place. To this end, several correlated, practical decisions need to be made: (1) VM pricing: what mechanism should be advocated for VM sale and purchase among the clouds, and at what prices? (2) Job scheduling: given time-varying job arrivals at each cloud, having different resource and SLA requirements, should a cloud serve the jobs right away or later, and use its own resources or others', in order to take advantage of lower electricity prices? (3) Server provisioning: is it more beneficial for a cloud to keep many of its servers running to serve jobs of its own and those from others, or to switch some of them off to save electricity costs? These decisions should be efficiently and optimally made in an online fashion, which in turn provide a guarantee for long-term optimality of individual cloud's profits.

In this paper, we design efficient algorithms for inter-cloud resource trading and scheduling, in a federation consisting of disparate cloud data centers. A double auction-based mechanism is applied for the sell and purchase of available VMs across cloud boundaries, which is strategy-proof, individual rational, and ex-post budget balanced. Closely combined with the auction mechanism is an efficient, dynamic VM trading and scheduling algorithm, which carefully decides the true valuations of VMs to participate in the auction, optimally schedules randomly-arriving jobs with different resource requirements (*e.g.*, number of VMs) and SLAs (*e.g.*, maximum job scheduling delay) onto different data centers, and judiciously turns on and off servers in the clouds based on the current electricity prices. The contributions of this work are summarized as follows.

First, we address selfishness of individual clouds in a cloud federation, and design efficient mechanisms to maximize the net profit of each participating cloud. This profit is not only guaranteed to be larger than that when the cloud operates alone, but also maximized over the long run, in the presence of time-varying job arrivals and electricity prices.

Second, we novelly combine a truthful double auction mechanism with stochastic Lyapunov optimization techniques, and design an online VM trading and scheduling algorithm for a cloud to optimally price the VMs and judiciously schedule the VM and server usages. Each cloud values different VMs based on the back pressure in its job queues, and bids for them in the auction for effective VM acquisition.

Third, we demonstrate that by applying the dynamic algorithm with double auction, each cloud can achieve a time-averaged profit arbitrarily close to its offline optimum (obtained if the cloud has complete knowledge of the incoming

jobs and electricity prices over the entire time span).

In the rest of the paper, we discuss related literature in Sec. II, present the system model in Sec. III, introduce the detailed resource trading and scheduling mechanisms in Sec. IV, and conclude the paper in Sec. V.

II. Related work

A. Optimal Scheduling in Cloud Computing

Existing literature ([1]–[3] and references therein) on resource scheduling in cloud systems focuses mainly on a single cloud that operates alone. A common theme is to minimize the operational costs (mainly consisting of electricity bills) in one or multiple data centers of the cloud, while providing certain performance guarantee for job scheduling, *e.g.*, in terms of average job completion times [1]–[3]. Different from these studies, this work investigates bounded scheduling delay for each job even in the worst cases, and profit maximization for individual selfish clouds in a cloud federation.

B. Resource Trading Mechanisms

There is a large body of literature devoting to resource trading in computing grids [4] and wireless spectrum leasing [5] [6]. Various mechanisms have been studied, *e.g.*, bargaining [4], fixed or dynamic pricing based on a contract or the supply-demand ratio [7], and auctions [5], [6].

A typical bargaining mechanism [4] tends to have unacceptable complexity when negotiating between each pair of traders. Fixed pricing, *e.g.*, Amazon EC2 on-demand instances, has been shown to be inefficient in profit maximization given there are system dynamics [8]. Dynamic pricing, such as Amazon EC2 spot instances, could be inefficient too, as the participants may quote the resources untruthfully [9].

Auction stands out as a promising mechanism, for which there are many solutions ([5], [6] and references therein) with truthful design and polynomial complexity. Although some recent works [8], [9] aim to design an auction mechanism with individual rationality (non-negative profit gain) for trading in federated clouds, they do not explicitly address individual profit maximization over the long run. Moreover, there is very little discussion in the literature on auctions about methods to quantitatively calculate the true valuation in each bid, which is usually assumed as known. Our design addresses these issues.

III. System Model and Auction Framework

A. Federation of Clouds

We consider a federation of F clouds, each of which assigned to a different geometric location and operates autonomously to gain profit by serving its customers' job requests, managing server provisioning and trading resources with other clouds.

Service demands: Each individual cloud $i \in [1, F]$ has a front-end proxy server, which accepts job requests from its customers. There are S types of jobs being serviced at each cloud, each specified by a three-tuple $< m_s, g_s, d_s >$. Here, $m_s \in [1, M]$ refers to the type of the required VM instances, where M is the maximum number of VM types, and each type corresponds to a different set of configurations of CPU, storage and memory; g_s is the number of type-m_s VMs that the job needs simultaneously (see Amazon EC2 API [7]); and d_s stands for the SLA (Service Level Agreement) of job type $s \in [1, S]$, evaluated by the maximal response delay for scheduling a job, *i.e.*, the time-span from when the job arrives to when it starts to run on the scheduled VMs. In a real cloud, it is common to buy servers of the same configuration and provision the same type of VMs on one machine [10]. Therefore, we suppose each cloud i has N_i^m homogeneous servers to provision VMs of type $m \in [1, M]$, each of which can provide a maximum of C_i^m VMs of this type; the total number of servers in cloud i is $\sum_{m=1}^{M} N_i^m$.

The system runs in a time-slotted fashion. At the beginning of each time slot t, $r_i^s(t) \in [0, R_i^s]$ jobs arrive at cloud i, for each job type s. R_i^s is an upper-bound on the number of type-s jobs submitted to cloud i in a time slot. The arrival of jobs is an ergodic process at each cloud. We assume the arrival rate is given, and how a customer decides which cloud to use is not a concern in this study. Let $p_i^s(t) \in [0, p_i^{s(max)}]$ be the given service charge to the customer by cloud i, for accepting a job of type s in time slot t, which remains fixed within a time slot, but may vary across time slots. Here, $p_i^{s(max)}$ is the maximum possible price for $p_i^s(t)$.

Job scheduling: Each incoming job to cloud i enters a FIFO queue of its type—a cloud i maintains a queue to buffer unscheduled jobs of each type s, with $Q_i^s(t)$ as its length in t. When the required VMs of a job are allocated, the job departs from its queue and starts to run on the VMs. A cloud may schedule its jobs on either its own VMs or VMs leased from other clouds, whichever yields the best economic benefits. Let $\mu_{ij}^s(t)$ be the number of type-s jobs of cloud i that are scheduled for processing in cloud j at the beginning of slot t.

When a job's maximum response time (the SLA) cannot be met, probably because of system overload, the job is dropped. A penalty is raised in this case, to compensate for the customer's loss. Let

$$D_i^s(t) \in [0, D_i^{s(max)}] \tag{1}$$

be the number of type-s jobs dropped by cloud i in t, where $D_i^{s(max)}$ is the maximum value of $D_i^s(t)$. Let $\xi_i^s \geq p_i^{s(max)}$ be the penalty to drop one such job, which is at least equal to the maximum price charged to the customer if the job were accepted.

Hence, the number of unscheduled jobs buffered at each cloud can be updated with the following the queueing law:

$$\begin{aligned} Q_i^s(t+1) = &\max\{Q_i^s(t) - \sum_{j=1}^{F} \mu_{ij}^s(t) - D_i^s(t), 0\} \\ &+ r_i^s(t), \quad \forall s \in [1, S], \forall i \in [1, F]. \end{aligned} \tag{2}$$

Job scheduling should satisfy the following SLA constraint:

Each type-s job in cloud i is either scheduled or dropped (subject to a penalty) before its maximum response delay d_s, $\forall s \in [1, S]$. (3)

To satisfy this SLA constraint, we seek to bound the lengths of job queues and the following virtual queues Z_i^s, each associated with a job queue Q_i^s. The virtual queues are created based on the $\epsilon-$persistence queue technique [11].

$$Z_i^s(t+1) = \max\{Z_i^s(t) + \mathbf{1}_{\{Q_i^s(t)>0\}} \cdot [\epsilon_s - \sum_{j=1}^{F} \mu_{ij}^s(t)] - D_i^s(t)$$

$$- \mathbf{1}_{\{Q_i^s(t)=0\}} \cdot \sum_{j=1}^{F} \frac{C_j^{m_s} \cdot N_j^{m_s}}{g_s}, 0\}, \forall i \in [1,F], s \in [1,S]. \quad (4)$$

Here, $\epsilon_s > 0$ is a constant. $\mathbf{1}_{\{Q_i^s(t)>0\}}$ and $\mathbf{1}_{\{Q_i^s(t)=0\}}$ are indicator functions such that

$$\mathbf{1}_{\{Q_i^s(t)>0\}} = \begin{cases} 1 & \text{if } Q_i^s(t) > 0 \\ 0 & \text{Otherwise} \end{cases}; \ \mathbf{1}_{\{Q_i^s(t)=0\}} = \begin{cases} 1 & \text{if } Q_i^s(t) = 0 \\ 0 & \text{Otherwise} \end{cases}.$$

Length of a virtual queue reflects the cumulated response delay of jobs from the respective job queue.

Server provisioning: We consider the electricity cost, for running and cooling the servers [12], as the main component of the operational cost in a cloud. Other costs, *e.g.*, space rental and labour, remain relatively fixed for a long time, and therefore are of less interest. Given that electricity prices vary at different locations and from time to time [1], we model the operational cost $\beta_i(t)$ in each cloud i as a general ergodic process over time, varying across time slots between $\beta_i^{(min)}$ and $\beta_i^{(max)}$.

Each cloud strategically decides the number of active servers at each time, to optimize its profit. Let $n_i^m(t)$ be the number of active servers provisioning type-m VMs at cloud i in t. The available server capacities constrain any feasible job scheduling strategy at time t as follows:

$$\sum_{j\in[1,F]} \sum_{s:m_s=m, s\in[1,S]} g_s \mu_{ji}^s(t) \le C_i^m \cdot n_i^m(t),$$
$$\forall m \in [1,M], \forall i \in [1,F], \quad (5)$$

$$n_i^m(t) \le N_i^m, \ \forall m \in [1,M], \forall i \in [1,F]. \quad (6)$$

(5) states that the overall demand for type-m VMs in cloud i from itself and other clouds should be no larger than the maximum number of available type-m VMs on the active servers in cloud i. Here $g_s\mu_{ji}^s(t)$ is the total number of VMs needed by type-s jobs scheduled from cloud j to cloud i in t. Considering practical job execution efficiency, we only consider the scheduling of a job to VMs in a single cloud, but not to VMs across different clouds. (6) ensures that the number of active servers is limited by the total number of on-premise servers of the corresponding VM configuration at each cloud.

B. Inter-cloud VM Trading with Double Auction

In an inter-cloud resource market, VMs are the items for trading. For each type of VMs, multiple clouds may have them on sale while multiple other clouds can request them. A double auction is a natural fit to implement efficient trading in this case, allowing both selling and buying clouds to actively participate in pricing to strive for their own benefits. In our dynamic system, a *multi-unit double auction* is carried out among the clouds at the beginning of each time slot, to decide the VM trades within that time slot.

Buyers & Sellers: A cloud can be both a buyer and a seller. A buy-bid $< b_i^m(t), \gamma_i^m(t) >$ records the unit price and maximum quantity for which cloud i is willing to buy VMs of type m, in t. Similarly, a sell-bid $< s_i^m(t), \eta_i^m(t) >$ records the unit price and maximum quantity for which cloud i is willing to sell VMs of type m in t.

Let $\tilde{b}_i^m(t)$ and $\tilde{s}_i^m(t)$ be cloud i's true valuations of buying and selling a type-m VM respectively (the max/min price it is willing to pay/accept). Similarly, let $\tilde{\gamma}_i^m(t)$ and $\tilde{\eta}_i^m(t)$ be cloud i's true valuations of the quantities to buy and sell VMs of type m respectively (the maximum number of VMs it is willing to purchase/sell). A cloud i may strategically manipulate the bid prices and volumes, in the hope of maximizing its profit.

Auctioneer: We assume that there is a broker in the cloud federation, assuming the role of the auctioneer. After collecting all the buy and sell bids, the auctioneer executes a double auction to decide the set of successful buy and sell bids, their clearing prices and the numbers of VMs to trade in each type. Let $\hat{b}_i^m(t)$ be the actual charge price for cloud i to buy one type-m VM, and $\hat{\gamma}_i^m(t)$ be the actual number of VMs purchased. Similarly, let $\hat{s}_i^m(t)$ be the actual income cloud i receives for selling one type-m VM, and $\hat{\eta}_i^m(t)$ be the actual number of VMs sold.

Let $\alpha_{ij}^m(t)$ be the number of type-m VMs that cloud i purchases from cloud j in t, as decided by the auctioneer:

$$\hat{\gamma}_i^m(t) = \sum_{j\in[1,F], j\neq i} \alpha_{ij}^m(t), \ \forall m \in [1,M], i \in [1,F], \quad (7)$$

$$\hat{\eta}_i^m(t) = \sum_{j\in[1,F], j\neq i} \alpha_{ji}^m(t), \ \forall m \in [1,M], i \in [1,F]. \quad (8)$$

Since VMs are purchased for serving jobs, the job scheduling decisions $\mu_{ij}^s(t)$ at each cloud i, $\forall j \in [1,F], s \in [1,S]$, are related to the number of VMs it purchases:

$$\sum_{s:s\in[1,S], m_s=m} g_s \cdot \mu_{ij}^s(t) = \alpha_{ij}^m(t),$$
$$\forall m \in [1,M], \forall i,j \in [1,F], i \neq j. \quad (9)$$

Three economic properties are desirable for the auctioneer's mechanism. (i) *Truthfulness*: Bidding true valuations is a dominant strategy, and consequently, both bidder strategies and auction design are simplified. (ii) *Individual Rationality*: Each cloud obtains a non-negative profit by participating in the auction. (iii) *Ex-post Budget Balance*: The auctioneer has a non-negative surplus, *i.e.*, the total payment from all winning buy-bids is no less than the total charge for all winning sell-bids in each time slot. Detailed design of an auction with these properties is given in our technical report [13].

C. Individual Selfishness

Each cloud in the federation aims to maximize its time-averaged profit (revenue minus cost) over the long run of the system, while striving to fulfil the resource and SLA requirements of each job.

Revenue: A cloud has two sources of revenue: i) job service charges paid by its customers, and ii) the proceeds from VM sales. The time-averaged revenue of cloud i by undertaking different types of jobs from its customers is

$$\Phi_1^i = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{s\in[1,S]} \mathbb{E}\{p_i^s(t) \cdot r_i^s(t)\}, \ \forall i \in [1,F]. \quad (10)$$

We assume the front-end charges, $p_i^s(t)$, from a cloud to its customers, are given. Hence, this part of the revenue is fixed in each time slot. The time-averaged income of cloud i from selling VMs to other clouds is:

$$\Phi_2^i = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{m\in[1,M]} \mathbb{E}\{\hat{s}_i^m(t) \cdot \hat{\eta}_i^m(t)\},\ \forall i \in [1, F]. \quad (11)$$

Cloud i can control this income by adjusting its sell-bids, *i.e.*, $s_i^m(t)$ and $\eta_i^m(t)$, $\forall m \in [1, M]$, at each time.

Cost: The cost of cloud i consists of three parts: i) operational costs incurred for running its active servers, ii) the penalties for dropping jobs, and iii) the expenditure on buying VMs from other clouds. The time-averaged cost for operating servers is decided by the number of active servers in each time, *i.e.*,

$$\Psi_1^i = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{\beta_i(t) \cdot \sum_{m=1}^{M} n_i^m(t)\},\ \forall i \in [1, F]. \quad (12)$$

The time-averaged penalty is determined by the number of dropped jobs over time, *i.e.*, $D_i^s(t),\ \forall s \in [1, S], t \in [0, T-1]$:

$$\Psi_2^i = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{s\in[1,S]} \mathbb{E}\{\xi_i^s \cdot D_i^s(t)\},\ \forall i \in [1, F]. \quad (13)$$

The time-averaged expenditure for VM purchase is decided by the actual VM trading prices and numbers, as decided by the buy-bids $(b_i^m(t), \gamma_i^m(t))$ from cloud i:

$$\Psi_3^i = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{\sum_{m=1}^{M} \hat{b}_i^m(t) \cdot \hat{\gamma}_i^m(t)\}. \quad (14)$$

Profit Maximization: The profit maximization problem at cloud $i \in [1, F]$ can be formulated as follows:

$$\begin{aligned} \max \quad & \Phi_1^i + \Phi_2^i - \Psi_1^i - \Psi_2^i - \Psi_3^i \quad (15)\\ s.t. \quad & \text{Constraints (1)-(9).} \end{aligned}$$

IV. Dynamic algorithms

We next present a dynamic algorithm for each cloud to trade VMs and schedule jobs/servers, which is in fact applicable under any truthful, individual-rational and ex-post budget balanced double auction mechanism. Fig. 1 illustrates the relation among these algorithm modules.

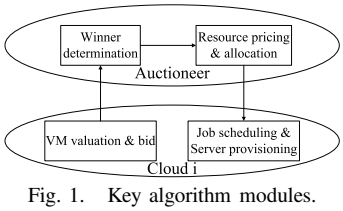

Fig. 1. Key algorithm modules.

The goal of the dynamic algorithm at each cloud i is to maximize its time-averaged profit, *i.e.*, to solve optimization (15), by dynamically making decisions in each time slot. We apply the drift-plus-penalty framework in Lyapunov optimization theory [14], and derive (from (15)) the following one-shot optimization problem to be solved by cloud i in each time slot t (detailed derivation in our technical report [13]):

$$\begin{aligned} \max \quad & \varphi_1^i(t) + \varphi_2^i(t) + \varphi_3^i(t) \quad (16)\\ s.t. \quad & \text{Constraints (1), (5)-(9).} \end{aligned}$$

where

$$\varphi_1^i(t) = V \sum_{m\in[1,M]} [\hat{s}_i^m(t)\hat{\eta}_i^m(t) - \hat{b}_i^m(t)\hat{\gamma}_i^m(t) - \beta_i(t)n_i^m(t)],$$

$$\varphi_2^i(t) = \sum_{s=\in[1,S]} \sum_{j\in[1,F]} \mu_{ij}^s(t)[Q_i^s(t) + Z_i^s(t)],$$

$$\varphi_3^i(t) = \sum_{s\in[1,S]} D_i^s(t)[Q_i^s(t) + Z_i^s(t) - V \cdot \xi_i^s],$$

and $V > 0$ is a user-defined parameter for gauging the optimality of the time-averaged profit.

In solving the one-shot optimization, cloud i observes the states of job and virtual queues $(Q_i(s)(t), Z_i^s(t))$, job arrival rates, the current cost for server operation $(\beta_i(t))$, and then decides the optimal values of variables for optimal decisions on i) buy/sell bids for different types of VMs, ii) scheduling of active servers and jobs to these servers, and iii) jobs to drop.

1. VM Valuation and Bid: Optimization (16) is related to the actual charges $\hat{b}_i^m(t)$ and $\hat{s}_i^m(t)$ $(\forall m \in [1, M])$ and traded VM numbers $\hat{\gamma}_i^m(t)$ and $\hat{\eta}_i^m(t)$ $(\forall m \in [1, M])$, from the double auction. These values are determined by the auctioneer according to buy-bids $(b_i^m(t), \gamma_i^m(t))$ and sell-bids $(s_i^m(t), \eta_i^m(t))$ submitted by all clouds, and its double auction mechanism.

When a truthful double auction is employed, sellers and buyers bid their *true values* of the prices and quantities, in order to maximize their individual utilities. (16) is the utility maximization problem for each cloud. If we can find true values of each cloud i, $\tilde{b}_i^m(t), \tilde{\gamma}_i^m(t)$, $\tilde{s}_i^m(t)$ and $\tilde{\eta}_i^m(t)$, and let the cloud bid using these values, the achieved profit in (16) is guaranteed to be the largest, as compared to bidding other values.

The true values of the buy/sell prices for cloud i are derived as follows (detailed derivation is presented in the technical report [13]):

$$\tilde{b}_i^m(t) = \frac{Q_i^{s_m^*}(t) + Z_i^{s_m^*}(t)}{V \cdot g_{s_m^*}}, \quad (17)$$

and

$$\tilde{s}_i^m(t) = \begin{cases} \frac{Q_i^{s_m^*}(t) + Z_i^{s_m^*}(t)}{V \cdot g_{s_m^*}} & \text{if } \frac{Q_i^{s_m^*}(t) + Z_i^{s_m^*}(t)}{V \cdot g_{s_m^*}} > \beta_i(t)/C_i^m \\ \beta_i(t)/C_i^m & \text{Otherwise} \end{cases}, \quad (18)$$

respectively, where

$$s_m^* = arg \max_{s'\in[1,S], m_{s'}=m} \{W_i^{s'}(t)\}, \quad (19)$$

$$\text{and} \qquad W_i^{s'}(t) = \frac{Q_i^{s'}(t) + Z_i^{s'}(t)}{g_{s'}}. \quad (20)$$

$W_i^{s'}(t)$ denotes the weight for scheduling one type-s' job (to run on type-$m_{s'}$ VM(s)) by cloud i in t, and s_m^* specifies the job type with the largest weight (ties broken arbitrarily), among all types of jobs requiring type-m VMs.

The true values of the number of type-m VMs to buy and to sell at cloud i are

$$\tilde{\gamma}_i^m(t) = \sum_{j\in[1,F]} C_i^m \cdot N_i^m, \quad (21)$$

$$\text{and} \qquad \tilde{\eta}_i^m(t) = C_i^m \cdot N_i^m. \quad (22)$$

They state that the maximum number of type-m VMs cloud i is willing to buy (sell) at the price in (17) (in (18)), is the number of all potential type-m VMs in the federation (the number i can provision).

Algorithm 1 Dynamic Profit Maximization Algorithm at cloud i in Time Slot t

Input: $r_i^s(t)$, $Q_i^s(t)$, $Z_i^s(t)$, g_s, m_s, ξ_i^s, C_i^m, N_i^m and $\beta_i(t)$, $\forall s \in [1,S]$.

Output: $b_i^m(t)$, $s_i^m(t)$, $\gamma_i^m(t)$, $\eta_i^m(t)$, $D_i^s(t)$, $\mu_{ij}^s(t)$ and $n_i^m(t)$, $\forall m \in [1,M], s \in [1,S], j \in [1,F]$.

1: **VM valuation and bid**: Decide $b_i^m(t)$, $s_i^m(t)$, $\gamma_i^m(t)$ and $\eta_i^m(t)$ with Eqn. (17)-(22);
2: **Server provisioning, job scheduling and dropping**: Decide $\mu_{ij}^s(t)$, $D_i^s(t)$ and $n_i^m(t)$ with Eqn. (23), (25) and (26);
3: Update $Q_i^s(t)$ and $Z_i^s(t)$ with Eqn. (2) and (4).

To conclude, in each time slot t, cloud i submits its bids as $b_i^m(t) = \tilde{b}_i^m(t)$, $s_i^m(t) = \tilde{s}_i^m(t)$, $\gamma_i^m(t) = \tilde{\gamma}_i^m(t)$ and $\eta_i^m(t) = \tilde{\eta}_i^m(t)$, for each type of VMs $m \in [1,M]$.

2. Server Provisioning, Job scheduling and Dropping: After receiving results of the double auction (actual charges and VM allocation $\alpha_{ji}^{m_s}(t), \forall s \in [1,S], \forall j \in [1,F]$), cloud i schedules its jobs on its local servers and (potentially) purchased VMs from other clouds, decides which jobs to drop and the number of active servers to provision, by solving the one-shot optimization in (16). Detailed steps can be found in our technical report [13].

The derived number of type-s jobs scheduled to run on the local servers is

$$\mu_{ii}^s(t) = \begin{cases} \frac{C_i^{m_s} \cdot N_i^{m_s} - \sum_{j \neq i} \alpha_{ji}^{m_s}(t)}{g_s} & \text{if } \frac{Q_i^s(t)+Z_i^s(t)}{V \cdot g_s} > \beta_i(t)/C_i^{m_s} \\ & \text{and } \ s = s_{m_s}^* \\ 0 & \text{Otherwise} \end{cases}, \tag{23}$$

and the number of type-s jobs to run at cloud $j(\neq i)$ is

$$\mu_{ij}^s(t) = \begin{cases} \alpha_{ij}^{m_s}(t)/g_s & \text{if } s = s_{m_s}^* \\ 0 & \text{Otherwise} \end{cases}. \tag{24}$$

The number of type-s jobs dropped by cloud i in t is

$$D_i^s(t) = \begin{cases} D_i^{s(max)} & \text{if } Q_i^s(t) + Z_i^s(t) > V \cdot \xi_i^s \\ 0 & \text{Otherwise} \end{cases}. \tag{25}$$

The number of activated servers at cloud i to provision type-m VM is calculated as

$$n_i^m(t) = \Big(\sum_{s \in [1,S], m_s = m} \mu_{ii}^s(t) \cdot g_s + \sum_{j \neq i} \alpha_{ji}^m(t) \Big) / C_i^m. \tag{26}$$

These many servers can provide enough type-m VMs for serving local jobs and selling to other clouds.

Alg. 1 summarizes the dynamic algorithm at each cloud.

We next analyze the performance guarantee provided by our dynamic algorithm. Due to space limit, all detailed proofs can be found in [13].

Lemma 1: Let $Q_i^{s(max)} = V\xi_i^s + R_i^s$ and $Z_i^{s(max)} = V\xi_i^s + \epsilon_s$. If $D_i^{s(max)} \geq \max\{R_i^s, \epsilon_s\}$, each job queue $Q_i^s(t)$ and each virtual queue $Z_i^s(t)$ are upper-bounded by $Q_i^{s(max)}$ and $Z_i^{s(max)}$, respectively, in $t \in [0, T-1]$, $\forall i \in [1,F], s \in [1,S]$.

Theorem 1 (SLA Guarantee): Each job of type $s \in [1,S]$ is either scheduled or dropped with Alg. 1 before its maximum response delay d_s, if we set $\epsilon_s = \frac{Q_i^{s(max)} + Z_i^{s(max)}}{d_s}$.

Theorem 2 (Individual Profit Optimality): Let Ω_i^* be the offline-optimal time-averaged profit of cloud $i \in [1,F]$, obtained with a truthful, individual-rational, ex-post budget-balanced double auction, with complete information about its own job arrivals and prices in the entire time span $[0, T-1]$. The dynamic Algorithm 1 can achieve a time-averaged profit Ω_i for cloud i within a constant gap B_i/V to Ω_i^*, *i.e.*,

$$\Omega_i \geq \Omega_i^* - B_i/V,$$

where $V > 0$ and $B_i = \frac{1}{2} \sum_{s \in [1,S]} [[\sum_{j=1}^F C_j^{m_s} N_j^{m_s}/g_s + D_i^{s(max)}]^2 + [R_i^s]^2 + [\epsilon_s]^2 + [D_i^{s(max)} + \sum_{j=1}^F C_j^{m_s} N_j^{m_s}/g_s]^2]$ is a constant.

The gap B_i/V can be close to zero by fixing ϵ_s and increasing V. Detailed proof is included in [13].

V. Conclusion

This paper investigates profit maximization strategies at individual selfish clouds in a cloud federation where VM trading happens across cloud boundaries. We adopt a truthful, individual-rational, ex-post budget-balanced double auction as the inter-cloud trading mechanism, and design a dynamic algorithm for each cloud to decide the best VM valuation and bidding strategies, and to schedule job service/drop and server provisioning in the most economic fashion, under time-varying job arrivals and operational costs. The proposed algorithm can obtain a time-averaged profit for each cloud within a constant gap from its offline maximum, based on solid theoretical analysis.

Acknowledgements

The research was supported in part by a grant from Hong Kong RGC under the contract HKU 717812E.

References

[1] L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *Prof. of IEEE INFOCOM'10*, 2010.

[2] R. Urgaonkar, U. Kozat, K. Igarashi, and M. Neely, "Dynamic resource allocation and power management in virtualized data centers," in *Prof. of IEEE/IFIP NOMS'10*, 2010.

[3] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," in *Proc. of IEEE INFOCOM'12*, 2012.

[4] R. Buyya, D. Abramson, and J. Giddy., "Nimrod/g: An architecture of a resource management and scheduling system in a global computational grid," in *Proc. of HPC Asia'00*, 2000.

[5] X. Zhou and H. Zheng, "Trust: A general framework for truthful double spectrum auctions," in *Proc. of IEEE INFOCOM'09*, 2009.

[6] H. Xu, J. Jin, and B. Li, "A secondary market for spectrum," in *Proc. of IEEE INFOCOM'10, Mini Conference*, 2010.

[7] [Online]. Available: http://aws.amazon.com/ec2

[8] M. Mihailescu and Y. M. Teo, "Dynamic resource pricing on federated clouds," in *Proc. of IEEE/ACM CCGrid'10*, 2010.

[9] ——, "The impact of user rationality in federated clouds," in *Proc. of IEEE/ACM CCGrid'12*, 2012.

[10] [Online]. Available: http://www.linode.com/faq.cfm

[11] M. J. Neely, "Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks," in *Proc. of IEEE INFOCOM'11*, 2011.

[12] U. Hoelzle and L. A. Barroso, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines.* Morgan & Claypool, 2009.

[13] H. Li, C. Wu, Z. Li, and F. C. Lau, "Profit-maximizing virtual machine trading in a federation of selfish clouds," The University of Hong Kong, http://i.cs.hku.hk/∼hxli/profit-federation.pdf, Tech. Rep., 2012.

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