Performance Comparison between Selfish and Cooperative Scheduling of Tasks in Federated Hybrid Clouds

Xuanjia Qiu*, 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. We then adopt Lyapunov optimization technique to develop algorithms which achieve performance with a small constant gap away from the optimality and comply with SLA in terms of queueing delay. After that, we show that local revenue maximization meet optimal social welfare.

In summary, we have the following analytical results:

- A ready-to-use algorithm for each cloud provider to maximize its local revenue.
- SLA(in terms of queueing delay) are guaranteed.
- Local revenue maximization meet optimal social welfare.

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 [**?**]. 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 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, user group n send $a_n(t)$ requests to cloud provider n. Upon receipt of a request, a cloud provider determines whether it will process the request or delegate it to other cloud provider. 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 $r_n^{(m)}$, which is assumed to be an arbitrary random process over time, with R_max being the upper bound of the number of requests arising from user group n. 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. The system model is illustrated in Fig.1.

To simplify the model, we assume each user request consumes one unit of computing resource. When computing resource is tapped from users or other cloud providers, the cloud

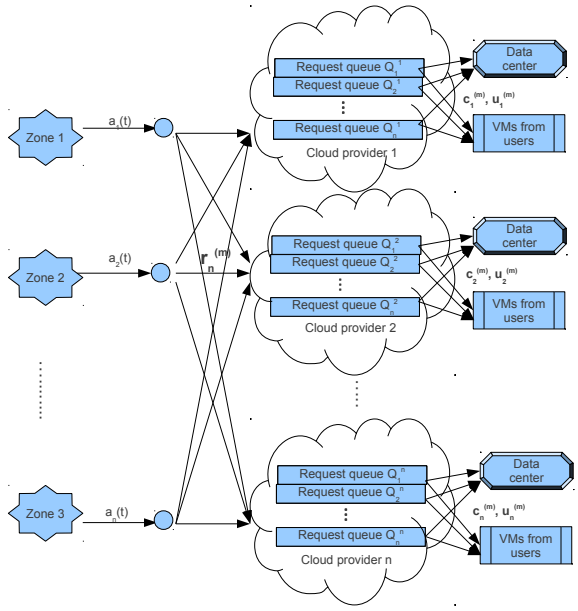

Fig. 1. Illustration for the system model.

provider would pay the other entity for processing the request at a pre-set price. Specifically, cloud provider n charges its user $i_n(t)$ dollars per task. Cloud provider n charges cloud provider m at the wholesale rate $b_n^{(m)}(t)$. The operational cost of executing a task in the data center at cloud provider n is $k_n(t)$. Cloud provider n pays its user for executing a task $h_n(t)$.

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:

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

$r_n^{(n)}$ is decision variable for cloud n, while $r_n^{(m)}, \forall m \in \mathcal{C}/n$ are all inputs. Some decision variables, i.e., $r_m^{(n)}, \forall m \in \mathcal{C}/n$, do not occur in the update equation of queues. Is this phenomenon OK?

TABLE I
IMPORTANT NOTATIONS

$a_n(t)$	Number of arrival requests 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 buffering requests routed from cloud provider n to cloud provider 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.
$i_n(t)$	Charging rate from users for a request by cloud provider n at time slot t.
$b_n^{(m)}(t)$	Charging rate for a request accepted by cloud provider n from cloud provider m at time slot t.
$k_n(t)$	Cost of processing a request at the data center of cloud provider n at time slot t.
$h_n(t)$	Payment to users for serving a request from cloud provider n at time slot t.
A_n	Maximum number of arrival requests at cloud provider n
C_n	Capacity of data center at cloud provider n
U_n	Number of VMs contributed by users at cloud provider n

B. Problem Formulation

For cloud provider n ($\forall n \in \mathcal{N}$), the profit at time slot t is

$$\begin{aligned} M(t) = \; & i_n(t) \sum_{m\in\mathcal{C}} r_m^{(n)}(t) \\ & - \sum_{m\in\mathcal{C}} b_m^{(n)}(t) r_m^{(n)}(t) \\ & + \sum_{m\in\mathcal{C}} b_n^{(m)}(t) r_n^{(m)}(t) \\ & - h_n(t) \sum_{m\in\mathcal{C}} u_n^{(m)}(t) \\ & - k_n(t) \sum_{m\in\mathcal{C}} c_n^{(m)}(t) \end{aligned} \quad (2)$$

, in which the income consists of income from users' requests and income for accepting requests from other cloud providers, while the expense consisting of expense for routing requests to other cloud providers and expense for payment to users for serving requests and cost of processing requests at its data center.

The decision variables are $r_m^{(n)}, u_n^{(m)}, c_n^{(m)}, \forall m \in \mathcal{C}$, while $r_n^{(m)}, \forall m \in \mathcal{C}/n$ are considered as inputs.

Each cloud provider n wants to maximize its time-average profit over a long term, which is called Local Profit Optimization Problem (LPOP). Let $\overline{x(t)} = lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} x(t)$ represents the time-average value of $x(t)$, then the LPOP for cloud provider n is expressed as follows:

$$\max \quad \overline{M(t)} \quad (3)$$

subject to:

$$\sum_{m\in\mathcal{C}} u_n^{(m)}(t) \le U_n, \forall t \tag{4}$$

$$\sum_{m\in\mathcal{C}} c_n^{(m)}(t) \le C_n, \forall t \tag{5}$$

$$u_n^{(m)}(t) + c_n^{(m)}(t) \le Q_n^{(m)}(t), \forall m \in \mathcal{C}, \forall t \tag{6}$$

$$\sum_{m\in\mathcal{C}} r_m^{(n)}(t) \le a_n(t), \forall t. \tag{7}$$

$$\text{Queues } Q_n^{(m)}(t), \forall m \in \mathcal{C} \text{ are stable.} \tag{8}$$

(4) means that the sum of requests dispatched from all queues in cloud provider n to the VMs contributed by users of cloud n should not exceed the number of VMs contributed by users of cloud provider n. (5) means that the sum of requests dispatched from all queues in cloud provider n to its data center should not exceed the capacity of its data center. (6) states that the number of requests dispatched from queue $Q_n^{(m)}$ cannot be larger than the current queue size. (7) states that the number of requests dispatched from user group n to all cloud providers cannot be larger than the number of arrival requests.

For the whole cloud community, the social welfare at time t is defined as:

$$\begin{aligned} G(t) \quad = \quad & \sum_n [\sum_m r_m^{(n)}(t) i_n(t) \\ & - h_n(t) \sum_{m\in\mathcal{C}} u_n^{(m)}(t) \\ & - k_n(t) \sum_{m\in\mathcal{C}} c_n^{(m)}(t)] \end{aligned}$$

, 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 whole cloud community hopes the time-average social welfare over the long term could be maximized, which is called Social Welfare Optimization Problem (SWOP). The SWOP could be expressed as follows:

$$\max \quad \overline{G(t)} \tag{9}$$

subject to:

$$\sum_{m\in\mathcal{C}} u_n^{(m)}(t) < U_n, \forall n \in \mathcal{C} \tag{10}$$

$$\sum_{m\in\mathcal{C}} c_n^{(m)}(t) < C_n, \forall n \in \mathcal{C} \tag{11}$$

$$u_n^{(m)}(t) + c_n^{(m)}(t) \le Q_n^{(m)}(t), \forall n, m \in \mathcal{C} \tag{12}$$

$$\sum_{m\in\mathcal{C}} r_m^{(n)}(t) \le a_n(t), \forall n \in \mathcal{C} \tag{13}$$

$$\text{Queues } Q_n^{(m)}(t), \forall n, m \in \mathcal{C} \text{ are stable.} \tag{14}$$

(10) (11) (12) (13).

IV. Dynamic Algorithms

In this section, we design dynamic control algorithms using Lyapunov optimization techniques, which solves LPOP's in (3) and SWOP in (9), and show that the performance gaps between LPOP and SWOP is small.

If we focus on the analysis on the performance gap of LPOP and SWOP, we may not need to add the delay-bounding mechanism.

A. Dynamic Algorithm Design for SWOP

In this sub-section, we apply Lyapunov Optimization techniques on SWOP and LPOP, and design dynamic algorithms for them respectively.

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

Define our Lyapunov function as

$$L(\mathbf{\Theta}_s(t)) = \sum_{n\in\mathcal{C}} \sum_{m\in\mathcal{C}} \frac{1}{2} (Q_n^{(m)}(t))^2). \tag{15}$$

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 [1], 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 $\overline{G(t)}$ is minimized (see Chapter 5 in [1]):

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

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 (**??**), we derive the following inequality:

$$\begin{aligned} & \Delta(\mathbf{\Theta}(\mathbf{t})_s) - VG(t) \\ \le \quad & B + \sum_{n\in\mathcal{C}} \sum_{m\in\mathcal{C}} Q_n^{(m)}(t)(r_n^{(m)}(t) - c_n^{(m)}(t) - u_n^{(m)}(t)) \\ & - V \sum_{n\in\mathcal{C}} \sum_{m\in\mathcal{C}} (r_m^{(n)}(t) i_n(t) - h_n(t) u_n^{(m)}(t) - k_n^{(m)}(t) c_n^{(m)}(t)) \\ = \quad & B + \sum_{n\in\mathcal{C}} \sum_{m\in\mathcal{C}} \\ & [r_n^{(m)}(t)(Q_n^{(m)}(t) - V i_m(t)) \\ & + c_n^{(m)}(t)(V k_n^{(m)} - Q_n^{(m)}(t)) \\ & + u_n^{(m)}(t)(V h_n(t) - Q_n^{(m)}(t)], \end{aligned} \tag{16}$$

where $B = \frac{1}{2}\sum_{n\in\mathcal{C}}(A_n^2 + (U_n + C_n)^2)$ is a constant.

By minimizing the right-hand-side of inequality (16), that is the upper bound for $\Delta(\mathbf{\Theta_s}(t)) - VG(t)$, $\overline{G(t)}$ could be minimized. To minimize right-hand-side of inequality (16), we design decompose the optimization problem trivially and solve the following optimization problem at each cloud provider n respectively:

$$\begin{aligned} \min \quad & \sum_{m\in\mathcal{C}} [r_n^{(m)}(t)(Q_n^{(n)}(t) - V i_m(t)) \\ & + c_n^{(m)}(t)(V k_n^{(m)} - Q_n^{(m)}(t)) \\ & + u_n^{(m)}(t)(V h_n(t) - Q_n^{(m)}(t)] \end{aligned}$$

Subject to:

$$(4)(5)(6)(7)$$

The above optimization problem is a Linear Programming (LP), which could be solved fast using some standard tools. The aggregate results of all cloud providers give the optimal schedule of tasks for SWOP.

B. Dynamic Algorithm Design for LPOP

To deal with LPOP, we define $\Theta_L(t)$ as the vector of all queues $Q_n^{(m)}, \forall n \in \mathcal{C}$.

Define the corresponding Lyapunov function as

$$L(\Theta_L(t)) = \mathbb{E}\{L(\Theta_L(t+1)) - L(\Theta_L(t))|\Theta_L(t)\}. \quad (17)$$

Then,

$$\begin{aligned} & \Delta(\Theta_L(t)) - VM(t) \\ \leq \; & B' + \sum_{m\in\mathcal{C}} [Q_n^{(m)}(t)(r_n^{(m)}(t) - c_n^{(m)}(t) - u_n^{(m)}(t)) \\ & - V(i_n(t)r_m^{(n)}(t) - b_m^{(n)}(t)r_m^{(n)}(t) + b_n^{(m)}(t)r_n^{(m)}(t) \\ & - h_n(t)u_n^{(m)}(t) - k_n(t)c_n^{(m)}(t))] \\ = \; & B'' + r_n^{(n)}(t)(Q_n^{(m)}(t) - Vb_n^{(n)}(t)) \\ & + \sum_{m\in\mathcal{C}} [c_n^{(m)}(t)(Vk_n(t) - Q_n^{(m)}(t)) + u_n^{(m)}(t)(Vh_n(t) - Q_n^{(m)}(t))] \\ & + \sum_{m\in\mathcal{C}/n} r_m^{(n)}(t)(Vb_m^{(n)}(t) - Vi_n(t)) \end{aligned} \quad (18)$$

where $B' = \frac{1}{2}[(U_n + C_n)^2 + \sum_{m\in\mathcal{C}} A_m^2]$ and $B'' = B' + \sum_{m\in\mathcal{C}/n} r_n^{(m)}(t)(Q_n^{(m)}(t) - Vb_n^{(m)}(t))$ are constants.

Following the techniques like the above, we derive the right-hand-side of the inequality as follows:

$$\begin{aligned} \min \quad & r_n^{(n)}(t)(Q_n^{(m)}(t) - Vb_n^{(n)}(t)) \\ & + \sum_{m\in\mathcal{C}} [c_n^{(m)}(t)(Vk_n(t) - Q_n^{(m)}(t)) + u_n^{(m)}(t)(Vh_n(t) - Q_n^{(m)}(t))] \\ & + \sum_{m\in\mathcal{C}/n} r_m^{(n)}(t)(Vb_m^{(n)}(t) - Vi_n(t)) \end{aligned} \quad (19)$$

Subject to:

$$(4)(5)(6)(7),$$

which is run on each cloud provider, in order to maximize the profit of each cloud provider.

Observe the above optimization problem, we find that $r_n^{(n)}$ is dependent on a queue length, but $r_m^{(n)}, \forall m \in \mathcal{C}/n$ do not. This means that the queues which accept $r_m^{(n)}, \forall m \in \mathcal{C}/n$ as arrivals may not be stable!!

V. Performance Analysis

A. Bound of Request Queue Length

B. Bound of Queueing Delay

C. Performance Gap between LPOP and SWOP

In this sub-section, we want to analyze the performance gap between our dynamic algorithms for LPOP and SWOP.

VI. Empirical Studies

VII. Conclusion

References

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