Dynamic Optimal Task Scheduling in Federated Hybrid Clouds

Xuanjia Qiu*, Hongxing Li*, Chuan Wu*, 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 Calgary, Canada, zongpeng@ucalgary.ca

***Abstract*—**

I. Introduction

Although cloud computing provides an illusion of unlimited resource pool, there still exists inherently limited scalability of single-provider clouds because a cloud provider provisioning capacity to meet the spike demand would suffer from under-utilization of resource. Therefore there is a need to federate multiple cloud providers to form a seamless computing resource pool and resource can be tapped into any cloud provider who can not serve requests locally at any time.

Even if public cloud service prevails, on-premise server cluster would still exist for many years. Idle computing resource at cloud users becomes a waste if it is not pooled into the global cloud and transferred to other users in need. We envision clouds would follow the development path like Internet which have developed from a conventional client-server architecture to Web 2.0. A cloud user equipped with racks of local computing servers would be tapped into the cloud, to form a hybrid architecture consisting of public clouds and locally managed computing resource from users at the edge of the Internet.

Because requests of computing resource could be served at either local data center, VMs contributed by user groups, or routed to other cloud providers, so there exists an optimal scheduling that maximizes the revenue of the community of cloud providers. We first model the system as a queueing network where requests for executing tasks enter the queues and departure when there are proper VMs to process them. In this framework, we consider the impact of buying and selling prices of VMs between clouds and users on user willingness to buy and to sell. We also consider that the help among clouds should be utilized in a fair way. We then adopt Lyapunov optimization technique to develop algorithms which dynamically decides selling and buying prices of VMs between clouds and users and the scheduling of requests among the cloud community. The analytical results show that our dynamic algorithm achieves social welfare with a small constant gap away from the optimality.

II. Related work

Reservoir project is motivated by the vision of implementing an architecture that would enable providers of cloud infrastructure to dynamically partner with each other to create a seemingly infinite pool of IT resources while preserving their individual autonomy in making technological and business management decisions [1]. The goal is to facilitate an open service-based online economy in which resources and services are transparently provisioned and managed across clouds on an on-demand basis at competitive costs with high-quality service.

III. The Problem

A. System Model

We consider a geo-distributed federated hybrid cloud consisting of a number of cloud providers, denoted as set $\mathcal{C}$. Each cloud provider has a group of subscribing users, who send requests and sell their idle resource to the cloud provider. Suppose the system executes in a time-slotted fashion. Each time slot is one unit time that is enough for processing a request.

We assume at time slot t, in subscribing user group n, $a_n(t)$ users have demand to rent VMs from cloud provider n, which is assumed to be an arbitrary random process over time, with upper bound A_n. Cloud provider n determines the price $i_n(t)$ it will charges from each user for a VM. A non-increasing demand curve $e(\cdot)$ tells us that only a portion $e(i_n(t))$ of users would accept such a price and stay to buy the service.

For accepted requests, cloud providers could process by itself, or ask for help from other cloud providers to process when its data center is fully utilized. If the cloud provider n decides to process it by itself, the request would be placed in its request queue $Q_n^{(n)}(t)$. Otherwise, it would be routed to one of the other cloud, say, cloud m, where the request would be placed in request queue $Q_n^{(m)}(t)$. The number of requests routed from user group n to cloud provider m is $r_n^{(m)}$.

In each time slot, cloud provider n determine the departure of the request queues $Q_n^{(m)}(t), \forall n \in \mathcal{C}$. The departure of queue $Q_n^{(m)}(t)$ has two destinations: $c_n^{(m)}(t)$ requests are processed at its data center, and $u_n^{(m)}(t)$ are processed at the VMs contributed by its users. We assume the capacity of data center is fixed. And the percentage of users willing to sell VMs to cloud in the population with size U_n is determined by the price $h_n(t)$ offered by the cloud provider n and a non-decreasing function $v(\cdot)$.

The system model is illustrated in Fig.1.

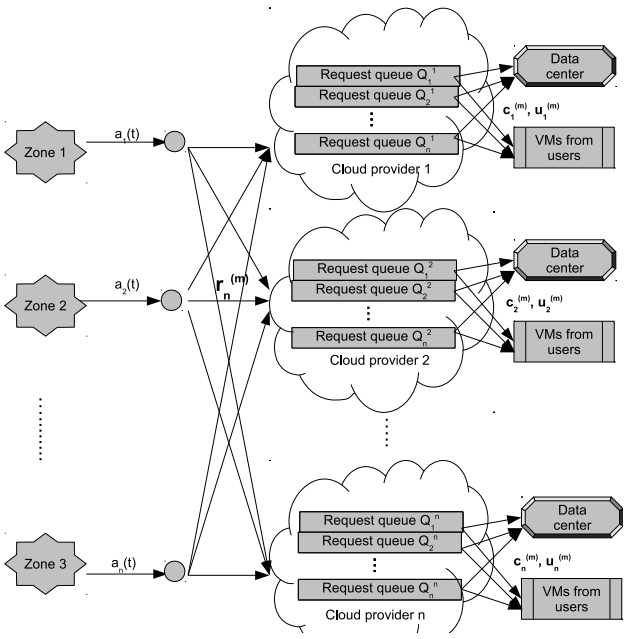

Fig. 1. Illustration for the system model.

To simplify the model, we assume each user request consumes one unit of computing resource. To guarantee that cloud providers help each other in a fair way, we should enforce that any pair of cloud providers should help each other process equal number of requests in the long time.

The important notations are summarized in Table I.

Each cloud provider $n, \forall n \in \mathcal{C}$ maintains $|C|$ queues, i.e., $Q_n^{(m)}, \forall m \in \mathcal{C}$. The law of queue update is as follows:

$$Q_n^{(m)}(t+1) = max[Q_n^{(m)}(t) - c_n^{(m)}(t) - u_n^{(m)}(t), 0] + r_n^{(m)}(t). \quad (1)$$

B. Problem Formulation

Let $\overline{x} = lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} x(t)$ represents the time-average value of $x(t)$.

We define social welfare of the all cloud providers the following:

$$\begin{aligned} G_{\mathbf{1}}(t) \quad = \quad & \sum_{n\in\mathcal{C}} [\sum_{m\in\mathcal{C}} r_n^{(m)}(t) i_m(t) \\ & - h_n(t) \sum_{m\in\mathcal{C}} u_n^{(m)}(t) \\ & - k_n(t) \sum_{m\in\mathcal{C}} c_n^{(m)}(t)] \end{aligned} \quad (2)$$

, which is the income for requests of all users, minus the cost of processing requests in data centers and payment to users for serving requests.

The social welfare of all cloud users is defined as follows:

TABLE I
IMPORTANT NOTATIONS

$a_n(t)$	Number of arrival requests from region n in time slot t.
$r_n^{(m)}(t)$	Number of requests accepted by cloud provider n from cloud provider m. If $n == m$, it represents the requests would be processed locally.
$Q_n^{(m)}(t)$	Backlog of request queue at cloud provider n buffering requests from region m.
$c_n^{(m)}(t)$	Number of requests departing from request queue $Q_n^{(m)}$ and being processed at data center.
$u_n^{(m)}(t)$	Number of requests departing from request queue $Q_n^{(m)}$ and being processed at user contributed VMs.
$v(\cdot)$	A non-decreasing function describing percentage of users who are willing to sell VMs where the parameter is the price a cloud offers
$e(\cdot)$	A non-increasing function describing percentage of users who are willing to buy VMs where the parameter is the price a cloud charges
$i_n(t)$	(end-user price) Charging rate from users for a request by cloud provider n at time slot t.
$i'(t)$	(user utility) expected utility of submitting a request by the user.
$\theta(\cdot)$	Utility function of users. The parameter is the number of requests accepted by the clouds.
$k_n(t)$	(data-center cost) Cost of processing a request at the data center of cloud provider n at time slot t.
$h_n(t)$	(user-vm price) Payment to users for serving a request from cloud provider n at time slot t.
$h'(t)$	(user-actual-vm cost) Actual cost for users to lend a VM to clouds.
A_n	Maximum number of arrival requests at cloud provider n
C_n	Capacity of data center at cloud provider n
U_n	Maximum number of VMs contributed by users at cloud provider n
$H_m^{(n)}$	Virtual queue which counts the number of requests processed by cloud n for m minus that processed by cloud m for n.

$$\begin{aligned} G_{\mathbf{2}}(t) \quad = \quad & \theta(\sum_{n\in\mathcal{C}} \sum_{m\in\mathcal{C}} r_n^{(m)}(t)) \\ & - \sum_{n\in\mathcal{C}} \sum_{m\in\mathcal{C}} r_n^{(m)}(t) i_m(t) \\ & + \sum_{n\in\mathcal{C}} \sum_{m\in\mathcal{C}} h_n(t) u_n^{(m)}(t) \\ & - \sum_{n\in\mathcal{C}} \sum_{m\in\mathcal{C}} h'(t) u_n^{(m)}(t) \end{aligned} \quad (3)$$

which is the utility gained from accepted requests, minus the payment from users to clouds, plus the money earned by selling VMs to clouds and minus the actual cost for running VMs.

The global social welfare is the sum of $G_{\mathbf{1}}$ and $G_{\mathbf{2}}$, i.e.,

$$\begin{aligned} G_{\mathbf{12}}(t) \quad = \quad & \theta(\sum_{n\in\mathcal{C}} \sum_{m\in\mathcal{C}} r_n^{(m)}(t)) \\ & - \sum_{n\in\mathcal{C}} h'(t) \sum_{m\in\mathcal{C}} u_n^{(m)}(t) \\ & - \sum_{n\in\mathcal{C}} k_n(t) \sum_{m\in\mathcal{C}} c_n^{(m)}(t) \end{aligned}$$

The simplest form of $\theta(x)$ is $\theta(x) = i'(t)x$. Then,

$$\begin{aligned} G_{\mathbf{12}}(t) &= \sum_{n\in\mathcal{C}}[\sum_{m\in\mathcal{C}} r_n^{(m)}(t)i'(t) \\ &\quad -h'(t)\sum_{n\in\mathcal{C}}\sum_{m\in\mathcal{C}} u_n^{(m)}(t) \\ &\quad -k_n(t)\sum_{n\in\mathcal{C}}\sum_{m\in\mathcal{C}} c_n^{(m)}(t)] \end{aligned} \tag{4}$$

Observe G_1 and G_{12} we find that if $\theta(x)$ is in the form of $\theta(x) = i'(t)x$, the *Cloud Community Social Welfare Optimization Problem* has a very similar structure as that of *Global Social Welfare Optimization Problem*.

The *Cloud Community Social Welfare Optimization Problem* (CC-SWOP) is defined as follows:

$$\max \quad \overline{G_{\mathbf{1}}(t)} \tag{5}$$

subject to:

$$\sum_{m\in\mathcal{C}} c_n^{(m)}(t) < C_n, \forall n \in \mathcal{C} \tag{6}$$

$$\sum_{m\in\mathcal{C}} u_n^{(m)}(t) < \lfloor U_n v(h_n(t))s_1 \rfloor, \forall n \in \mathcal{C} \tag{7}$$

$$\sum_{m\in\mathcal{C}} r_m^{(n)}(t) = \lfloor a_n(t)e(i_n(t))s_2 \rfloor, \forall n \in \mathcal{C} \tag{8}$$

$$\text{Queues } Q_n^{(m)}(t), \forall n, m \in \mathcal{C} \text{ are stable} \tag{9}$$

$$\overline{r_m^{(n)}} = \overline{r_n^{(m)}}, \forall m \le n, m, n \in \mathcal{C} \tag{10}$$

In (12) we assume that the capacity of the data center in a cloud is fixed. In (13) we assume that in the users with population size U_n, if the cloud offers a price $h_n(t)$, a portion $v(h_n(t))$ of users among them would be willing to sells VMs to the cloud. Each user holds s_1 VMs in expectation. In (14) we assume that in the arriving users with size $a_n(t)$, if a cloud charges $i_n(t)$ per VM, only a portion of $e(i_n(t))$ of them will stay. Each arriving user needs s_2 VMs in expectation. (16) guarantees that the number of requests processed by cloud n for m is equal to that by cloud m for n in the long time.

The decision variables in the whole problem are the number of requests $r_m^{(n)}, u_n^{(m)}, c_n^{(m)}$, and the prices $h_n(t), i_n(t), \forall m, n \in \mathcal{C}$

Meanwhile, The *Global Social Welfare Optimization Problem* (G-SWOP) is defined as follows:

$$\max \quad \overline{G_{\mathbf{12}}(t)} \tag{11}$$

subject to:

$$\sum_{m\in\mathcal{C}} c_n^{(m)}(t) < C_n, \forall n \in \mathcal{C} \tag{12}$$

$$\sum_{m\in\mathcal{C}} u_n^{(m)}(t) < \lfloor U_n v(h_n(t))s_1 \rfloor, \forall n \in \mathcal{C} \tag{13}$$

$$\sum_{m\in\mathcal{C}} r_m^{(n)}(t) = \lfloor a_n(t)e(i_n(t))s_2 \rfloor, \forall n \in \mathcal{C} \tag{14}$$

$$\text{Queues } Q_n^{(m)}(t), \forall n, m \in \mathcal{C} \text{ are stable} \tag{15}$$

$$\overline{r_m^{(n)}} = \overline{r_n^{(m)}}, \forall m \le n, m, n \in \mathcal{C} \tag{16}$$

The constraints and the variables are the same as CC-SWOP. However, the difference is that variables $h_n(t)$ and $i_n(t)$ do not appear in the objective function ($h'(t)$ and $k_n(t)$ in the objective functions are not variables).

IV. Dynamic Algorithms

In this section, we design dynamic control algorithms using Lyapunov optimization techniques, which solves CC-SWOP in (5) and G-SWOP in (11) .

A. Introducing Virtual Queues

To deal with (16), we introduce virtual queues $H_m^{(n)}, \forall m < n, m, n \in \mathcal{C}$, whose update law is:

$$H_m^{(n)}(t+1) = H_m^{(n)}(t) + r_m^{(n)}(t) - r_n^{(m)}(t). \tag{17}$$

The value of $H_m^{(n)}(t)$ could be positive or negative. If our dynamic algorithm can stabilize $H_m^{(n)}$, the time-average number of requests processed by cloud n for m is the almost the same as that processed by cloud m for n.

B. Distributed Dynamic Algorithm Design for CC-SWOP

In this sub-section, we apply Lyapunov Optimization techniques on SWOP to a design dynamic algorithm for optimal scheduling.

To deal with SWOP , we define $\Theta_s(t)$ as the vector of all queues $Q_n^{(m)}, \forall n, m \in \mathcal{C}$ and all virtual queues $H_m^{(n)}, \forall m \le n, m, n \in \mathcal{C}$.

Define our Lyapunov function as

$$L(\mathbf{\Theta}_s(t)) = \frac{1}{2}\sum_{n\in\mathcal{C}}(\sum_{m\in\mathcal{C}}(Q_n^{(m)}(t))^2 + \sum_{m\le n, m\in\mathcal{C}}(H_m^{(n)}(t))^2). \tag{18}$$

The one-slot conditional Lyapunov drift is

$$\Delta(\mathbf{\Theta}_s(t)) = \mathbb{E}\{L(\mathbf{\Theta}_s(t+1)) - L(\mathbf{\Theta}_s(t))|\mathbf{\Theta}_s(t)\}.$$

According to the *drift-plus-penalty* framework in Lyapunov optimization [2], an upper bound for the following expression should be minimized in each time slot, with the observation of the queue states $\mathbf{\Theta}_s(t)$, and data arrival rates $a_n(t), \forall j \in \mathcal{N}$, such that an upper bound for $\overline{G_1}$ is minimized (see Chapter 5 in [2]):

$$\Delta(\mathbf{\Theta}_s(t)) - VG_1(t).$$

Here, V is a non-negative parameter chosen by the community of clouds to control the tradeoff between optimal social welfare and service response delays. Squaring the queueing laws (1) and (17), we derive the following inequality:

$$
\begin{aligned}
& \Delta(\mathbf{\Theta_s(t)}) - VG_1(t) \\
\leq\ & B + \sum_{n\in\mathcal{C}}\sum_{m\in\mathcal{C}} Q_n^{(m)}(t)(r_n^{(m)}(t) - c_n^{(m)}(t) - u_n^{(m)}(t)) \\
& + \sum_{n\in\mathcal{C}}\sum_{m\leq n,m\in\mathcal{C}} H_m^{(n)}(t)(r_m^{(n)}(t) - r_n^{(m)}(t)) \\
& - V\sum_{n\in\mathcal{C}}\sum_{m\in\mathcal{C}}(r_n^{(m)}(t)i_n(t) - h_n(t)u_n^{(m)}(t) - k_n^{(m)}(t)c_n^{(m)}(t)) \\
=\ & B + \sum_{n\in\mathcal{C}}\sum_{m\leq n,m\in\mathcal{C}} r_n^{(m)}(t)(Q_n^{(m)}(t) + H_m^{(n)}(t) - Vi_n(t)) \\
& + \sum_{n\in\mathcal{C}}\sum_{m>n,m\in\mathcal{C}} r_n^{(m)}(t)(Q_n^{(m)}(t) - H_m^{(n)}(t) - Vi_n(t)) \\
& + \sum_{n\in\mathcal{C}}\sum_{m\in\mathcal{C}} c_n^{(m)}(t)(Vk_n^{(m)} - Q_n^{(m)}(t)) \\
& + \sum_{n\in\mathcal{C}}\sum_{m\in\mathcal{C}} u_n^{(m)}(t)(Vh_n(t) - Q_n^{(m)}(t)
\end{aligned} \tag{19}
$$

where B is a constant.

By minimizing the right-hand-side of inequality (19), that is the upper bound for $\Delta(\mathbf{\Theta_s}(t)) - VG_1(t)$, $\overline{G_1}$ could be minimized. To minimize right-hand-side of inequality (19), we design decompose the optimization problem trivially and solve the following optimization problem at each cloud provider n respectively:

$$
\begin{aligned}
\min\quad & \sum_{m<n,m\in\mathcal{C}} r_m^{(n)}(t)(Q_m^{(n)}(t) + H_m^{(n)}(t) - Vi_n(t)) \\
& + r_n^{(n)}(Q_n^{(n)}(t) - Vi_n(t)) \\
& + \sum_{m>n,m\in\mathcal{C}} r_m^{(n)}(t)(Q_m^{(n)}(t) - H_n^{(m)}(t) - Vi_n(t)) \\
& + \sum_{m\in\mathcal{C}}[c_n^{(m)}(t)(Vk_n^{(m)} - Q_n^{(m)}(t)) \\
& + u_n^{(m)}(t)(Vh_n(t) - Q_n^{(m)}(t)]
\end{aligned} \tag{20}
$$

Subject to:

$$(13)(12)(14)$$

We could further decompose the above optimization problem into two separate problems:

sub-problem 1:

$$
\begin{aligned}
\min\quad & \sum_{m<n,m\in\mathcal{C}} r_m^{(n)}(t)(Q_m^{(n)}(t) + H_m^{(n)}(t) - Vi_n(t)) \\
& + r_n^{(n)}(Q_n^{(n)}(t) - Vi_n(t)) \\
& + \sum_{m>n,m\in\mathcal{C}} r_m^{(n)}(t)(Q_m^{(n)}(t) - H_n^{(m)}(t) - Vi_n(t))
\end{aligned} \tag{21}
$$

Subject to:

$$\textstyle\sum_{m\in\mathcal{C}} r_m^{(n)}(t) = \lfloor a_n(t)e(i_n(t))s_2 \rfloor \tag{22}$$

Analysis: First we relax $\sum_{m\in\mathcal{C}} r_m^{(n)}(t) = \lfloor a_n(t)e(i_n(t))s_2 \rfloor$ to be $\sum_{m\in\mathcal{C}} r_m^{(n)}(t) = a_n(t)e(i_n(t))s_2$

Denote $\alpha_m^{(n)} = Q_m^{(n)}(t) + H_m^{(n)}(t)$, if $m < n$; $\alpha_m^{(n)} = Q_n^{(n)}$, if $m = n$; $\alpha_m^{(n)} = Q_m^{(n)}(t) - H_n^{(m)}(t)$, if $m > n$.Assume $\alpha_s = min\{\alpha_m^{(n)} | m \in \mathcal{C}\}$.

(1) If $\alpha_s \leq 0$, $i_n^*(t) = 0$

(2) If $\alpha_s \geq \frac{i_n^{max}}{V}$, $i_n^*(t) = i_n^{(max)}$.

(3) If $0 < \alpha_s < \frac{i_n^{max}}{V}$, assuming $e(i_n(t)) = di_n(t)$, where d is a constant, then $i_n^*(t) = \frac{\alpha_s}{2V}$.

Because three cases all possibly occur, there need to be more than 2 prices.

and **sub-problem 2:**

$$\min\quad \sum_{m\in\mathcal{C}} c_n^{(m)}(t)(Vk_n^{(m)} - Q_n^{(m)}(t)) \tag{23}$$

Subject to:

$$\textstyle\sum_{m\in\mathcal{C}} c_n^{(m)}(t) \leq C_n. \tag{24}$$

and **sub-problem 3:**

$$\min\quad \sum_{m\in\mathcal{C}} u_n^{(m)}(t)(Vh_n(t) - Q_n^{(m)}(t) \tag{25}$$

Subject to:

$$\textstyle\sum_{m\in\mathcal{C}} u_n^{(m)}(t) \leq \lfloor U_n v(h_n(t))s_1 \rfloor. \tag{26}$$

In practical scheduling, according to the results of sub-problem 2 & 3, a request may be routed to the data center or users' VMs. Which destination should be preferred?

C. Distributed Dynamic Algorithm Design for G-SWOP

sub-problem 1:

$$
\begin{aligned}
\min\quad & \sum_{m<n,m\in\mathcal{C}} r_m^{(n)}(t)(Q_m^{(n)}(t) + H_m^{(n)}(t) - Vi'(t)) \\
& + r_n^{(n)}(Q_n^{(n)}(t) - Vi_n(t)) \\
& + \sum_{m>n,m\in\mathcal{C}} r_m^{(n)}(t)(Q_m^{(n)}(t) - H_n^{(m)}(t) - Vi'(t))
\end{aligned} \tag{27}
$$

Subject to:

$$\textstyle\sum_{m\in\mathcal{C}} r_m^{(n)}(t) = \lfloor a_n(t)e(i_n(t))s_2 \rfloor \tag{28}$$

and **sub-problem 2:**

$$\min\quad \sum_{m\in\mathcal{C}} c_n^{(m)}(t)(Vk_n^{(m)} - Q_n^{(m)}(t)) \tag{29}$$

Subject to:

$$\textstyle\sum_{m\in\mathcal{C}} c_n^{(m)}(t) \leq C_n. \tag{30}$$

and **sub-problem 3:**

$$\min\quad \sum_{m\in\mathcal{C}} u_n^{(m)}(t)(Vh'(t) - Q_n^{(m)}(t) \tag{31}$$

Subject to:

$$\textstyle\sum_{m\in\mathcal{C}} u_n^{(m)}(t) \leq \lfloor U_n v(h_n(t))s_1 \rfloor. \tag{32}$$

Compare between CC-SWOP and G-SWOP: the problem structures are similar. The difference is that there are no price variables in CC-SWOP's objective functions but there are for G-SWOP.

REFERENCES

[1] B. Rochwerger, D. Breitgand, E. Levy, a. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan, "The Reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 1–11, Jul. 2009.

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