IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 2 June 2015; revised 12 November 2015; accepted 6 January 2016.
Date of Publication 13 January 2016; date of current version 8 June 2016.

Digital Object Identifier 10.1109/TETC.2016.2517930

Pricing and Repurchasing for Big Data Processing in Multi-Clouds

HE LI[1], MIANXIONG DONG[1], KAORU OTA[1], AND MINYI GUO[2], (Senior Member, IEEE)

[1]Department of Information and Electronic Engineering, Muroran Institute of Technology, Muroran 050-8585, Japan
[2]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

CORRESPONDING AUTHOR: M. DONG (mx.dong@ieee.org)

This work was supported in part by the National Natural Science Foundation of China under Grant 61261160502 and Grant 61272099, in part by the National Basic Research (973 Program) of China under Grant 2015CB352403, in part by the Japan Society for the Promotion of Science (JSPS) within the Grants-in-Aid for Scientific Research under Grant 15K15976 and Grant 26730056, in part by the Scientific Innovation Act of Science and Technology Commission of Shanghai Municipality under Grant 13511504200, in part by the JSPS through the A3 Foresight Program, and in part by the Research Fund for Post-Doctoral Program of Muroran Institute of Technology.

ABSTRACT Processing streaming big data becomes critical as new diver Internet of Thing applications begin to emerge. The existing cloud pricing strategy is unfriendly for processing streaming big data with varying loads. Multiple cloud environments are a potential solution with an efficient pay-on-demand pricing strategy for processing streaming big data. In this paper, we propose an intermediary framework with multiple cloud environments to provide streaming big data computing service with lower cost per load, in which a cloud service intermediary rents the cloud service from multiple cloud providers and provides streaming processing service to the users with multiple service interfaces. In this framework, we also propose a pricing strategy to maximize the revenue of the multiple cloud intermediaries. With extensive simulations, our pricing strategy brings higher revenue than other pricing methods.

INDEX TERMS Streaming big data, cloud computing, multiple cloud.

I. INTRODUCTION

Streaming big data processing is becoming a very important part of Internet of Things (IoT) in recent years. Usually, for lower maintenance cost, users often use cloud services for processing big data [1]–[3]. With cloud services, it is no need to maintain a large scale cluster and consider the details of big data computing. Furthermore, some cloud providers also provide computing services based on some popular distributed systems (e.g., Hadoop, etc.). With these services, users conveniently put their data and processing programs on the cloud platform then wait for the result [4].

Usually, cloud provides give users reasonable price for their services, especially for some long-term users [5]. However, for most streaming big data computing scenarios, their price, especially the rate per load, seems too expensive [6]. To reduce the cost for streaming big data computing, an optional method is choose some small cloud providers with lower rate per load. However, small cloud providers have not enough capacity to support large scale work loads [7], [8]. Meanwhile, their services are short of support for big data computing. Multiple cloud service mode is a better solution that users can deploy their computing in multiple cloud providers [9]. However, with multiple cloud providers, users have to considerate about the difficulty of management and the deployment of big data computing systems. Thus, it needs multiple cloud intermediaries to provide flexible services for these users to conveniently deploy data and processing programs.

Another problem is that the rate with long-term rent is much lower than the rate of pay-as-use while users choose long-term rent can get a lower rate. Usually, in many scenarios of streaming computing, the scale of the workload will vary in different periods drastically. If the users want to meet the requirement from the peak load, they need to rent many computing resources from cloud providers while most of rented resources will be idle with off-peak workloads. In this case, we consider a potential solution that intermediaries repurchase this part of computing capacity to recover a part of the user cost if possible.

Therefore, as shown in Fig. 1, we propose a multiple cloud intermediary concept combining multiple cloud providers and user subletting. This intermediary framework

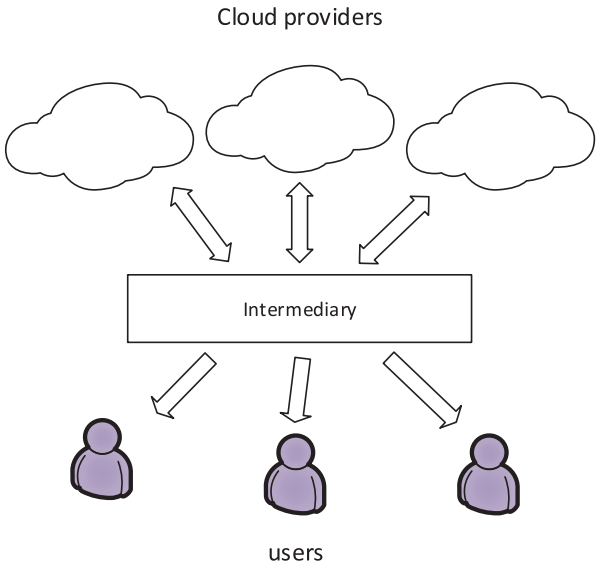


FIGURE 1. Illustration of the multiple cloud intermediary scheme: Cloud users rent cloud computing resources from the intermediary.

has compatibility with different cloud services and provides on-demand streaming processing services for users. Meanwhile, we present a well-designed pricing strategy names Pricing-Repurchasing for this intermediary framework. First, the intermediary can repurchase the sparse capacity with dynamic rate per load which depends on the duration time and the amount of sparse resources that the users hold. Second, the intermediary can choose different prices to users for different users according to the amount and time of computing resources rented. For this framework, we want to design the optimal pricing and subletting strategy for the intermediary that maximizes its total revenue, considering the necessary refunds to the users. Notice that changing the price or changing the repurchase rate has different impacts on renting decisions from users, hence on the intermediaries revenue. Thus, this pricing model brings new challenges in the design of the revenue maximizing policy for intermediary.

We employ a game-theoretic analysis, and model the interaction between the intermediary and the users as a two-stage leader-follower (Stackelberg) game. In the first stage, the intermediary decides the long-term rate, pay-as-use rate and the repurchase rate plan. Accordingly, in the second stage, every user decides how many units of computing capacity with long-term rate and how many units with pay-as-use rate. We analyze the best decisions of both the users and intermediary, and find the game equilibrium. The game model with equilibrium analysis uses a variety of system characteristics, including the computing style and scale of users, and the capacity of the intermediary. As a result, it is possible to apply the derivation of the optimal decisions to other cloud computing scenarios. The main contributions of our work are summaries as follows.

We first introduce an multiple cloud intermediary framework to provide streaming big data computing services. Based on this framework, we propose a pricing strategy with the Pricing-Repurchasing plan. To the best of our knowledge, this is the first work that studies such a Pricing-Repurchasing cloud service.

We then design the optimal Pricing-Repurchasing plan that maximizes the intermediary's revenue. It is a challenging problem which needs to understand thoroughly the impact of pricing and repurchasing strategies on the hosts renting decisions.

We model the interaction of the intermediary and the users as a two-stage Stackelberg game, and analyze the game equilibrium [10]. The analysis is generic and use a variety of system characteristics, and thus is applicable to various cloud computing scenarios.

Last, we take the performance evaluation of the strategy with extensive simulations, and discuss the revenue with different settings. We also compare our pricing strategy with some other pricing methods and the results shows our strategy performs better.

The rest of the paper is organized as follows. In section II, we discuss the related work. In Section III, we discuss the design concepts and brief the main structure in the framework. In Section IV, we state the system model. Then we analyze the optimal pricing and repurchasing policies in Section V. In Section 27, we present the simulations. Last, we conclude our work in Section.

II. RELATED WORKS

In this section, we first try to introduce the state of art and the tread of streaming big data computing. We also introduce some works which focused on the cloud based streaming big data processing. Finally, we discuss the some typical systems and scheduling algorithms in multiple cloud environment.

A. STREAMING BIG DATA COMPUTING

In rent years, researchers and companies developed some successful systems focus on streaming big data computing.

Earlier steaming processing systems are usually deployed on single computer. Aurora [11] is a streaming management system developed by the cooperation of Brown, Brandis, and MIT University. It is a single infrastructure which can efficiently and seamlessly support real-time monitoring applications, archival applications and spanning applications.

Borealis [12] is a distributed extension of Aurora which can process streaming data through multiple processors and computers. For support distributed architecture, Borealis presents an efficient algorithm for the distribution of jobs between nodes.

Cougar [13] is a streaming processing system that works with small-scale sensors, actuators, and embedded systems. Unlike general sensor networks use offline querying and analysis, Cougar project distributes queries to nodes and as a result only the desired data collected by the central processing nodes.

To meet the demands from the big data computing, large companies also developed some commercial streaming

processing system. For example, IBM InfoSphere [14] Streams is an advanced analytic platform that allows users develop applications for analyzing and correlating information from thousands of real-time sources. InfoSphere is a distributed runtime platform which can be scaled from a single server to an unlimited number of nodes to process millions of events per second. Microsoft StreamInsight [15] is another platform for developing and deploying complex event processing applications, which analyses and correlates data incrementally without storing data with low latency.

B. STREAMING BIG DATA COMPUTING IN CLOUD

While distributed and scalable cloud environment is very suitable for deploying streaming big data computing, existing cloud provides offers many solutions.

Meanwhile, some existing stream processing frameworks (e.g., Apache S4, Storm, IBM InfoSphere Streams, etc), which are designed for distributed systems, can be easily deployed to existing cloud environment [16].

Storm [17] is a clojure procject based on Pallet9, which aims to simplify the development of Storm topologies on cloud platforms including AWS EC2.

Apache Kafka [18] is a real-time publish-subscribe infrastructure aiming to address the requirements from streaming big data processing, in which data streams are partitioned and spread over a cluster of machines.

Meanwhile, since the cloud environment is different from the general distributed environment, more and more works focus on development of original cloud systems for processing streaming big data. Samze [19] is a streaming big data processing framework that blends Kafka and Hadoop YARN, which provides a model that YARN completely handle the execution where streams are the input and output ot jobs.

AWS Kinesis [20] is an cloud service provided from Amazon, which process stream data with the capacity to handl multiple sources. Kinesis is an efficient service especially on handling and generating alerts and allows for integration with other AWS services.

C. MULTIPLE CLOUD COMPUTING

Some existing works focus on integrate computing resources from multiple cloud providers.

Apache CloudStack [21] is a software to integrate cloud computing resources with resource management, user management, API and graphical user interface. Eucalyptus is also a similar software which focuses on building Amazon AWS-compatible private and hybrid clouds.

OpenNebula [22] is a multiple cloud software aiming at providing an industry standard solution for creating and managing virtual data centers across multiple cloud provides.

OpenStack [23] is the most famous cloud management system which provides an API and a dashboard to manage pools of computing, storage, and network resources from single or multiple cloud environment.

VMware vCloud [24] is a multiple cloud infrastructure that allowing to organize cloud computing at three levels including infrastructure level, platform level and service level.

FOG [25] is a Ruby API for providing access to computing and storage resources across multiple cloud provides. It also provides an in-memory cloud resource representation to help developers to test and simulate their deployment.

jcloud [26] is also an API for delivering an abstraction layer over the APIs from cloud providers, which facilitates users using means of templates to describe generic virtual machines and allows deploying and grouping of multiple virtual machines.

Cloud4SOA [27] is a multi-cloud PaaS management which enables software developers to create, deploy, execute, and mange business applications through multiple cloud providers.

The multicloud based evacuation services architecture [28] maintains basic monitoring and maintenance sercices during of normal activity but quickly scales up service capacity during an emergency.

Furthermore, besides multi-cloud frameworks and systems, there are several research works focused on the scheduling strategies between multiple cloud scenarios for optimization cost or performance.

An optimal virtual machine placement algorithm is proposed to minimize the total cost due to purchasing reserved and on-demand resources from multiple cloud providers [29]. In this research, an optimal strategy is explored to avoid the resources over/under-provisioning problem to cope with uncertainly demands. The goal of this algorithm is archived by adjusting the trade-off between resources and pay for the on-demand requirement of load peaks.

A management algorithm is presented to reallocate the placement of virtual machines for better performance in multiple cloud environment and optimize the resource utilization [30]. To archive this goal, the algorithm considers the host load profile and the guest load trend behavior instead of thresholds.

A modular broker architecture is proposed for optimal deployment for virtual services across multiple clouds with different scheduling strategies [31]. This optimization of this research is based on different criteria, different user constraints, and different environmental conditions.

A hybrid decision support is proposed for automating the migration of web application clusters to public clouds [32]. In this research, a selection algorithm based on analytic hierarchy process is designed for the migration decision over multiple clouds with several criteria. Further, a genetic algorithm-based approach is developed to cope with computational complexities in a growing market.

III. FRAMEWORK DESIGN

In this section, we first discuss the design concepts of the multiple cloud intermediary framework for streaming big data computing. Then, we brief the framework structure and introduce the main modules in the framework.

A. DESIGN CONCEPTS

1) MULTIPLE CLOUD COMPATIBILITY

Multiple cloud compatibility means the intermediary can rent computer resources from different cloud providers with different services, which means there are two levels of compatibility including platform compatibility and service compatibility.

Platform compatibility is that the intermediary applies the computer resources from multiple cloud platforms with different interfaces. This is the first design concept of the multiple cloud services that the users can deploy their applications to multiple cloud platforms transparently. The benefit of this compatibility is that the intermediary can schedule the resource requirement between multiple cloud providers to increase the service capacity and decrease the cost of the computer resources.

Service compatibility is that the intermediary applies the computer resources at different service levels. Usually, there are three levels of services from existing cloud providers, which including the Infrastructure as a Service (IaaS) level, MapReduce level and Streaming computing level. IaaS level means the cloud providers encapsulate their services as compute instances and the users use these instances as general servers. MapReduce level means the computing resources are provided as general MapReduce computing systems and users deploy their tasks as MapReduce applications. Streaming computing level is that the streaming computing applications can be executed in this cloud platform. Considering the intermediary focuses on the streaming computing, it can apply more flexible scheduling strategies due to the service compatibility.

2) ON-DEMAND SERVICES

On-demand services mean the intermediary can provide different service types to satisfy the user requirements. As well as the multiple cloud compatibility, there are also two levels of on-demand services including on-demand service levels and on-demand interfaces.

On-demand service levels mean the intermediary framework can provide the specific service level needed by users. In the discussion of the service compatibility, there are three service levels in general cloud providers. For the service levels, different users will adopt different levels for their tasks. For example, if users want to deploy their specific processing systems in the cloud platforms, they will choose IaaS level while if users want to execute their tasks on general streaming processing system, they will choose the streaming computing level. Thus, to satisfy the requirement of different users, the intermediary framework needs to provide these three service levels at least.

On-demand interfaces mean the intermediary framework can provide the specific service interface needed by users. Service interfaces are usually including the interfaces of the computing systems (e.g., POSIX [33], etc.), MapReduce systems (e.g., hadoop [34], etc.) and the streaming processing systems (e.g., SPARK [35], etc.). Before using the intermediary service, users usually have developed some applications or systems to execute their streaming processing tasks with specific interfaces. For example, if user developed their streaming processing applications on the Apache SPARK, they will prefer the cloud service through the same interfaces with the SPARK. Therefore, the intermediary framework needs to integrate general interfaces into the service levels.

3) SPECIFIC LONG-TERM RENTING

This design concept focuses on the revenue and the risk of the intermediary framework. Specific Long-term renting means the users subscribe the services from the intermediary framework with long-term contracts with specific prices. Similarly, there are also two levels in this concept including long-term renting and specific pricing.

Long-term renting means each user needs to rent a fix amount of computer resources with a long period. It is a little unacceptable that most of the cloud providers use pay-as-use mode which means users only need to pay the part of units they used. However, since the computing resources in the intermediary are also rented from the cloud providers, it is hard to decrease the cost of the pay-as-use mode. Thus, the intermediary need to rent the computer resources from multiple cloud providers with long-term contracts. Considering it is hard to predict user behaviors, long-term renting mode brings higher risk than pay-as-use mode that the revenue and cost are determined.

Specific pricing means the intermediary provide different price for users with their workloads or other factors of the processing tasks. The benefit of specific pricing is the intermediary can increase the revenue with better strategy and promote the cloud service to those users with more workloads or low cost processing mode.

We will discuss the first two concepts by introducing the framework structure first. Then, we will state the problem of the third concept and give a well-designed pricing strategy in the rest of this paper.

B. FRAMEWORK STRUCTURE

As the structure shown in Figure 2, the multiple cloud intermediary framework for streaming computing consists of several modules to meet the design concepts. There are seven main modules in the framework including the cloud instance management, streaming node management, MapReduce node management, streaming service, MapReduce service, IaaS service and user management modules.

Cloud instance management module manages all compute instances at the IaaS service level. This module records all status of the instances and assigns appropriate instances to other modules.

Streaming node management module manages the computing resources which are provided to users at streaming computing service level. The streaming computing resources are generated in three types of methods. First type is that

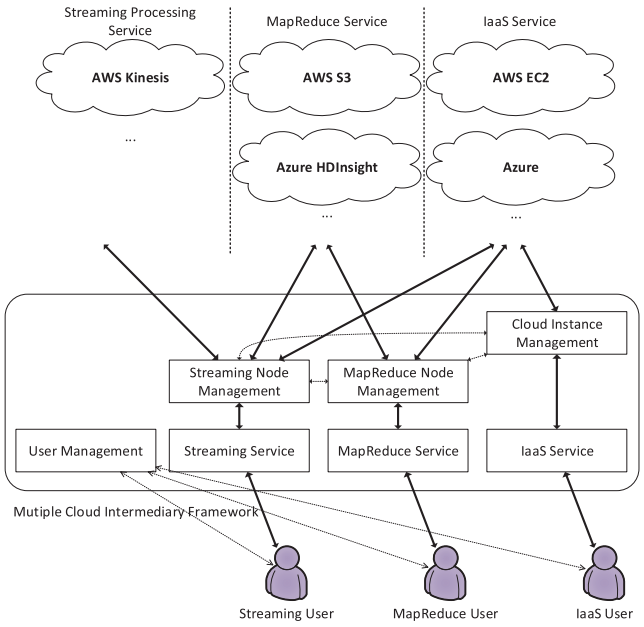


FIGURE 2. Multiple Cloud Intermediary Framework Structure.

the intermediary rents resources from the streaming processing cloud services. Second type is that the module deploys the streaming processing systems on the MapReduce nodes. Third type is that the module deploys the streaming processing systems in the compute instances directly.

MapReduce node management module manages the computing resources provided at MapReduce service level. Similarly with the streaming computing resources, the MapReduce computing resources are generated from two types: the resources rented from the cloud MapReduce services, and the module deploys the MapReduce systems in the compute instances.

Streaming service module provides streaming computing services to the users. To provide the required service interface from users, the streaming service module integrates general streaming processing systems.

MapReduce service module provides MapReduce services to the users. Similarly, the MapReduce service module integrates general MapReduce implementations to provide the compatible interfaces to support the streaming systems from users.

IaaS service module provides IaaS services to the users. Usually, users can get compute instances from this module with the required version of the operating systems and some necessary software.

User management module manages all users in the intermediary framework including access control, usage history, billing, etc.

IV. SYSTEM MODEL

The intermediary model is as shown in Figure 1. We consider users purchase cloud computing resources from the intermediary with enough capacity and low cost than large cloud providers. The intermediary get discount prices from multiple cloud provider with long-term contracts. The intermediary usually combines these computing resources into different big data computing systems as the service units for cloud users. We use set $N = \{u_1, u_2, ..., u_{|N|}\}$ to denote the cloud users who use computing resources from the intermediary. Since in processing streaming big data, the scale of workloads will vary with the time period, we assume a time-slotted system, and study the system for one time period and use $T = \{t_1, t_2, t_3, ...t_{|T|}\}$ to denote the T time slots.

The intermediary pays the cloud providers (e.g., Amazone EC2) a pay-as-use price $p \geq 0$ per one computing unit. Meanwhile, the intermediary charges the cloud users with long-term renting. As the intermediary provide rates for users according to the usage of cloud services, we use r_i^l to denote these different rates for user u_i. If users want pay-as-use rate, according to their usage and application style, they should pay r_i^p to bought additional computer resource from the market.

The intermediary repurchases the users computing capacity when the rented computer resources are more than the requirements. The repurchasing rate is not fixed, but depends on the amount of the over-rented capacity from the intermediary. We use $\eta_i \in [0, 0.9]$ to denote the repurchasing ratio to user u_i. When the intermediary repurchases one unit of computing capacity from user u_i, user u_i can get a refund of $\eta_i \cdot r_i^l$ from the intermediary. Similar with the rate strategies, we consider the intermediary can provide different repurchasing rates for users according to the computing scale and style.

The strategy of the intermediary includes the long-term renting price r_i^l and the repurchasing ratio η_i. The objective of the intermediary is to decide the best strategy to maximize the revenue. As we

For user u_i, we define a utility function $U_i(\cdot)$ to denote the certain computing needs. The utility function is defined to computes the utility of assignment resources to user i. As we seek a elastic model of the pricing strategy and the user utility function is compatible with multiple previous models [36], [37].

We also use w_{ij} to denote the consumed computing resources during slot t_j, and $w_i = (w_{ij}\, t_j \in T)$ as the computing resource consumption vector over the entire time period.

Each user u_i can rent computing resources with two different ways including long-term renting from the intermediary and pay-as-use from other cloud providers. Since the long-term renting price is much lower than pay-as-use price, each user needs to make a contract with the intermediary to get the long-term renting sale. As a result, in the entire time period, the amount of rented computing resources is fixed to each user u_i. We use $c_i^l \geq 0$ to denote fixed part of computing resources. Some users will choose pay-as-use mode to rent computing resources from other cloud providers as supplementary of long-term renting. We use $c_{ij}^p \geq 0$ to denote the part that user u_i choose pay-as-use mode to rent computing resources in time slot t_j. Considering the required amount of

computing resources is different in each time slot t_j, the part with pay-as-use mode is also different. The total computing resources of these two part are equal to the requirements of the workloads in time slot t_j as follows.

$$c_{ij}^{p} = \begin{cases} w_{ij} - c^{l}, & w_{ij} \geq c_i^{l} \\ 0, & c_i^{l} > w_{ij} \end{cases} \tag{1}$$

If the computing resources rented by user u_i within long-term renting mode exceed the requirement from the workloads, the intermediary will repurchase this part of the computing resources. Therefore, we use $c_{ij}^{r} \geq 0$ to denote the part of computing resources repurchased by the intermediary. That is, the repurchased computing resources during slot t can be calculated as follows.

$$c_{ij}^{r} = \begin{cases} c_i^{l} - w_{ij}, & c_i^{l} > w_{ij} \\ 0, & w_{ij} \geq c_i^{l} \end{cases} \tag{2}$$

We use $c_{ij}^{t} \geq 0$ to denote the total amount of the computing resources rented from the intermediary by user u_i in time slot t_j. With three parts of the computing resources, in time slot t_j, the total amount of the computing resources rented from the intermediary by user u_i should satisfy following equation.

$$c_{ij}^{t} = \begin{cases} c_{ij}^{l} + c_{ij}^{p} = w_{ij}, & w_{ij} \geq c_i^{l} \\ c_{ij}^{l} - c_{ij}^{r}, & c_i^{l} > w_{ij} \end{cases} \tag{3}$$

We list all notations used in the pricing strategy of the multiple cloud intermediary model in Table 1. The system is assumed to be quasi-static, as some variables (i.e., those marked with the subscript *j*) may change in different time slot $t_j \in T$, while others are fixed in the entire time period.

TABLE 1. Notations in the multiple cloud intermediary model.

Notation	Description
N	Set of all cloud users
u_i	Cloud user
T	Set of all time slots
t_j	Time slot
p	Price of Intermediary paid to cloud providers
r_i^{l}	Price of long-term renting for user u_i
r_i^{p}	Price of additional usage for user u_i
η_i	Repurchasing ratio for user u_i
w_{ij}	Computing resources consumed by user u_i in time slot t_j
w_i	Computing resource consumption vector of user u_i in the entire time period
c_i^{l}	Long-term renting amount of user u_i
c_i^{p}	Rented amount within pay-as-use of user u_i
c_i^{r}	Repurchasing amount of user u_i
c_i^{t}	Total rented amount of user u_i

We focus on the interaction of the intermediary and the users, and formulate is as a two-stage leader-follower (Stackelberg) game. A stacklberg game is leadership model in economics in which the leader firm moves before the follower. In the game terms, the game players are a leader and a follower and they compete on quantity. Thus, in our model, the game players are the intermediary and the cloud user. In the first stage, the intermediary (leader) decides the long-term renting price, the pay-as-use price and the repurchasing ratio for maximizing its payoff. The object of the intermediary is to maximize its payoff, which consist of the revenue from the long-term renting, pay-as-use renting, and the cost for repurchasing from the cloud users, and the payment(negative) to the cloud providers. In the second stage, under the decisions from the leader, the user u_i decides the long-term renting amount. The payoff of each user u_i depends on the utility U_i from the computing requirement, the payment on the long-term renting, the payment on the pay-as-use cost, and the refund from repurchasing of over rented computing resources.

Specifically, given the strategy (r^{l}, η) of the intermediary, the payoff of user u_i, when choosing a strategy (c^{l}), is as follows.

$$\begin{aligned} J_i(c_i^{l}; r_i^{l}, \eta_i) = {} & U_i(w_i) - r_i^{l} \cdot c_i^{l} \cdot |T| \\ & - \sum_{j=1}^{|T|} r_i^{p} \cdot c_{ij}^{p} + \sum_{j=1}^{|T|} \eta_i \cdot r_i^{l} \cdot c_{ij}^{r} \end{aligned} \tag{4}$$

From equation (1) and (2), the payoff of user u_i can be denoted as follows.

$$J_i(c_i^{l}; r_i^{l}, \eta_i) = \begin{cases} U_i(w_i) - r_i^{l} \cdot c_i^{l} \cdot |T| \\ - \sum_{j=1}^{|T|} r_i^{p} \cdot (w_{ij} - c_i^{l}), & w_{ij} \geq c_i^{l}, \\ U_i(w_i) - r_i^{l} \cdot c_i^{l} \cdot |T| \\ + \sum_{j=1}^{|T|} \eta_i \cdot r_i^{l} \cdot (c_i^{l} - w_{ij}), & w_{ij} < c_i^{l} \end{cases} \tag{5}$$

Formally, the intermediary's payoff can be defined as follows.

$$V(r^{l}, \eta; (c_i^{l})_{u_i \in U}) = \sum_{i=1}^{|U|} \sum_{j=1}^{|T|} r_i^{l} \cdot c_i^{l} - \eta_i \cdot r^{l} \cdot c_{ij}^{r} - p \cdot c_{ij}^{l} \tag{6}$$

Similar with the payoff of cloud users, the payoff of the intermediary can be denoted as follows.

$$\begin{aligned} & V(r^{l}, \eta; (c_i^{l})_{u_i \in U}) \\ & = \begin{cases} \sum_{i=1}^{|U|} r_i^{l} \cdot (c_i^{l} - p) \cdot |T|, & w_{ij} \geq c_i^{l} \\ \sum_{i=1}^{|U|} r_i^{l} \cdot (c_i^{l} - p) \cdot |T| \\ - \sum_{i=1}^{|U|} \sum_{j=1}^{|T|} \eta_i \cdot r_i^{l} \cdot (c_i^{l} - w_{ij}), & w_{ij} < c_i^{l} \end{cases} \end{aligned} \tag{7}$$

Considering users will choose cheaper price from the other cloud service, we assume that the intermediary provide a lower price than general cloud service. Meanwhile, we also assume the long-term price is lower the pay-as-use price. Therefore, we can get following constraints.

$$r_i^{l} < r_i^{p}, \quad i \in [1, |U|] \tag{8}$$

V. OPTIMAL PRICING-REIMBURSING STRATEGY

In this section, we study the intermediary-user game under complete information, where both the intermediary and the users know all system parameters mentioned above. We solve the game by backward induction. First, we solve the user's

best renting strategy in the second stage. Then, we study the intermediary's best pricing strategy in the first stage.

A. BEST DECISION OF USERS IN THE SECOND STAGE

We assume that computing tasks of the user are elastic that the analysis can be easily extended to other scenarios. Specifically, give the intermediary's pricing and repurchasing strategy (c_i^l, c_i^p, η_i), user u_i can derive the optimal scheduling strategy (c_i^{l*}) by solving the following problem.

$$\max_{c_i^l} J_i(c_i^l; r_i^l, \eta_i)$$
$$s.t., \ c_i^l \geq 0, \quad r_i^l < r_i^p, \ 0 \leq \eta_i \leq 0.9, \ i \in [1, |U|] \quad (9)$$

It is easy to check that (9) is a convex optimization. Meanwhile, there is no constraint for the value of c_i^l. Hence, usually it admits an optimal solution that can be characterized by the Fermat's theorem. However, considering the function $J_i(c_i^l; r_i^l, \eta_i)$ derived from a step function, we first study the characters of the payoff function.

First, we sort the w_i into numerical order and denote it by w_i^* in which $w_{i1}^* \leq w_{i2}^* \leq ... \leq w_{i|T|}^*$. To $c_i^l \leq w_{i1}^*$, the function $J_i(c_i^l; r_i^l, \eta_i)$ can be written as follows.

$$J_i(c_i^l; r_i^l, \eta_i) = U_i(w_i^*) + (r_i^p - r_i^l) \cdot |T| \cdot c_i^l - r_i^p \cdot \sum_1^T w_{ij}^*, \quad 0 \leq c_i^l \leq w_{i1}^* \quad (10)$$

It is easy to see it is a continuous and monotonic function where $0 \leq c_i^l \leq w_{i1}^*$.

Then we study the function where $c_i^l \geq w_{i|T|}^*$ as follows.

$$J_i(c_i^l; r_i^l, \eta_i) = U_i(w_i^*) - (1 - \eta_i) \cdot r_i^l \cdot |T| \cdot c_i^l + \eta_i \cdot r_i^l \cdot \sum_{j=1}^{|T|} w_{ij}^*, \quad c_i^l \geq w_{i|T|}^* \quad (11)$$

Obviously, the payoff function is continuous and monotonic where $c_i^l \geq w_{i|T|}^*$.

Then, given an interval $(w_{ik}^*, w_{i(k+1)}^*)$ where $w_{ik}^* < c_i^l < w_{i(k+1)}^*$, the payoff function of user u_i can be written as follows.

$$J_i(c_i^l; r_i^l, \eta_i) = U_i(w_i^*) - [(r_i^l - r_i^p) \cdot |T| - (\eta_i \cdot r_i^l - r_i^p) \cdot k] \cdot c_i^l + \eta_i \cdot r_i^l \cdot \sum_{j=1}^{k} w_{ij}^* - r_i^p \cdot \sum_{j=k+1}^{|T|} w_{ij}^*, \quad w_{ik}^* < c_i^l < w_{i(k+1)}^* \quad (12)$$

Therefore, the payoff function is continuous where $w_{ik}^* < c_i^l < w_{i(k+1)}^*$. Then, to the interval $(w_{ik}^*, w_{i(k+1)}^*)$, we denote the function $J_i'(c_i^l; r_i^l, \eta_i)$ to denote the derivative of the payoff function as follows.

$$J_i'(c_i^l; r_i^l, \eta_i) = \frac{dJ_i(c_i^l; r_i^l, \eta_i)}{dc_i^l} = (r_i^l - r_i^p) \cdot |T| - (\eta_i \cdot r_i^l - r_i^p) \cdot k \quad w_{ik}^* < c_i^l < w_{i(k+1)}^* \quad (13)$$

We can get the value of k^* after setting the $J_i'(c_i^l; r_i^l, \eta_i) = 0$ as follows.

$$k^* = \frac{(r_i^l - r_i^p) \cdot |T|}{\eta_i \cdot r_i^l - r_i^p} \quad (14)$$

As a result, to each interval $(w_{ik}, w_{i(k+1)})$ between w_{i1} and $w_{i|T|}$, the payoff function is continuous and monotonic except when $k = k^*$.

Lemma 1: The function $J_i(c_i^l; r_i^l, \eta_i)$ is a continuous function where $c_i^l \geq 0$.

Proof: As discussed above, the function $J_i(c_i^l; r_i^l, \eta_i)$ is continuous except $c_t^l = w_{ik}$ for each $t_k \in T$. Therefore, for a give $k \in (0, |T|)$, the value $J_i(w_{ik} + \Delta c; r_i^l, \eta_i)$ is as follows.

$$J_i(w_{ik} + \Delta c; r_i^l, \eta_i) = J_i(w_{ik}; r_i^l, \eta_i) + [(r_i^l - r_i^p) \cdot |T| - (\eta_i \cdot -r_i^p) \cdot k] \cdot \Delta c \quad (15)$$

For a given $k \in (0, |T|)$, the value $J_i(w_{ik} - \Delta c; r_i^l, \eta_i)$ is as follows.

$$J_i(w_{ik} + \Delta c; r_i^l, \eta_i) = J_i(w_{ik}; r_i^l, \eta_i) - [(r_i^l - r_i^p) \cdot |T| - (\eta_i \cdot -r_i^p) \cdot (k-1)] \cdot \Delta c \quad (16)$$

When $\Delta c \leftarrow 0$, since $\lim_{\Delta c \to 0} J_i(w_{ik} + \Delta c; r_i^l, \eta_i) = J_i(w_{ik}; r_i^l, \eta_i)$ and $\lim_{\Delta c \to 0} J_i(w_{ik} - \Delta c; r_i^l, \eta_i) = J_i(w_{ik}; r_i^l, \eta_i)$, the function is continuous where $c_t^l = w_{ik}$, $k \in (0, |T|)$. Similarly, we can prove the function $J_i(w_{ik}; r_i^l, \eta_i)$ is continuous where $c_t^l = w_{i1}$ and $c_t^l = w_{i|K|}$. Thus, we conclude that this function is continuous where $c_i^l \geq 0$. □

Lemma 2: The optimal solution of function $J_i(c_i^l; r_i^l, \eta_i)$ is $c_i^l = w_{i\lceil k^* \rceil}$, where $|T| \geq k^* > 0$

Proof: For the value where $c_p^l < w_{i\lceil k^* \rceil}$, we set $k = \lceil k^* \rceil - \delta < k^*$, the value of the function $J_i(c_i^l; r_i^l, \eta_i)$ is as follows.

$$J_i(c_i^l; r_i^l, \eta_i) = U_i(w_i^*) - [(r_i^l - r_i^p) \cdot |T| + (\eta_i \cdot r_i^l - r_i^p) \cdot \delta] \cdot c_i^l + \eta_i \cdot r_i^l \cdot \sum_{j=1}^{k} w_{ij}^* - r_i^p \cdot \sum_{j=k+1}^{|T|} w_{ij}^*, \quad w_{ik}^* < c_i^l < w_{i(k+1)}^* \quad (17)$$

Obviously, since $-[(r_i^l - r_i^p) \cdot |T| + (\eta_i \cdot r_i^l - r_i^p) \cdot \delta] < 0$ where $k^* > 0$, the function is monotonically decreasing. Similarly, when $c_i^l > w_{i(\lceil k^* \rceil + 1)}$, the function is monotonically increasing.

Considering k is an integer, we study two conditions of k^* that k^* is an integer or not. First, when k^* is an integer, we can get a interval $[w_{ik^*}, w_{i(k^*+1)}]$ in which the value of the payoff function is a constant. Therefore, when $c_i^l \in [w_{ik^*}, w_{i(k^*+1)}]$, the value of function $J_i(c_i^l; r_i^l, \eta_i)$ is minimum. When k^* is not an integer, we can get a interval $[w_{i\lceil k^*\rceil}, w_{i(\lceil k^*\rceil+1)}]$ in while the payoff function is monotonically increasing. That is, when $c_i^l = w_{i\lceil k^*\rceil}$, the value of function $J_i(c_i^l; r_i^l, \eta_i)$ is minimum. Finally, we can conduct that The optimal solution of function $J_i(c_i^l; r_i^l, \eta_i)$ is $c_i^l = w_{i\lceil k^*\rceil}$ where $|T| \geq k^* > 0$. □

B. BEST DECISION OF THE INTERMEDIARY IN THE FIRST STAGE

Based on the users' best strategy in the second stage, the intermediary determines the best pricing and repurchasing strategy (r^{l*}, η^*) that maximum the payoff defined in (7). Specifically, the intermediary's optimization problem is as follows.

$$\begin{aligned} &\max_{r^l,\eta} V(r^l, \eta; (c_i^{l*})_{u_i \in U}) \\ &s.t., \ c_i^{l*} \text{ is solved in (9)}, \quad c_i^l \geq 0, \\ &\quad r_i^l < r_i^p, \quad 0 \leq \eta_i \leq 1 \ \forall i \in [0, |U|] \end{aligned} \tag{18}$$

Since (c_i^{l*}) is the user u_i's best strategy under r_i^l, r_i^p and η_i, and c_i^{l*} is functions of r_i^l, r_i^p and η_i. That is, we can rewrite the intermediary's payoff as follows.

$$V(r^l, \eta; (c_i^{l*})) = \sum_{i=1}^{|U|} V_i(r_i^l, \eta_i; (c_i^{l*})) \tag{19}$$

With equation (14), the payoff function $V_i(r_i^l, \eta_i; (c_i^{l*}))$ can be written as follows.

$$\begin{aligned} V_i(r_i^l, \eta_i; (c_i^{l*})) = &[(r_i^l - p) \cdot |T| - \eta_i \cdot r_i^l \cdot \lceil k^* \rceil] \cdot w^*_{i\lceil k^*\rceil} \\ &+ \eta_i \cdot r_i^l \cdot \sum_{j=1}^{\lceil k^* \rceil} w_{ij}^* \end{aligned} \tag{20}$$

From the value of k^* in (14), we can get the payoff function as follows.

$$\begin{aligned} V_i(r_i^l, k^*; (w_{ik^*}^*)) = &[(r_i^l - p) \cdot |T| - \frac{\lceil k^* \rceil \cdot (r_i^l - r_i^p) \cdot |T|}{k^*} \\ &- r_i^p \cdot \lceil k^* \rceil] \cdot w^*_{i\lceil k^*\rceil} \\ &+ [\frac{(r_i^l - r_i^p) \cdot |T|}{k^*} + r_i^p] \cdot \sum_{j=1}^{\lceil k^* \rceil} w_{ij}^* \end{aligned} \tag{21}$$

To simplify this problem, we choose an approximation that $k^* = \lceil k^* \rceil$ which means k^* is an integer. With this approximation, the problem can be simplified as follows.

$$\begin{aligned} V_i(r_i^l, k^*; (w_{ik^*}^*)) = &[r_i^p \cdot (|T| - k^*) - p \cdot |T|] \cdot w_{ik^*}^* \\ &+ [\frac{(r_i^l - r_i^p) \cdot |T|}{k^*} + r_i^p] \cdot \sum_{j=1}^{k^*} w_{ij}^*, \\ &k^* \in [1, |T|] \end{aligned} \tag{22}$$

Considering k^* in an integer which is no more than T, we first maintain k^* is constant and study the optimal solution of r_i^l with a given k^*. That is, we can get the solution as follows.

$$\begin{aligned} V_i'(r_i^l, k^*; (w_{ik^*}^*)) &= \frac{dV_i(r_i^l, k^*; (w_{ik^*}^*))}{dr_i^l} \\ &= \frac{|T| \sum_{j=1}^{k^*} w_{ij}^*}{k^*} \end{aligned} \tag{23}$$

Since the derivative of the payoff function is always negative, this function is monotonically increasing with a given k^*. Therefore, the optimal solution is using a long-term renting price as max as possible. With a give $k*$, we can get r_i^l as follows.

$$r_i^l = \frac{r_i^p \cdot (|T| - k^*)}{|T| - k^* \cdot \eta_i} \tag{24}$$

It is easily find the maximum value of r_i^{l*} is $\frac{r_i^p \cdot (|T| - k^*)}{|T| - 0.9k^*}$ where $\eta*_i = 0.9$. Therefore, with a given k^*, we can get the maximum value of $V_i^*(k*; (w_{ik^*}^*))$ as follows.

$$\begin{aligned} V_i(k^*, 0.9; (w_{ik^*}^*)) = &[(r_i^{l*} - p) \cdot |T| - 0.9 \cdot r_i^{l*} \cdot k^*] \cdot w_{ik^*}^* \\ &+ 0.9 \cdot r_i^{l*} \cdot \sum_{j=1}^{k^*} w_{ij}^* \end{aligned} \tag{25}$$

After that, we study the optimal solution of k^* with a give c_i^l. Now we study the value of the payoff function with different given k^*. The incremental value that $V_i(k^* + 1; (w_{i(k^*+1)}^*)) - V_i(k^*; (w_{ik^*}^*))$ is as follows.

$$\begin{aligned} \Delta V_i &= V_i(k^* + 1; (w_{i(k^*+1)}^*)) - V_i(r_i^l, k^*; (w_{ik^*}^*)) \\ &= [r_i^p \cdot (|T| - k^*) - p \cdot |T|] \cdot (w_{i(k*+1)}^* - w_{ik^*}^*) \\ &\quad - r_i^p \cdot w_{i(k^*+1)}^* + 0.9\Delta(r^{l*} \cdot \sum_{j=1}^{k^*} w_{ij}^*) \end{aligned} \tag{26}$$

Unfortunately, since the varying value ΔV_i is related to the workload in each slot of user u_i, it is hard to describe the payoff function without detail workload. To illustrate the value of the payoff function, we calculate some distribution functions of the workload as shown in Figure 3.

In this example, we set the $c_i^p = 30$, $|T| = 720$ and $p = 2$ then use four distribution functions of the workload including normal distribution, Poisson distribution, Binomial distribution and random (average) distribution. The parameters of those distribution functions are dimensioned in the figure. From the value of these four distribution, the maximum value of the payoff function is related to the workload distribution. For example, with the random(average) distribution, we can get the maximum value of the payoff function when $k^* = 314$ while with the Poisson distribution, the maximum value can be get when $k^* = 14$.

Therefore, it needs to enumeration all values of the payoff function with $k^* \in [1, |T|]$ and find the maximum value of (25) with related k' as follows.

$$k' = \arg\max_{k^* \in [1, |T|]} (V_i(k^*; (w_{ik^*}^*)) \tag{27}$$

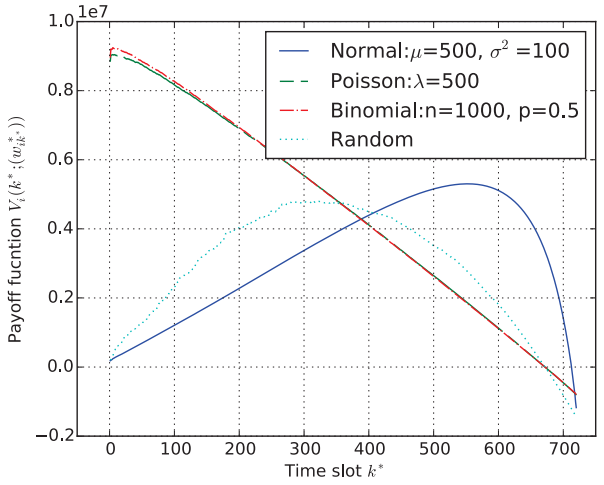


FIGURE 3. Value of the payoff function with different distribution function of the workload.

The time complexity of this enumeration is $O(|T|)$ which is an acceptable overhead to calculate the optimal solution.

With the result of k', we can get the optimal pricing r_i^l of the long-term renting to the user u_i is $\frac{r_i^p \cdot (|T|-k')}{|T|-k' \cdot 0.9}$ with a repurchasing ratio of $\eta_i = 0.9$.

VI. PERFORMANCE EVALUATION

In this section, we execute extensive simulations to evaluate the pricing strategy. We first describe the setting of the simulations then discuss the result of the performance evaluation.

We use a workstation computer as the simulation platform which equips a Core^{TM} i7 4770 (8M Cache, up to 3.90GHz) CPU, 16GByte RAM and 2TByte HDD. We use Python 2.7.3 as the script tools with networkx and numpy library. We test each simulation 20 times and record the average result.

In all simulations, we use 40 to 200 users as the N in the simulations and the time period T has 240 to 720 time slots. For comparison, we use two simple pricing strategies include pay-as-use mode and long-term renting mode as following.

(1) The pay-as-use mode pricing strategy uses a discount price of general pay-as-use mode price r_i^p per each user u_i from cloud providers. In the simulations, we use different discount ratio with 80%, 70% and 60%. Considering additional risks, the cost of this mode is 1.5 times of the cost of the long-term mode.

(2) The long-term mode pricing strategy uses a increased price on the cost for the intermediary renting computer resources from cloud providers. The incremental prices are set 5 cents and 10 cents per unit. To simplify the simulation, we assume users will rent average workload with the long-term mode.

We first take two simulations to study the general performance of our pricing strategy. We study the revenue of the proposed pricing strategy under different scales of users. We increase the number of users from 40 and 200 and in each step, the number of users increases 40. The cost p per units for renting computer resources from cloud providers is set 15 cents per unit. We set the workload amount w_{ij} per time slot t_j of each user u_i uniformly distributed in range [10, 1000]. The price c_i^p for each user u_i is uniformly distribute in range [35, 112] which is accepted price range according to existing cloud providers. As shown in Figure 4(a), the revenue of all pricing strategy increases with the user number scales up. When the number of users is 40, the revenue of the pricing-repurchase is near the pay-as-use mode of 80% while the number of users increases to 200, the difference between modes becomes larger.

We also study the revenue of the proposed pricing strategy under different service periods. We increase the number of time slots from 240 to 720 and in each step, the number of time slots increases 120. The number of user is set to 100 and other settings remain the same with previous. As shown in Figure 4(b), the revenue of the pricing-purchasing strategy is near to other modes when the number of time slot is set to 240. With longer service period, obviously, the revenue of our method performs better than other solutions. When the number of time slots increases to 720, the revenue of our strategy is near to 1 million dollars while pay-as-use 60% is near to the 500000.

After testing the overall performance, we study the revenue under different settings of the parameters of the pricing problem. We study the revenue of the proposed pricing strategy under different cost p per unit for renting computer resources from the cloud providers. The cost p per units increases from 5 cents per unit to 30 cents and in each step, the cost increases 5 cents. The number of users is set to 100. As shown in Figure 4(c), compared to other modes, the revenue of the Pricing-Repurchasing mode perform better with the increasing cost. The revenue of the long-term renting mode remains the same with the increasing cost. When the cost increases to 30 cents per unit, 60% discount price with the pay-as-use mode has less revenue than the $p + 10$ cents price with the long-term renting mode.

Then, we try to adjust the price c_i^p of the additional usage for each user. The price c_i^p for each user u_i increases from 40 cents to 120 cents and in each step, the price r_i^p, increases 20 cents. We still set the workload amount w_{ij} per time slot t_j of each user u_i uniformly distributed in range [10, 1000]. The cost p per unit for renting computer resources from cloud providers is set to 15 cents per unit. As shown in Figure 4(d), with the price in the cloud market increases, the revenue with the Pricing-Repurchasing and the pay-as-use mode is increased while the revenue of the long-term renting mode still remains the same. When the price r_i^p, is less than 60 cents, the $p + 10$ cents price with the long-term renting mode has more revenue than the 80% discount price with pay-as-use mode and the $p + 5$ cents price with long-term renting mode has more revenue than the 60% discount price with pay-as-use mode.

Third, we study the revenue of each pricing strategies with different workload of each user u_i. We set the

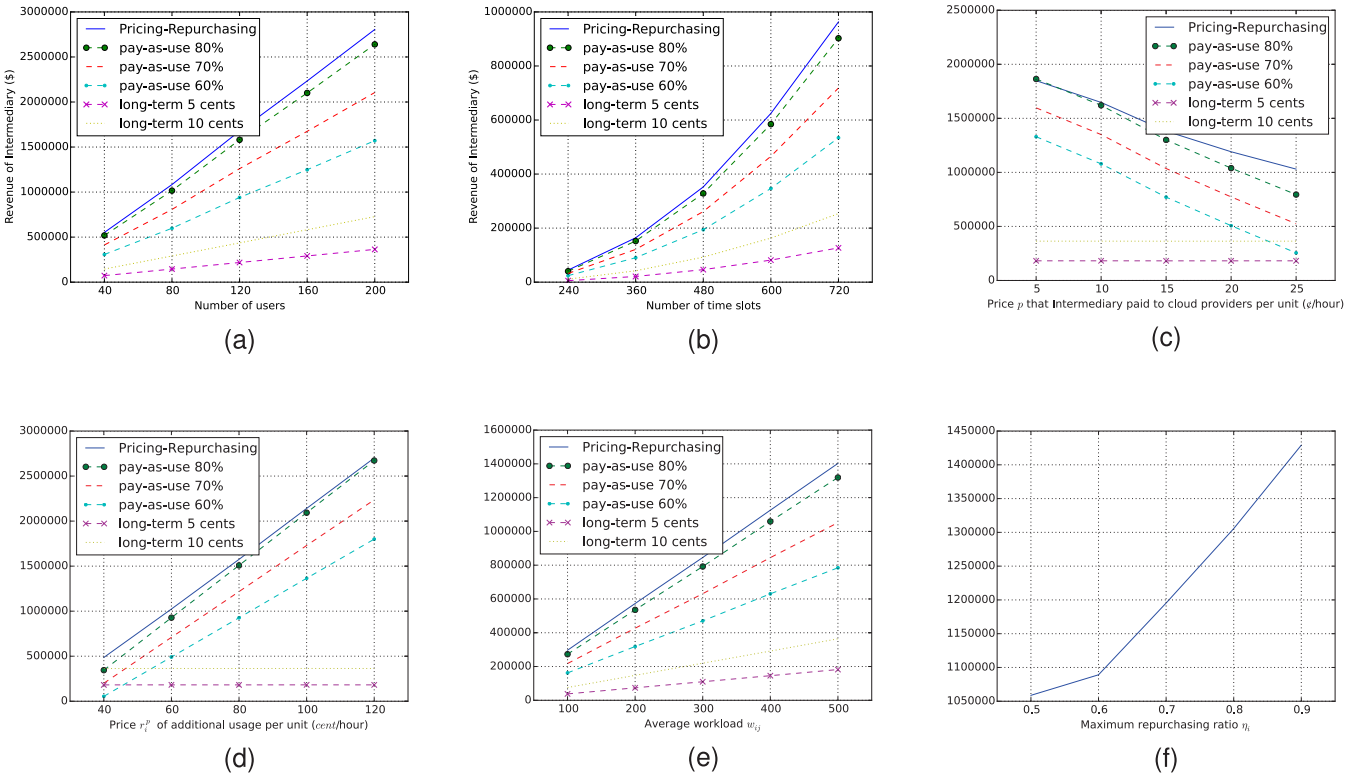


FIGURE 4. Revenue results with different user scales, service periods, and different settings. (a) Revenue with different number of users. (b) Revenue with different number of time slots. (c) Revenue with different price p from cloud providers. (d) Revenue with different price r_i^p of the additional usage. (e) Revenue with different average workload $\bar{w}_{ij}$. (f) Revenue with different maximum repurchasing ratio η_i.

average workload amount w_{ij} per time slot of user u_i increases from 100 to 500 and the average workload amount increases 100 in each step. We set the cost p per units to 15 and the price r_i^p for each user u_i uniformly distributed in range [35, 112]. As shown in Figure 4(e), the revenue with the Pricing-Repurchasing is still more than other pricing strategy. The rate of increasing revenue with the increasing workload with the Pricing-Repurchasing is higher than other pricing strategies. Differently from the previous simulations, the revenue of the long-term renting mode increases with the increasing workload event it is lowest in the all pricing strategies.

Since the repurchasing is very important to our pricing strategy, we test the revenue of different maximum repurchasing ratio η_i for studying the influence from repurchasing strategy. We set the workload amount w_{ij} uniformly distributed in range [10, 1000], the cost p per units to 15, and the price r_i^p, for each user u_i is uniformly distributed in range [35, 112]. As shown in Figure 4(f), obviously, the revenue of the Pricing-Repurchasing strategy increases with the increasing repurchasing ratio. With a repurchasing ratio of 0.9, the revenue increases 33% than the revenue of 0.5.

Finally, from the results of performance evaluation, we can conclude that the Pricing-Repurchasing strategy brings more revenue to the intermediary framework than other pricing strategy especially with more workloads, higher cost of the cloud resources and lower spreads between the cost and the price in the market. Further, That is, the Pricing-Repurchasing strategy can adapt the competitive cloud service market.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose a multiple cloud intermediary framework for streaming big data computing to provide streaming big data processing cloud services to the users. The intermediary rents computer resources from different cloud services and provides different service interfaces to users. We also design a Pricing-Repurchasing strategy to maximum the revenue of the intermediary and decrease the risks by long-term renting contracts with users. We formulate the Pricing-Repurchasing problem as a two-stage leader-follower (Stackelberg) game, and analyze the game equilibrium. We also evaluate our pricing strategy with extensive simulations and compare the revenue with other pricing strategies. From the result of performance evaluation, the Pricing-Repurchasing strategy brings more revenue to the intermediary than other methods.

In the future, we will plan to implement a complete multiple cloud intermediary solution with modified OpenStack to support streaming big data processing management. Meanwhile, it is signification to find scheduling

method to optimize the streaming computing performance in the multiple cloud environment. A deeper experiment with the real word testbed is also needed to evaluate the efficiency of the new multiple cloud intermediary solution.

REFERENCES

[1] L. Wang, Y. Ma, A. Y. Zomaya, R. Ranjan, and D. Chen, "A parallel file system with application-aware data layout policies for massive remote sensing image processing in digital earth," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1497–1508, Jun. 2015.

[2] Z. Deng *et al.*, "Parallel processing of dynamic continuous queries over streaming data flows," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, pp. 834–846, Mar. 2015.

[3] W. Xue *et al.*, "Ultra-scalable CPU-MIC acceleration of mesoscale atmospheric modeling on Tianhe-2," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2382–2393, Aug. 2015.

[4] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[5] 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. IEEE 13th Int. Conf. High Perform. Comput. Commun. (HPCC)*, Sep. 2011, pp. 279–287.

[6] Y.-J. Hong, J. Xue, and M. Thottethodi, "Dynamic server provisioning to minimize cost in an IaaS cloud," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Modeling Comput. Syst. (SIGMETRICS)*, 2011, pp. 147–148.

[7] M. Dong, H. Li, K. Ota, and H. Zhu, "HVSTO: Efficient privacy preserving hybrid storage in cloud data center," in *Proc. IEEE INFOCOM WKSHPS*, Apr./May 2014, pp. 529–534.

[8] H. Li, M. Dong, X. Liao, and H. Jin, "Deduplication-based energy efficient storage system in cloud environment," *Comput. J.*, vol. 58, no. 6, pp. 1373–1383, Jun. 2014.

[9] A. Iordache, C. Morin, N. Parlavantzas, E. Feller, and P. Riteau, "Resilin: Elastic MapReduce over multiple clouds," in *Proc. 13th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGrid)*, May 2013, pp. 261–268.

[10] S. van Hoesel, "An overview of Stackelberg pricing in networks," *Eur. J. Oper. Res.*, vol. 189, no. 3, pp. 1393–1402, 2008.

[11] D. J. Abadi *et al.*, "Aurora: A new model and architecture for data stream management," *VLDB J.*, vol. 12, no. 2, pp. 120–139, Aug. 2003.

[12] D. J. Abadi *et al.*, "The design of the borealis stream processing engine," in *Proc. CIDR*, 2005, pp. 277–289.

[13] J. Gehrke and S. Madden, "Query processing in sensor networks," *IEEE Pervasive Comput.*, vol. 3, no. 1, pp. 46–55, Jan. 2004.

[14] P. Zikopoulos and C. Eaton, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. New York, NY, USA: McGraw-Hill, 2011.

[15] M. Ali, B. Chandramouli, B. Sethu, and R. Katibah, "Spatio-temporal stream processing in microsoft StreamInsight," *IEEE Comput. Soc. Data Eng. Bull.*, vol. 33, no. 2, pp. 69–74, Jun. 2010.

[16] G. De Francisci Morales, "SAMOA: A platform for mining big data streams," in *Proc. 22nd Int. Conf. World Wide Web Companion*, 2013, pp. 777–778.

[17] J. Leibiusky, G. Eisbruch, and D. Simonassi, *Getting Started With Storm*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2012.

[18] N. Garg, *Apache Kafka*. Birmingham, U.K.: Packt Publishing, 2013.

[19] *Apache Samza*. [Online]. Available: http://samza.apache.org/, accessed Jun. 1, 2015.

[20] R. Ranjan, "Streaming big data processing in datacenter clouds," *IEEE Cloud Comput.*, vol. 1, no. 1, pp. 78–83, May 2014.

[21] *Apache Cloudstack*. [Online]. Available: https://cloudstack.apache.org/, accessed Jun. 1, 2015.

[22] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Capacity leasing in cloud systems using the OpenNebula engine," in *Proc. Workshop Cloud Comput. Appl.*, 2008, pp. 1–5.

[23] K. Pepple, *Deploying OpenStack*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2011.

[24] O. Krieger, P. McGachey, and A. Kanevsky, "Enabling a marketplace of clouds: VMware's vCloud director," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 4, pp. 103–114, Dec. 2010.

[25] *FOG—The Ruby Cloud Services Library*. [Online]. Available: http://fog.io/, accessed Jun. 1, 2015.

[26] M. Alrokayan and R. Buyya, "A Web portal for management of aneka-based multicloud environments," in *Proc. 11th Austral. Symp. Parallel Distrib. Comput. (AusPDC)*, vol. 140. 2013, pp. 49–56.

[27] F. D'Andria, S. Bocconi, J. G. Cruz, J. Ahtes, and D. Zeginis, "Cloud4SOA: Multi-cloud application management across paas offerings," in *Proc. 14th Int. Symp. Symbolic Numer. Algorithms Sci. Comput. (SYNASC)*, Sep. 2012, pp. 407–414.

[28] M. Dong, H. Li, K. Ota, L. T. Yang, and H. Zhu, "Multicloud-based evacuation services for emergency management," *IEEE Cloud Comput.*, vol. 1, no. 4, pp. 50–59, Nov. 2014.

[29] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Proc. IEEE Asia-Pacific Services Comput. Conf. (APSCC)*, Dec. 2009, pp. 103–110.

[30] M. Andreolini, S. Casolari, M. Colajanni, and M. Messori, "Dynamic load management of virtual machines in cloud architectures," in *Cloud Computing* (Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), vol. 34, D. Avresky, M. Diaz, A. Bode, B. Ciciani, and E. Dekel, Eds. Berlin, Germany: Springer, 2010, pp. 201–214.

[31] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Scheduling strategies for optimal service deployment across multiple clouds," *Future Generat. Comput. Syst.*, vol. 29, no. 6, pp. 1431–1441, 2013.

[32] M. Menzel, R. Ranjan, L. Wang, S. U. Khan, and J. Chen, "CloudGenius: A hybrid decision support method for automating the migration of Web application clusters to public clouds," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1336–1348, May 2015.

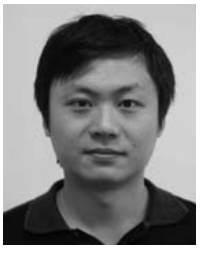
[33] D. R. Butenhof, *Programming With POSIX Threads*. Reading, MA, USA: Addison-Wesley, 1997.

[34] T. White, *Hadoop: The Definitive Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2012.

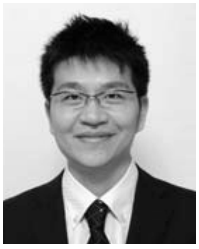
[35] M. Zaharia *et al.*, "Fast and interactive analytics over Hadoop data with spark," *USENIX Login*, vol. 37, no. 4, pp. 45–51, 2012.

[36] J. O. Kephart and R. Das, "Achieving self-management via utility functions," *IEEE Internet Comput.*, vol. 11, no. 1, pp. 40–48, Jan./Feb. 2007.

[37] N. W. Paton, M. A. T. de Aragão, K. Lee, A. A. A. Fernandes, and R. Sakellariou, "Optimizing utility in cloud computing through autonomic workload execution," *Bull. Tech. Committee Data Eng.*, vol. 32, no. 1, pp. 51–58, 2009.



HE LI received the B.S. and M.S. degrees in computer science and engineering from the Huazhong University of Science and Technology, in 2007 and 2009, respectively, and the Ph.D. degree in computer science and engineering from The University of Aizu, in 2015. He is currently a Post-Doctoral Fellow with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. His research interests include cloud computing and software defined networking. He serves as a Guest Associate Editor of *IEICE Transactions on Information and Systems*.



MIANXIONG DONG received the B.S., M.S., and Ph.D. degrees in computer science and engineering from The University of Aizu, Japan. He was a Researcher with the National Institute of Information and Communications Technology, Japan. He was a Japan Society for the Promotion of Sciences (JSPS) Research Fellow with the School of Computer Science and Engineering, The University of Aizu, and a Visiting Scholar with the BBCR Group, University of Waterloo, Canada, supported by the JSPS Excellent Young Researcher Overseas Visit Program from 2010 to 2011. He is currently an Assistant Professor with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. He was selected as a Foreigner Research Fellow (a total of three recipients all over Japan) by the NEC C&C Foundation in 2011. His research interests include wireless networks, cloud computing, and cyber-physical systems. His research results have been published in 120 research papers in international journals, conferences, and books. He received best paper awards from the IEEE HPCC 2008, the IEEE ICESS 2008, ICA3PP 2014, GPC 2015, and the IEEE DASC 2015. He is currently a Research Scientist with the A3 Foresight Program (2011-2016) funded by the JSPS, the NSFC of China, and the NRF of Korea. He serves as an Associate Editor of the IEEE Communications Surveys and Tutorials, the IEEE Network, the IEEE Access, and *Cyber-Physical Systems* (Taylor & Francis), and a Leading Guest Editor of *ACM Transactions on Multimedia Computing, Communications and Applications*, the IEEE Transactions on Emerging Topics in Computing, the IEEE Transactions on Computational Social Systems, *Peer-to-Peer Networking and Applications* (Springer), and *Sensors*, and a Guest Editor of *IEICE Transactions on Information and Systems*, *Mobile Information Systems*, and *International Journal of Distributed Sensor Networks*. He served as the Program Chair of the IEEE SmartCity 2015 and the Symposium Chair of the IEEE GLOBECOM 2016.

KAORU OTA received the B.S. degree in computer science and engineering from The University of Aizu, in 2006, the M.S. degree in computer science from Oklahoma State University, USA, in 2008, and the Ph.D. degree in computer science and engineering from The University of Aizu, in 2012. She is currently an Assistant Professor with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. From 2010 to 2011, she was a Visiting Scholar with the University of Waterloo, Canada. She was a Japan Society of the Promotion of Science Research Fellow with the Kato-Nishiyama Laboratory, Graduate School of Information Sciences, Tohoku University, Japan, from 2012 to 2013. Her research interests include wireless sensor networks, vehicular ad hoc networks, and ubiquitous computing. She serves as an Editor of *Peer-to-Peer Networking and Applications* (Springer), *Ad Hoc & Sensor Wireless Networks*, the *International Journal of Embedded Systems* (Inderscience), and the *Journal of Cyber-Physical Systems*, and a Guest Editor of the *IEEE Wireless Communications* and *IEICE Transactions on Information and Systems*. She is a Research Scientist with the A3 Foresight Program (2011-2016) funded by the Japan Society for the Promotion of Sciences, the NSFC of China, and the NRF of Korea.

MINYI GUO (SM'07) received the B.Sc. and M.E. degrees in computer science from Nanjing University, China, and the Ph.D. degree in computer science from the University of Tsukuba, Japan. He had been a Professor with the School of Computer Science and Engineering, The University of Aizu, Japan. He is currently a Zhiyuan Chair Professor and Chair of the Department of Computer Science and Engineering with Shanghai Jiao Tong University, China. His current research interests include parallel/distributed computing, compiler optimizations, embedded systems, pervasive computing, cloud computing, and big data. He has authored over 250 publications in major journals and international conferences in these areas. He is a member of ACM, IEICE, IPSJ, and CCF. He received the national science fund for distinguished young scholars from NSFC in 2007. He received five best paper awards from international conferences. He served as an Associate Editor of the IEEE Transactions on Parallel and Distributed Systems and the IEEE Transactions on Computers.