Optimized Contract-based Model for Resource Allocation in Federated Geo-distributed Clouds

Jinlai Xu *Student Member, IEEE,* Balaji Palanisamy *Member, IEEE,*

Abstract—In the era of Big Data, with data growing massively in scale and velocity, cloud computing and its pay-as-you-go model continues to provide significant cost benefits and a seamless service delivery model for cloud consumers. The evolution of small-scale and large-scale geo-distributed datacenters operated and managed by individual Cloud Service Providers (CSPs) raises new challenges in terms of effective global resource sharing and management of autonomously-controlled individual datacenter resources towards a globally efficient resource allocation model. Earlier solutions for geo-distributed clouds have focused primarily on achieving global efficiency in resource sharing, that although tries to maximize the global resource allocation, results in significant inefficiencies in local resource allocation for individual datacenters and individual cloud provi ders leading to unfairness in their revenue and profit earned. In this paper, we propose a new contracts-based resource sharing model for federated geo-distributed clouds that allows CSPs to establish resource sharing contracts with individual datacenters *apriori* for defined time intervals during a 24 hour time period. Based on the established contracts, individual CSPs employ a contracts cost and duration aware job scheduling and provisioning algorithm that enables jobs to complete and meet their response time requirements while achieving both global resource allocation efficiency and local fairness in the profit earned. The proposed techniques are evaluated through extensive experiments using realistic workloads generated using the SHARCNET cluster trace. The experiments demonstrate the effectiveness, scalability and resource sharing fairness of the proposed model.

Index Terms—federated cloud; geo-distributed cloud; resource sharing; resource sharing contracts

1 Introduction

In the era of Big Data, with data growing massively in scale and velocity [1], cloud computing and its pay-as-you-go model continues to provide significant cost benefits and a seamless service delivery model for cloud consumers. The recent explosion of large-scale data and the impact that big data analytics brings to enterprises and enterprise customers create an ever-increasing trend for adopting cloud technologies and moving applications to the cloud [2]–[4]. The evolution of small-scale and large-scale geo-distributed datacenters operated and managed by individual Cloud Service Providers (CSPs) raises new challenges in terms of effective global resource sharing and management of autonomously controlled individual datacenter resources towards a globally efficient resource allocation model. Individual datacenters have capacity limitations in terms of available server capacity and dynamically varying electricity prices that determine the cost and profitability of the datacenters at various electricity pricing and workload conditions. Cloud federation [5]–[7] aims at extending the capacity of the datacenter resources by leveraging resources in remote datacenters that are under utilized or available at a reduced cost. Cloud federation is intended to handle burstiness in workloads, fluctuations in electricity price and responding to emergency datacenter failures for high availability applications. It also provides an opportunity for datacenters to share resources to maximize revenue by leveraging remote datacenter resources that may be available at a lower cost due to dynamic electricity pricing.

- *J.Xu and B.Palanisamy are with the School of Information Sciences, University of Pittsburgh, Pittsburgh, PA 15213. Email: {jinlai.xu, bpalan}@pitt.edu*

Manuscript received XX XX, XXXX; revised XX XX, XXXX.

Earlier solutions for geo-distributed clouds have focused primarily on achieving global efficiency in resource sharing. Several mechanisms were designed to achieve higher utility and overall global profit [8] [9]. Most geo-distributed resource allocation in the past [10]–[16] have considered a completely co-operative model of a shared pool of geo-distributed resources that are allocated to optimize the global resource usage cost. Such schemes although try to optimize the global resource usage, they result in significant inefficiencies in local resource allocation for individual datacenters leading to unfairness in their profit earned. Furthermore, solutions that are based on a completely co-operative geo-distributed cloud model are not easily adaptable when consumers have constraints of geographic locations, business policies or security regulations that limit the datacenters in which their services/jobs can be deployed.

In this paper, we propose a new contracts-based resource sharing model for federated geo-distributed clouds that allows CSPs to establish resource sharing contracts with individual datacenters *apriori* for defined time intervals during a 24 hour time period. Based on the established contracts, individual CSPs employ a contracts cost and duration aware job scheduling and provisioning algorithm that enables jobs to complete and meet their response time requirements while achieving both global resource allocation efficiency and local fairness in the profit earned. Concretely, this paper makes the following contributions: first, we develop the proposed contracts-based resource sharing model and present an optimal contract establishment algorithm that produces the optimal design of resource sharing contracts considering the size and type of resources in each resource sharing contract. Second, we develop an auction-based contract allocation mechanism that ensures both fairness and rev-

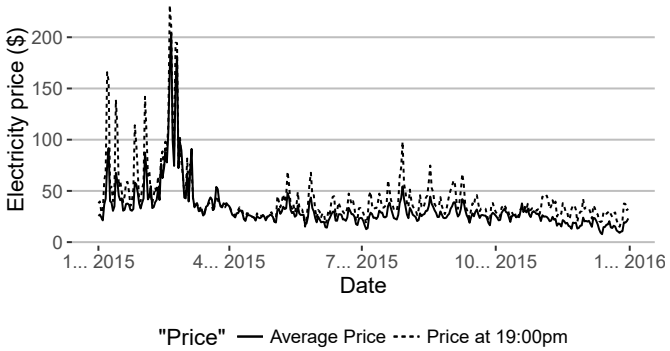


Fig. 1. Electricity price trends of NationalGrid in 2015

enue maximization for the individual datacenter providers. Third, we develop a suite of job scheduling and contracts-based resource provisioning algorithms that leverage the established contracts for each CSP and minimizes the resource usage cost of individual CSPs. We evaluate the proposed techniques through extensive experiments using realistic workloads generated using the SHARCNET cluster trace. The experiments demonstrate the effectiveness, scalability and resource sharing fairness of the proposed model.

The remainder of this paper is organized as follows. Section 2 provides a background of various resource sharing models for geo-distributed clouds and motivates the proposed contracts-based model. In Section 3, we develop the proposed contracts-based resource allocation system model. In Section 4, we present new techniques for optimal contracts designing and allocation. In Section 5, we present our proposed contracts-based job scheduling techniques. Section 6 evaluates the performance of the contracts-based resource allocation mechanisms in comparison with conventional complete cooperation geo-distributed clouds using real-world datacenter workload traces. Section 7 discusses the related work and we conclude in Section 8.

2 Background & Motivation

In this section, we briefly review the background concepts related to various models of operating a geo-distributed cloud and discuss their merits and demerits.

2.1 Stand-alone Clouds

Conventional cloud computing models (e.g., Amazon EC2 [17] and Google Cloud [18]) use a single datacenter or a set of datacenters jointly managed by a single CSP. Thus, the CSPs do not cooperate with each other and do not aim at optimizing resource allocation and cost across multiple CSPs (Figure 2a). Despite resulting in sub-optimal resource allocation and resource management, this centralized single-site resource management model has the benefit of easier resource management as each datacenter is managed independently of each other, providing higher autonomy and control for individual datacenters. Even though this "stand-alone" datacenter management may result in locally optimized resource management at individual datacenters, such an approach can be largely sub-optimal with respect to global resource management considering all datacenter resources jointly in a federated geo-distributed cloud environment. As an example, Figure 1 shows the dynamic electricity pricing from the NationalGrid [19] data in 2015. We observe that besides the notable long-term (e.g., one year) fluctuations, there are significant short-term price variations even on a single day: the highest per-day pricing on one given day can be as much as six times the lowest price observed on the same day. Thus, "stand-alone" clouds that have neither complete nor partial co-operation with each other can operate very sub-optimally forcing individual datacenters to run workloads locally at higher electricity prices even though resources for which may be available at remote datacenters at a possibly lower electricity cost.

2.2 Federated Clouds with Complete Cooperation

In the literature, several techniques for global management geo-distributed datacenters have been proposed. These mechanisms can be classified into two broad categories:

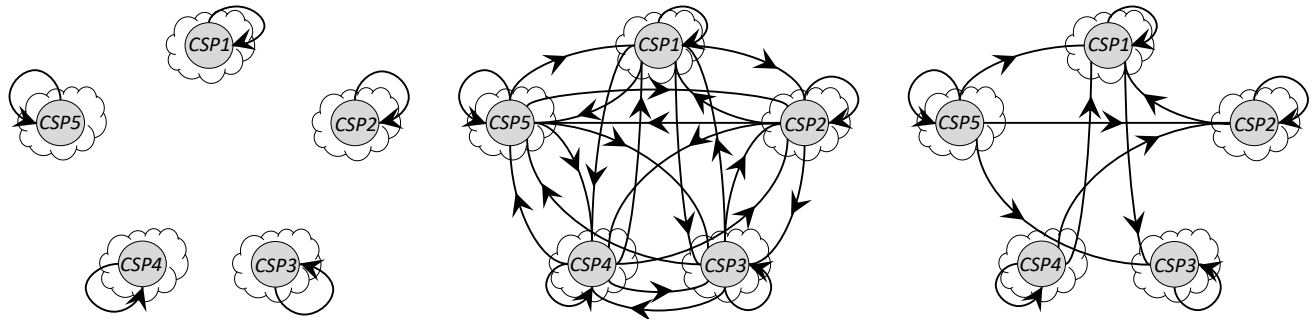
Virtual Geo-distributed Clusters: This class of techniques builds Virtual Machines (VMs) for users to use computing resources across the geo-distributed datacenters as a single virtual cluster. There are several works focusing on optimizing the performance in the geo-distributed environment [10]–[16]. Here, the datacenters are treated as one single virtual entity and having a single centralized cloud manager makes it easier to schedule the jobs and place data to achieve the overall goal. The cloud manager obtains the global information of the jobs and the individual workload requirements of each datacenter to balance the load and schedule the jobs.

Federated Cloud: Federated clouds provide a platform for the CSPs to share computing resources with each other. Each CSP is assumed to manage its datacenters autonomously. There is a centralized Cloud Exchange Institution that obtains all infrastructure information from the datacenters and provides the platform for the CSPs to discover the resources from the members of the federated cloud [5] [7]. The key objective for the CSPs is to share their resources on the federated cloud platform to maximize their resource utilization and increase the success rate of meeting the SLAs for the jobs.

We illustrate these two types of global resource management mechanisms in Figure 2b and we refer to this model as federated clouds with complete cooperation. This model enables the free use of the resources through a centralized broker such that all the resources in the geo-distributed datacenters can be used by all the other members participating in the system. However, this model suffers from a few key drawbacks, which include (i) lack of fairness in revenue earned by competing CSPs, i.e., since the global resource optimization objective of this approach does not lead to locally optimized profits for individual datacenters, the individual profit of each datacenter may be even lower than the profits they can get by operating stand-alone and (ii) limited scalability - as it is difficult for all the geo-distributed datacenters to globally synchronize the information necessary for sharing, provisioning and allocating resources in a real-time manner for job scheduling can be a significant challenge.

2.3 Contracts-based Resource Sharing

In this paper, we propose a new contracts-based resource sharing architecture for the CSPs to share resources across globally geo-distributed datacenters. The demerits of the complete cooperation model lead us to a more flexible and



(a) No resource sharing between the CSPs (b) Completely cooperating resource sharing (c) Contracts-based resource sharing

Fig. 2. Resource sharing mechanisms comparison

limited sharing mechanism that provides a controlled cost-aware resource sharing opportunity. Thus, the contracts-based resource sharing model finds suitable tradeoffs between traditional clouds without federation and that with complete cooperation as illustrated in Figure 2. The figure shows three different architectures for a geo-distributed cloud of five CSPs. Here, the edges represent the usage of resources among the CSPs. As illustrated in Figure 2a, none of the five CSPs can use others' resources when there is no federation. However, in Figure 2b, we find that there is a complete graph showing that every CSP can use every other's resources with complete co-operation. In Figure 2c, there are only six edges between the CSPs representing a partial graph. Here, each CSP does not share resources with every other CSP in the federation. Each edge represents a contract between the CSPs to share resources.

The proposed contracts-based resource sharing mechanism is based on resource sharing contracts that are established between the CSPs after negotiations. The resource sharing contracts could be signed by the CSPs stipulating the rights and duties of the CSPs to share the committed resources during the time duration and the negotiated price in the contract. For the contracts-based resource sharing mechanism, the CSPs design and trade the resource sharing contracts with each other. Thus, the contract may be predetermined and established apriori before the effective time. The establishment of contracts involves two key challenges namely (i) how to design and build contracts that can maximize individual profit of the CSP and (ii) how to schedule jobs to maximize the utility of using the contracts.

TABLE 1
The status of the five providers in the contracts-based example

	CSP1	CSP2	CSP3	CSP4	CSP5
Electricity Price	100	30	40	30	30
# of Servers	30,000	40,000	30,000	30,000	50,000
Require # of Servers	15,000	52,000	15,000	21,000	41,000

Figure 3 and Table 1 present an example scenario to illustrate the key benefits of using a contracts-based resource sharing mechanism namely (i) balancing the workload, (ii) minimizing the operating cost and (iii) increased resource utilization:

Balancing the workload: In the example shown in Figure 3a, CSP2 has overcapacity workload and needs 52K servers to meet the workload requirements. However, it has only 40K servers. Therefore, under normal operations, it has to either delay some jobs in the workload or drop them entirely. Alternately, in the contracts-based federated cloud model, CSP2 borrows 9K servers from CSP5 and 3K servers from CSP3 to meet workload requirements of 52K servers. As we can see, this not only increases the revenue for CSP2 but also for the other CSPs participating in the contracts-based resource sharing.

Minimizing operating cost: In Figure 3b, CSP1 experiences an increased electricity cost requiring to spend $100 per megawatt per hour. Even though it has the similar workload amount as CSP4 and CSP3, it uses contractual relationships to borrow 9K servers from CSP4 and 6K servers from CSP3 respectively. This minimizes the operating cost and saves up to 66% in operating cost for CSP1.

Increased resource utilization: Figure 3 also illustrates that some CSPs that have idle resources share their resources with other CSPs (e.g., CSP 3,4,5). Thus, contracts-based resource sharing results in an increased utilization of the computing resources in the datacenter infrastructures.

As discussed above, we find that contracts-based resource sharing has additional potential and flexibility to achieve a more efficient resource allocation while increasing the profit and minimizing the cost for each individual datacenter. In this paper, we model the problem formally, analyze and develop algorithms for contract establishment and job scheduling to efficiently and profitably share resources between CSPs.

3 System Model

In this section, we describe the system model for the proposed contracts-based federated geo-distributed cloud model. We discuss it in three steps: first, we describe the features of the CSPs that participate in the cloud federation process. We then discuss the agreements and the responsibilities of the federation coordinator and finally, we discuss the role of the contract manager that manages the resource sharing contracts agreed between the CSPs.

3.1 Cloud Service Provider

CSPs offer a variety of cloud computing services to the customers. We primarily consider CSPs offering Infrastructure as a Service (IaaS) [20] that provide customers with various computing resources such as VMs and virtual disk space to store and process their data. The providers may offer different types of VMs with different Quality of Service (QoS) guarantees and the VMs may be priced differently. The QoS provided by the VMs may depend on how many CPU cores are present in the VMs, memory, network and

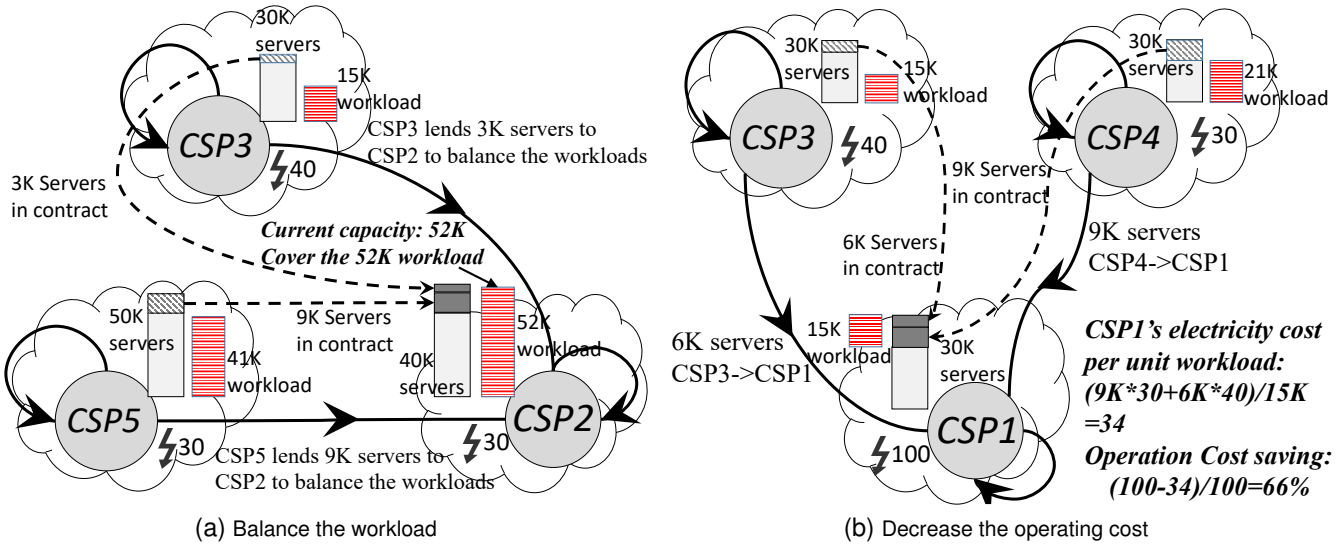


Fig. 3. Contracts-based cloud federation example of saving electricity cost and balance the workload

other resources that are guaranteed in the period of time when the resources are provided to the user. The price is set by the CSP which provides the service based on the QoS provided by the VM type, the Service Level Agreement (SLA) and the market demand and supply.

The provider charges the customers *on-demand* based on the length of the running time and the price of the VM type. The profit of the CSP is determined by the charges provided by the customers, the operating cost and the penalty for violating the SLA. The operating cost which varies with the time includes the electricity cost, management cost and cost for maintenance. The penalty is paid by the CSP to the customers to compensate their loss in case of violating the SLA. For example, in Amazon Elastic Compute Cloud(Amazon EC2) [17], the SLA stipulates that if the monthly uptime of the service is less than 99.95% and greater than 99.0%, Amazon EC2 will pay 10% of the charge of using the service back to the users' account and 30% if the uptime is fallen to less than 99.0% [17].

Every CSP has limited resources to serve the users.To handle the overcapacity workload that cannot be serviced within the CSP's own datacenter, the CSPs can engage in a federation process to share idle resources and handle overcapacity requests. The negotiating steps are done by a trusted third party which we refer to as the Federation Coordinator.

3.2 Federation Coordinator

The federation coordinator is a third-party service which is trusted by the CSPs in the federated cloud and it is responsible for providing a platform for the CSPs to trade computing resources with each other. There is an agreement signed with the coordinator before the CSP joins the federation. The agreement stipulates the rights and duties of the coordinator and the CSP. The coordinator follows the optimized contract establishment process proposed and discussed in Section 4 to establish the contracts between the CSPs.

When building the contracts, each CSP sends its demand and supply to the coordinator to compare with the demands and supplies from others. The demand and supply information may have private information of the CSPs and hence the agreement also stipulates the privacy policy for the coordinator which determines to which degree the coordinator can publish or share the information submitted by the CSPs. All the CSPs are autonomous and have their own customers. Each CSP has its own utility (which can be estimated approximately by the profit) and each CSP wants to increase the utility after participating in the federation. Under this condition, the problem contains both the cooperating and competing aspects with multiple participants which cannot be solved by the methods that assume that all resource allocation decisions are handled with a central objective of global resource optimization. So in this scenario, auction mechanisms that are widely studied in Game Theory are most suitable. Auctions allow the participants to both cooperate and compete [21]. The essence of the auction is to match the supply and demand at both sides which fit the characters of the problem intuitively. The coordinator uses an auction-based mechanism to match the demands and supplies of the CSPs. The auction ends with a set of results which contains the winning decisions and the market clearance prices. Based on the results, the coordinator establishes the contracts.

3.3 Contract Manager

The contract manager manages the resource sharing contracts agreed by the CSPs. The contract is an agreement between the CSPs which stipulates the rights and duties of both sides, the buyer and the seller in the contract.

An actual resource sharing contract contains the following four information: (i) the buyer and the seller of the resource in the contract, (ii) effective time of the contract which controls the starting and ending time of the contract, (iii) the resource type and quantity in the contract and (iv) agreed resource price. In our work, we use dedicated resources as the unit of trading in the contracts. A dedicated resource is a collection of servers which is hardware isolated from the other resources in the datacenter. Examples of such resources include IBM Bluemix Dedicated Cloud [22] and Amazon EC2 Dedicated Instances [23]. The pricing model

used for the contracts consists of two components namely (i) the reservation price that is paid when the two sides establish the contract and (ii) the usage price which is paid when the resources in the contract are actually used. The usage price is paid according to the usage amount and time of the resource. The contract can also include the SLA which provides more guarantees for the performance of the resource included in the contract. Other constraints that can be included in the contract include the location constraints of placing the jobs, the business policies and security requirements.

In the next section, we present the proposed mechanisms for establishing resource sharing contracts.

4 Resource Sharing Contracts Establishment

We design an auction-based mechanism for establishing resource sharing contracts as the nature of the contract establishment problems naturally fits the auction mechanism. Some key features such as *truthfulness* and *budget balance* of the auctioning protocol are highly desirable and essential for solving the contract establishment problem. *Truthfulness* or "strategy-proofness" is a feature provided by many auction mechanisms such as VCG mechanism [24] and McAfee mechanism [25] and it ensures that the participants of the auction can maximize its utility only by bidding with the true value which he/she values the goods in the auction. This feature simplifies the problem by narrowing down the choices of the participants. The *budget balance* feature guarantees that the payment from the buyers is equal to or more than the payment to the sellers. This feature guarantees that the coordinator will not subsidize in the auction.

As discussed above, we choose to design the proposed auction mechanism based on McAfee mechanism as it inherently guarantees both *truthfulness* and *budget-balance*. In the McAfee mechanism, the selling price and the buying price are determined separately which helps to keep the auction truthful and budget-balanced. Even though having separate selling and buying prices makes the trade efficiency sub-optimal, it is necessary to design a truthful auction mechanism. As the uniqueness-of-prices theorem [26] implies, this subsidy problem (the auctioneer need to subsidize the auction) is inevitable - any truthful mechanism that optimizes the social welfare will have the same prices (up to a function independent of the bid prices of each bidder). If we want to keep the mechanism truthful while not having to subsidize the trade, we must compromise on efficiency and implement a less-than-optimal social welfare function [27]. The McAfee mechanism is designed based on the above theorem which has a bounded trade efficiency loss, $1/\min(|B|,|S|)$ where B is the set of buy bids and S is the set of sell bids, but maintains both *truthfulness* and *budget-balance*.

We design the framework of the contracts establishment process in three parts: first we model the problem of establishing the resource sharing contracts into an auction; second we develop the strategy for the CSP to bid in the auction; third we design the auction algorithm which determines winning bids and the market clearance prices. Finally, we design the iterative process of building the contracts, which is based on the proposed auction algorithm.

4.1 Problem Description

We first model the contracts establishment problem as a sealed-bid double auction problem. In a sealed-bid double auction [24], there are three kinds of participants: first are the buyers who have the demands for the goods; second are the sellers that can supply the goods; the third is the auctioneer which is responsible for conducting the auction. In the contracts establishment problem, the CSPs can act as both buyers and sellers based on their demands and supplies. The coordinator of the federated cloud, a trusted third party acts as the auctioneer. The traded goods in the auction are the rights to use a certain amount of cloud resource in a certain period of time (time slot). We use dedicated resource types [23] [22] to represent a cloud resource. The dedicated resources can be considered as bundles of servers isolated from other resources in the data center. It is defined by $k \in \{1, 2, ..., K\}$. Each type-k resource may contain several servers which can be represented by a list D_k and each server $d \in D_k$ has a capacity V_k^d and the overall capacity of a type-k resource is $V_k = \sum_{d \in D_k} V_k^d$. We note that the resource types are sorted by the resource capacity which means that if $k_1 > k_2$, $V_{k_1} > V_{k_2}$.

We consider a federated cloud with N individual CSPs. The contracts establishing problem is formulated using discrete time slots τ. The cloud datacenters are located in geo-distributed locations and each of them is controlled by one CSP. We assume each CSP $i \in [1, N]$ has only one datacenter for simplicity. We assume that each datacenter has several types of servers. It has a server list $M_i(\tau)$ which contains all the servers controlled by the CSP i in time slot τ. The server list can be modified in each time slot τ by adding or removing the servers which are controlled by the cloud manager of the CSP. These operations simplify the representation of the resource which is changed every time slot with different contracts signed in each time slot. The capacity of each server $m \in M_i(\tau)$ is C_i^m. Therefore, the capacity of CSP i in time slot τ can be represented by $C_i(\tau) = \sum_{m \in M_i(\tau)} C_i^m$. Each CSP serves its customers by providing resources for running their computationally-intensive jobs. The job requests are sent to the CSP, which are pushed into a job queue. The jobs in the queue are processed according to a FCFS (First Come First Served) service policy. We assume that the demand for each time slot τ is $\lambda_i(\tau)$ for CSP i which can be determined by predicting the upcoming workloads through mechanisms such as ARIMA [28] or Hidden Markov Modeling (HMM) [29]. The profit earned by the CSPs is determined by the difference between payments from the users and the operating cost and the penalty. The payments are related to the demand $\lambda_i(\tau)$ and we use ϱ_i to denote the unit price for one unit resource (for example, one VM with one EC2 Compute Unit (ECU) and one-gigabyte memory in Amazon EC2 [30] can be a unit resource). Therefore, the capacity for each server C_i^m and V_k^d can also represent the number of unit resources that can be run on the server. We use $Cost_i^k(\tau)$ to denote the operating cost of CSP i for the type-k dedicated resource.

The problem of establishing the contracts in each time slot τ for each type-k dedicated resource can be represented as a double auction in which each CSP decides the sell bid (ask prices), $s_i^k(\tau)$, for each type-k dedicated resource in

time slot τ and buy bid, $b_i^k(\tau)$, based on the valuation of the resources and the expected utility. For simplicity, in the rest of the paper, we use the term sell bid to indicate the minimal price that the sellers expect in order to sell their resources. The coordinator base on the bids to decide the pairs of winning bids. Here $X_b(\tau) = \{x_{b_1}(\tau), x_{b_2}(\tau), ...\}^k$ (if $x_{b_i} = 1$ means b_i wins and vice versa) denotes the buy bids result and $X_s(\tau) = \{x_{s_1}(\tau), x_{s_2}(\tau), ...\}^k$ denotes the sell bids result. The resource sharing contracts establishment problem is solved by the auction. The resource sharing contract represents the following information: (i) the effective time of the contract determined by τ, (ii) the two sides of the contract determined by matching the bidders of the winning bids. We use index i to denote the seller's index and j to denote the buyer's index, (iii) the selling and buying price of the contract represented by $\pi_s^k(\tau)$ and $\pi_b^k(\tau)$, (iv) the resource type and quantity represented by k and D_k. The contract is denoted by $Contr_{ij}^k(\tau) =< \pi_s^k(\tau), \pi_b^k(\tau), D_k >$. The optimal solution for establishing the contracts maximize every CSP's utility after attending the auction. We first propose the suggested bidding strategy and we discuss the utility function of the CSPs participating in the auction.

4.2 Proposed Bidding Strategy

As discussed previously, in our model, each provider participates in the federated cloud to potentially increase its profit. We design the bidding strategy for the CSPs to ensure that the CSPs increase their profits through their participation in the federated cloud. Before we design the bidding strategy for the provider, we first discuss the utility function of the providers. The utility of participating in the auction is defined based on the profit a CSP can gain from running the jobs on the resources in the contracts and the profit it can gain from selling its local resources to other CSPs in the contract.

First, we define the utility function using the profit a provider i can get from renting type-k dedicated resources from others:

$$u_i^k(\tau) = \varrho_i \max\{\min\{Res(\lambda_i(\tau)) - C_i(\tau), V_k\}, 0\} - \pi_b^k(\tau) \tag{1}$$

where $Res()$ is a function that calculates the resource from the service demand.

Then, if the demand is under capacity but the operating cost is higher, the CSP can also participate in the auction to increase the utility from running jobs on other CSPs' resources. In this condition, the utility function is:

$$u_i^k(\tau) = Cost_i^k(\tau) - \pi_b^k(\tau) \tag{2}$$

There is only one condition in which the CSP wants to sell their resource to others: the demand is notably less than the capacity of the available resources. In this condition, the utility function of the CSP which wants to provide type-k dedicated resources to others can be represented by:

$$u_i^k(\tau) = \pi_s^k(\tau) - Cost_i^k(\tau) \tag{3}$$

When participating in the auction, provider i needs to consider the utility it can gain from the auction. For the potential seller who has idle resources, it wants to increase its profit by increasing the utilization of the idle resources. For the potential buyer, it also wants to increase its profit by either serving the demand which is over the capacity of the local resource or decreasing the operating cost by outsourcing the jobs to the other lower cost resource in the contracts.

From the above discussion, we can get the bidding strategy for the CSPs. For the potential seller, it only needs to estimate the usage of the resource and match the idle resource into one type of dedicated resource and set the bid price by the operating cost. For the potential buyer, it has a mixed strategy: if the predicted service demand is over the capacity of the current servers in the server list, it bids by the profit it can get from serving the overcapacity demand; otherwise, it bids by the operating cost instead.

From the above bidding strategies' discussion, provider i that has idle servers matching type-k dedicated resources can set the selling price by the operating cost:

$$s_i^k(\tau) = \begin{cases} Cost_i^k(\tau) \text{ if } Res(\lambda_i(\tau)) > C_i(\tau) - V_k \\ \qquad\qquad \text{and } D_k \subset M_i(\tau) \\ \text{NULL otherwise} \end{cases} \tag{4}$$

where "NULL" represents a null bid. Here, we note that the condition, $D_k \subset M_i(\tau)$, checks whether the available server list, $M_i(\tau)$, contains the type-k resource, D_k, or not.

The buyer who wants to buy type-k dedicated resource will bid in two conditions: first, the predicted demand is above the current capacity; second, the operating cost is relatively high in the time slot τ. So the bidding strategy is a combination of two separate strategies:

$$b_i^k(\tau) = \begin{cases} \varrho_i \min\{Res(\lambda_i(\tau)) - C_i(\tau), V_k\} \\ \qquad\qquad \text{if } Res(\lambda_i(\tau)) > C_i(\tau) \\ Cost_i^k(\tau) \text{ otherwise} \end{cases} \tag{5}$$

It is worth noting that as shown in the above strategies, when there are idle resources, the CSP will set the buy bid with its operating cost regardless of whether it needs the resources or not. We can understand this condition from two aspects: if the CSP does not bid when it has idle resources, it will always get zero utility in this round of auction; instead, if the CSP bids with the operating cost, it will at least get zero utility, which will become the dominant strategy for the CSP.

While above suggested bidding strategy provides a basic methodology to estimate the benefits CSPs can obtain from participating in the auction, we note that the bidding strategy can be extended with additional requirements for CSPs (e.g., reliability requirements, scheduling policy constraints, data locality constraints, etc.,) to further customize the bidding process.

4.3 Winning Bids Decision

In this subsection, we present the proposed algorithm for determining the winning bids. As mentioned earlier, the proposed auction algorithm (Algorithm 1.) guarantees both truthfulness and budget balance properties.

The winner decision algorithm for the auction is based on the McAfee mechanism which both guarantees truthfulness and budget balance. The decision of the auction is indicated by two sets of indicators, $X_b(\tau)$ and $X_s(\tau)$. The buying bid b_h's indicator is set to be $x_{s_h} = 1$ if bid b_h wins. For the selling bid, it is the same as for the buying bid. We note that the time complexity of Algorithm 1 is $O(N \log N)$. Here the key time-consuming operation is the initial sorting operation.

Algorithm 1: Algorithm for the double auction to choose winners for one type of dedicated resource

Input : Type of the dedicated resource : k;
Buy bids: $B^k(\tau) = \{b_1(\tau), b_2(\tau), ...\}^k$;
Sell bids: $S^k(\tau) = \{s_1(\tau), s_2(\tau), ...\}^k$;
Output: Clearing Buying Price: $\pi_s^k(\tau)$ Clearing Selling Price: $\pi_b^k(\tau)$;
Auction decision: $X_b(\tau) = \{x_{b_1}(\tau), x_{b_2}(\tau), ...\}^k$;
$X_s(\tau) = \{x_{s_1}(\tau), x_{s_2}(\tau), ...\}^k$
Sort $B(\tau)$ in descending order by $b_i(\tau)$ and $S(\tau)$ in ascending order by $s_i(\tau)$;
Initially, set current buying price b as b_k as the first bid (highest price) in $B_k(t)$ and current selling price s as s_k as the first bid (lowest price) in $S_k(t)$. current bid indicator $h = 0$;
while $b \geq s$ **do**
    $s = s_i(\tau)$;
    $b = b_i(\tau)$;
    $h = h + 1$;
    if h is larger than the size of $B^k(\tau)$ or $S^k(\tau)$ break;
end
$\rho = (b_{h+1} + s_{h+1})/2$;
if $b_h \geq \rho \geq s_h$ **then**
    All the first h buyers and first h sellers win with price:;
    $\pi_s^k(\tau) = \pi_b^k(\tau) = \rho$;
end
else
    All the first $h-1$ sellers win with selling price: $\pi_s^k(\tau) = s_h$;
    All the first $h-1$ buyers win with buying price: $\pi_b^k(\tau) = b_h$;
end

4.4 Contracts Establishment Process

In this subsection, we discuss the overall process for building the contracts based on the auction algorithm. We illustrate it as a sequence of procedural steps:

Step.1 Begin the auction from $k = K$ that represents the CSP with the largest amount of resources. The auction begins at time slot $\tau = 1$.

Step.2 CSPs send the bids to the coordinator of the federation using the strategy described above.

Step.3 The coordinator decides the winners by the algorithm as shown in Algorithm 1.

Step.4 The winning bids build the contracts one to one in the order of the sell and buy bids. Each winning buyer adds the servers in the dedicated resource into the server list $M_i(\tau)$ in time slot τ. Each winning seller also updates the server list $M_i(\tau)$ by removing the servers from the list.

Step.5 The losing bids of the type-k dedicated resource are sent back to the CSP. The CSP will separate it into several bids which may be used in the auctions for the smaller types of dedicated resource. These bids also obey the strategies defined in Subsection 4.2.

Step.6 For the next smaller type-$\{k-1\}$ dedicated resource, the CSPs execute the above steps from Step.2 until the smallest type dedicated resource is reached.

Step.7 The CSPs execute the above steps from Step.1 for the next time slot $\tau + 1$ until the last contract time slot $\tau = T$ is reached;

We note that the optimized contract provides the CSPs with a set of available remote resources in a cost-efficient manner. However, the individual CSP needs to employ intelligent job scheduling techniques that understand the cost implications of the underlying contract structure to leverage the remote resources available to the CSPs effectively. We discuss them in the next section.

5 Contracts-based Job Scheduling

In this section, we first formulate the contracts-based job scheduling problem and then propose our mechanisms to schedule jobs using the extended resources provided to the CSPs through the resource sharing contracts.

5.1 Job Scheduling Problem Model

We model the scheduling problem for each CSP that participates in the federated cloud. We note that in the scheduling model, the time slot indicated by t can be a very short time interval. It can be several orders of magnitude shorter than the time slot for establishing the contracts which is represented by τ. We introduce two new terms, t_τ^{begin} and t_τ^{end}, to indicate the beginning and end of the interval of the contract time slot τ during job scheduling. All the jobs come to a CSP enter into an FIFO queue. The arriving rate in time slot t is denoted as $r_i(t)$ for provider i. The queue is updated every time slot:

$$Q_i(t+1) = \max\{Q_i(t) - U_i(t) + r_i(t) - A_i(t), 0\} \quad (6)$$

where $U_i(t)$ is the set of scheduled jobs in time slot t, $A_i(t)$ is the set of the jobs which are dropped because of violating the SLA or other failures in time slot t. The queue is only updated at the beginning of the time slot.

In addition, there is a constraint that the amount of the used resources of the running jobs cannot exceed the current capacity of the CSPs. We use the notations below to represent the running jobs.

$$Z_i(t+1) = Z_i(t) + U_i(t) - F_i(t) \quad (7)$$

where $Z_i(t)$ is the running jobs' set at time slot t, $U_i(t)$ is the scheduled jobs in time slot t, $F_i(t)$ is the finished or failed jobs in the running jobs set in time slot t. Here the capacity constraint is:

$$Res(Z_i(t)) \leq \sum_m^{M_i(\tau)} C_i^m, \ \forall t \in [t_\tau^{begin}, t_\tau^{end}] \quad (8)$$

where $Res(Z_i(t))$ represents the estimated resources that all the running jobs need for CSP i in time slot t. The resource list $M_i(\tau)$ is updated at the beginning of every contract time slot τ.

The actual benefits a CSP i can get from the contract is obtained as:

$$\begin{aligned} Contract_i(t) =& (\sum_k^K \pi_s^k(\tau) x_{s_i}^k(\tau) - \sum_k^K \pi_b^k(\tau) x_{b_i}^k(\tau)) \\ & */(t_\tau^{end} - t_\tau^{begin} + 1), \ \forall t \in [t_\tau^{begin}, t_\tau^{end}] \end{aligned} \quad (9)$$

where $\pi_s^k(t)$ is the clearing selling price for the contracts in time slot $t \in [t_\tau^{begin}, t_\tau^{end}]$ for the type-k dedicated resource.

The actual cost of the electricity consumption is calculated by the consumption of each server in time slot t. For each server, the electricity consumption can be approximated by a linear model [31] as illustrated in Table 2. For each server $m \in [1, M_i]$, the electricity consumption in time slot t is calculated as:

$$E_i^m(t) = \begin{cases} \xi_i^m + \frac{\psi_i^m * u_i^m(t)}{C_i^m} \text{ if server } m \text{ is on} \\ 0 \text{ otherwise} \end{cases} \quad (10)$$

where ξ_i^m and ψ_i^m are the parameters in the linear model for estimating the electricity cost of the server, u_i^m is the utilization of server m in time slot t. Therefore the actual electricity cost for CSP i can be calculated as:

$$Electricity_i(t) = PUE_i * \sum_m^{M_i} E_i^m(t) * \delta_i(t) \quad (11)$$

TABLE 2
IBM server x3550 Xeon X5675 power consumption with different workload

Workload	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Power Consumption(Watts)	58.4	98	109	118	128	140	153	170	189	205	222

where PUE_i is the Power Usage Effectiveness (PUE) of CSP i, $\delta_i(t)$ is the electricity price for the time slot t for CSP i.

We note that the profit is approximately proportional to the resource consumption of running jobs. Therefore this is another objective for the provider to maximize:

$$Income_i(t) = \varrho_i * Res(Z_i(t)) \tag{12}$$

Finally, we note that there is a penalty of violating the SLA of the jobs. The penalty can be estimated as:

$$Penalty_i(t) = \sum_{p \in Q_i(t)} \vartheta_i * \varrho_i * \gamma_p \text{ if violates } p\text{'s SLA} \tag{13}$$

where ϑ_i is the parameter of the penalty which is determined in the SLA for violating the SLA, γ_p is the resource usage of job p.

Considering all the objectives and constraints together, we can obtain the objective for the CSP i as:

$$\begin{aligned} \max \lim_{T\to\infty} \sum_{t=0}^{T-1} & Income_i(t) - Electricity_i(t) - Penalty_i(t) \\ & + Contract_i(t), \\ & \forall i \in [1, N] \\ & s.t.\ Constrains\ (6) - (8) \end{aligned} \tag{14}$$

We note that the above-mentioned scheduling problem can be reduced to a bin packing problem. However, bin packing problems are shown to be NP-hard [32]. Thus, we need to resort to heuristic techniques to achieve a scalable solution. We describe them in the next subsection.

5.2 Contracts-based Job Scheduling Mechanisms

We propose a set of heuristic scheduling mechanisms to schedule jobs across the geo-distributed clouds based on the resource sharing contracts. We develop two mechanisms to optimize the scheduling decision based on the contracts: one is to use the contracts in a cost-aware manner; another is to schedule the job with minimal live migrations by understanding the duration of the contracts.

The objective of the CSPs in the job scheduling technique is to schedule the jobs to maximize the utility of using the contracts. The utility of using the contracts consists of two parts namely the payment for successfully completing the jobs and the cost of using the contracts. As optimizing the number of completed jobs is an NP-hard problem, we use the basic real-time FCFS service policy and first-fit scheduling algorithm as the preliminary approach. We then extend the basic scheduling algorithm to optimize the cost of using the contracts. We note that the cost of using the contracts can be separated into two components namely (i) the cost to use the contracts, which is decided by the price and (ii) the additional cost which occurs when using the contracts such as the job migrating cost and the penalty of violating the contracts.

Concretely, we propose three schemes. While the contracts cost-aware scheduling (ConBCA) adopts a lowest cost resource (resources in the contract or local datacenter) first policy to minimize the cost of using the contracts, the contracts duration-aware scheduling (ConBConA) considers the duration of the contracts in order to minimize the number and cost of live migrations. The contracts duration-aware and cost-aware scheduling (ConBCAConA) simultaneously aims to minimize both the cost of using the contracts as well as the number of live migrations based on contracts duration. z e space-0.1in

5.2.1 Contracts cost-aware scheduling

In the contracts cost-aware scheduling approach, the CSPs cooperate with each other based on the established contracts to maximize their profit. In an intuitive scheduling policy, the providers may choose only to use the contracts when the local resource is not sufficient to meet the workload demands, which we refer to as contracts-based local first (ConBLF) scheduling mechanism. The disadvantage of this mechanism is that if the local operating cost is higher than some of the negotiated price in the contracts, the contracts become poorly utilized. An alternate intuitive scheduling policy may approach the scheduling problem in the opposite manner which is the contracts first contracts-based scheduling mechanism. This scheme also has shortcomings, the cost of the contracts may be higher than the local resource in some scenarios, in which cases, the contracts-based remote resource may only be used when the local resources are exhausted.

The above discussion provides the intuition behind the contracts-based cost-aware (ConBCA) scheduling algorithm. The provider can estimate the unit cost compared with the local unit operating cost. It may use the contract which has a lower cost than the local operating cost first. It may then use the local resources. The contracts which have a higher cost than the local operating cost are used only when the previous two kinds of resources are exhausted. The detailed algorithm is illustrated in Algorithm 2.

5.2.2 Contracts duration-aware scheduling

For maximizing the utility of using the contracts, another aspect to consider is how to increase the utilization of the contracts and avoid violating the contracts. An example of contract violation includes using the resources in the contracts beyond the effective time. When the contract is near the end of effective time, the jobs which are not already finished should be moved to other places that have resources to continue the jobs. Therefore, we have another cost which occurs in this condition. This cost represents the cost of migrating the jobs when the contract ends but the jobs are not finished already. Furthermore, for providing a more continuous service, most of the jobs should be moved using a seamless live migration method [33] that incurs minimal or no impact on the performance of the job. Live migration is a way of migrating the VMs which minimize the down-time of the VM. This mechanism iteratively copies the dirty memory to the remote server and moves the job to the server in a short down-time. However, the migrating operation is expensive and may consume network bandwidth between the two geo-distributed datacenters and the CPU resources on both servers. Therefore, the provider should avoid the

migrating process by carefully understanding the duration of the contracts. We call this scheme as the contracts-based contracts duration-aware mechanism (ConBConA).

The provider needs to minimize the probability of live migrating the jobs when the contract is near expiration time. If the expected finish time of the job is beyond the expiration time of the contract, the job should not be scheduled to the datacenter unless the benefit of running outside is more than the migration cost.

We model the probability of migrating the job as follows. The remaining time from the expected finish time of each job p in time slot t can be calculated as:

$$t_p^{remain} = t_\tau^{end} - (t + l_p) \tag{15}$$

Where t_τ^{end} represents the end time of the contract, t is the starting time of job p which is indicated by the current time slot t, l_p is the expected length of job p.

We assume that the probability of migrating is inversely proportional to the remaining time from the expected end time to the expiration time of current effective contract. The possibility of migrating can be estimated by the above remaining time of the job p:

$$Pr(z_p = 1) = \alpha * \frac{1}{t_p^{remain}} \tag{16}$$

where $z_p = 1$ is an indicator that indicates whether the job needs to be live migrated, $Pr(z_p = 1)$ is the probability that the job needs to be live migrated, α is a parameter. The probability that the job needs to be migrated live is inversely proportional to the remaining time from end of the job to the contract expiration time.

The migration cost can be estimated by a function which is related to the dirty rate and the size of memory. We use the equation in [34] to estimate the migration cost:

$$Migration_p(t) = \begin{cases} 0, \ p \text{ is scheduled locally} \\ (\eta Size(v_p) + \iota) Pr(z_p = 1), \ otherwise \end{cases} \tag{17}$$

where η, ι is the parameter in the live migrating cost estimating model, $Size(v_p)$ is the estimated live migration size of the job p which can be calculated by the algorithm in [34]. The actual size of migrating is related to the kind of jobs running on the VM. This function calculates the migrating cost if the job is scheduled in time slot t to the contract.

As shown above, we use a simplified live migration model to calculate the migration cost. For a more accurate estimation of the migration cost, the migration model may be replaced with other sophisticated models such as [35] that considers the migration bandwidth and the page dirty rate, [36] which considers the bandwidth waste of in-band migration and the downtime of the job or [37] which optimizes the live migration cost on a Wide Area Network (WAN). Thus, the key idea behind the contracts-based contracts duration-aware scheduling algorithm is to minimize the possible cost of having to migrate the tasks.

5.2.3 Contracts duration-aware and cost-aware scheduling

Considering the two mechanisms discussed above, we develop a new mechanism for scheduling the jobs with the features of both contracts duration-aware and cost-aware scheduling with using the contracts (ConBCAConA).

In this approach, the provider first sorts all the contracts by their unit buying price and separates them into two sets: lower cost contracts which has lower cost than the local resource; higher cost contracts which has higher cost than the local resource which is only used when the workload is beyond the capacity of the previous two sets of resources.

In every time slot t, the provider schedules the tasks by considering the utility of finishing the task p:

$$u_i^p(t) = \varrho_i Res(p) * l_p - Migration_p(t) - Cost_i^p \tag{18}$$

where $Res(p)$ is the resource task p needs, $Cost_i^p$ is the operating cost depending on whether the job runs locally (calculated by the local operating cost) or on the remote contract resources (calculated by the buying price).

The scheduler needs to maximize the outsourcing profit and minimize the cost of using the contracts and live migrating the jobs as shown in Algorithm 2. For each time slot t, the time complexity of the scheduling Algorithm 2 is determined by the number of jobs $|P|$ so that the time complexity is $O(|P|)$.

Algorithm 2: Algorithm of Scheduling the jobs based on contracts

```
Separate the contracts in time slot τ into two set: Contracts_low and
  Contracts_high;
foreach Scheduling time slot t ∈ [t_τ^begin, t_τ^end] do
    if job p can be scheduled to Contr_j ∈ Contracts_low then
        Schedule the job p which Pr(z_p = 1) ≤ ε with Contr_j;
    end
    else
        if Job p can be scheduled to local resource then
            Schedule the job p to the local resource
        end
        else
            Schedule the job p which Pr(z_p = 1) ≤ ε with
              Contr_j ∈ Contracts_high;
        end
    end
end
```

6 Evaluation

In this section, we present our experimental study on the performance of the proposed contracts-based resource management techniques.

6.1 Setup

We implement the simulator and the proposed algorithms in JAVA and the simulator runs with a global virtual clock. We log the status of all the servers in the CSPs for each virtual second which includes the available resources, the job status (submitted, running or finished) information, and the performance metrics of the jobs. We run all the experiments on an Intel i5-3210M Machine with 8GB memory.

6.1.1 datacenters

We consider that each provider has one datacenter in our evaluation and we use the default configuration shown in Table 3 for each datacenter. The default server has the same

TABLE 3
Datacenters' Default Configuration

# of providers	**25**	# of servers / provider	**300**
Cores per server	**6**	MIPS per Core	**3067**
Memory per Server	**16 GB**	Bandwidth per Server	**1 GB**
PUE	**1.2**	Contract Interval	**4 hours**
Prepaid ratio	**0.5**	Maximal response time	**3600**

performance as the IBM server x3550 (2 x [Xeon X5675 3067 MHz, 6 cores], 16GB). The different power consumption of the server from [31] is shown in Table 2. The locations of the datacenters are chosen from Amazon's AWS datacenters' locations with the timezone and location from [30]. The price model uses the AWS EC2 On-Demand price model [38]. The actually model is given by: $\beta_0 + \beta_1 * ECU + \beta_2 * Memory$, and from the linear regression, we get $\beta_0 = 0.0005884, \beta_1 = 0.0093460, \beta_2 = 0.0076067$. Here, one ECU equals 1000 MIPS (Million Instructions Per Second) in our definition.

6.1.2 Real electricity price

The electricity price is generated based on the hourly real-time electricity price from [19]. We obtained the distribution of the data in 2015 from NationalGrid's hourly electricity price and the distribution includes two type of features: one is auto-correlation which means that the electricity price's trend has very high possibility to be similar in a period of observation; another is the burstiness, which indicates that the electricity price can fluctuate significantly in a short period. In our simulation, we use the distribution of the electricity price and randomly choose each day's price from it by shifting the time based on the datacenters' time zones.

6.1.3 Workload

We conduct the experiments using a real-world workload trace from the online Parallel Workload Archive (PWA) repository [39]. We choose the SHARCNET clusters' trace from the archive as it is a computational-intensive High-performance Computing (HPC) workload trace logged in real clusters. The trace is for a duration of thirteen months (From Dec. 2005 to Jan. 2007) with 1,195,242 independent jobs [40], [41]. As suggested by the publisher of PWA, we do not use the entire SHARCNET trace as the configuration of the clusters has changed during the duration of the trace. Instead, as recommended, we extract a two-day (From Dec. 1st to Dec. 2ed) period that does not contain cluster configuration changes. For our simulation, we have extracted the following information from the trace: the job submitted time, the job running time, the requested number of CPUs, the requested size of memory and the utilization of CPUs. In order to keep the workload same in every sub-experiment, we have used two different methods to generate the workloads. Except for the evaluation of testing the impact of number of providers in Section 6.2.2, in other scenarios, all the jobs in the workload are replicated in each provider so as to keep the same amount of jobs in each sub-evaluation. For the evaluation of the different number of providers, we have replicated the extracted trace 25 times that corresponds to half of the maximum number of providers in the evaluation and we have randomly assigned the jobs to the CSPs so as to maintain the same amount of jobs in each sub-evaluation. We set the dedicated resources in the simulation using a set of hierarchical categories having 256, 128, 64, 32, 16, 8, 4 servers respectively. By default, we assume an accurate prediction of the workload at each CSPs. For evaluating under conditions of erroneous predictions, we dedicate a separate set of experiments to test the performance of our algorithms under various levels of errors in workload prediction.

6.1.4 Algorithms

The reference algorithms include: (i) no federation (NF), which does not share any resources and workload with others and the scheduler tries the best effort to minimize the electricity cost; (ii) contracts-based scheduling algorithm using the local resource first (ConBLF); (iii) contracts-based scheduling algorithm with contracts cost-aware (ConBCA) scheduling; (iv) contracts-based contracts duration-aware scheduling algorithm that avoids live migration (ConBConA); (v) contracts-based contracts cost-aware and duration-aware scheduling (ConBCAConA); (vi) the last candidate approach for comparison is the unrealistic method which uses a greedy algorithm (filling the jobs to the datacenter with lowest operating cost by the FCFS policy) to optimize the operating cost across all the datacenters without considering the local datacenter's profit. We refer to it as real-time complete cooperation scheduling (RT). The summary of the algorithms is shown in Table 4.

TABLE 4
Compared Algorithms

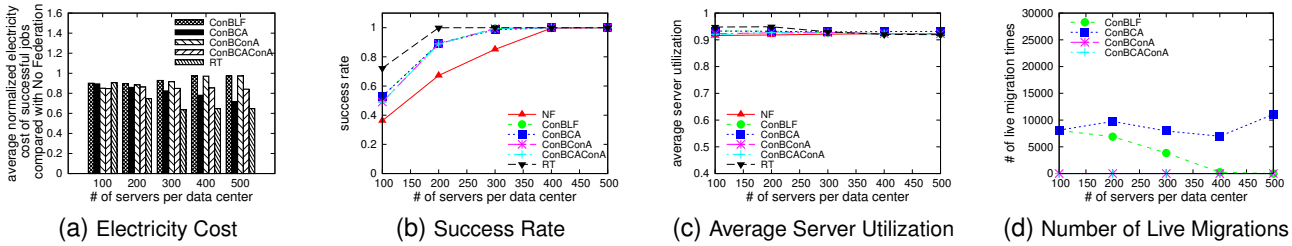
Algorithm	Description	Use Contracts	Cost-aware	Contracts duration-aware
NF	No Federation			
ConBLF	Contracts-based Local First	√		
ConBCA	Contracts-based Cost-aware	√	√	
ConBConA	Contracts-based Contracts duration-aware	√		√
ConBCAConA	Contracts-based Contracts cost-aware and duration-aware	√	√	√
RT	Real Time complete cooperation			

6.2 Experimental Results

For illustrating the performance of our contracts-based algorithms, we complete five sets of experiments: first, we study the impact of increasing the number of servers in the datacenters; second, our experiment analyses the impact when the number of datacenters is increased; third, we add different amount of errors to the prediction of the workloads and study its impact; fourth, we test the influence of different contract intervals on the performance; finally, we evaluate the fairness of our algorithm compared to global optimization approaches that do not consider local profits of the individual datacenters. For each set of the first four experiments, we measure the electricity cost per successful job, the success rate, the average server utilization and the number of job live migrations.

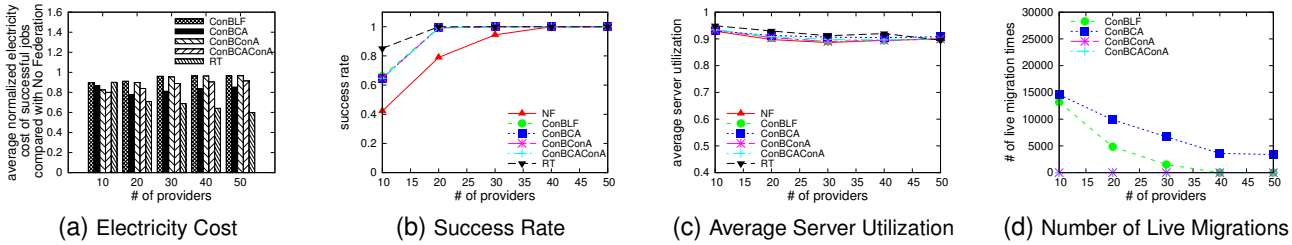
6.2.1 Impact of Number of Servers

We first test the performance of our mechanisms using different number of servers per datacenter. The number of servers increases from 100 to 500 per datacenter in the evaluation. As shown in Figure 4a, the y-axis is the normalized electricity cost per successful job compared with NF. The x-axis is the number of servers per datacenter. We observe that with the ability to share resources in the cloud federation, the electricity cost compared with no federation has been optimized from about 10% to 40% as the number of servers increase. ConBCA achieves the best result and it is close to that of RT. This is due to the fact that if the provider does not need to consider violating the contracts or the cost of live migrations, it can send all the jobs to the lower



(a) Electricity Cost (b) Success Rate (c) Average Server Utilization (d) Number of Live Migrations

Fig. 4. Evaluation results for a different number of servers per datacenter



(a) Electricity Cost (b) Success Rate (c) Average Server Utilization (d) Number of Live Migrations

Fig. 5. Evaluation results for a different number of datacenters

cost contracts it holds. As this will increase the utilization of the low cost contracts, it can potentially decrease the operating cost per successful job. In Figure 4b, we observe that the success rate increases with increase in the number of servers. The priority of sharing the resources are mainly reflected when the resources are scarce. Thus, if the resource is scarcer, the difference can be larger. In Figure 4c, we can observe that our contracts-based mechanism and the RT mechanism obtain better results (around 2%) as the sharing mechanism makes the workload more balanced in the providers which increases the utilization of the servers. In Figure 4d, we observe that the techniques that avoid live migration (ConACB and ConACAConB) achieve the best result. The cost-aware algorithm performs poorly as it uses the contracts regardless of the effective time of the contracts. ConBLF which is the local resource first mechanism also does not perform well (near 8K live migrations when the number of servers is 100) as it ignores the length of the contract. ConBLF performs better than the ConBCA when the number of servers is increased as ConBLF uses local resource first strategy. Therefore, if the resource is available, the usage of the contracts will decrease and the number of live migrations will be decreased as well.

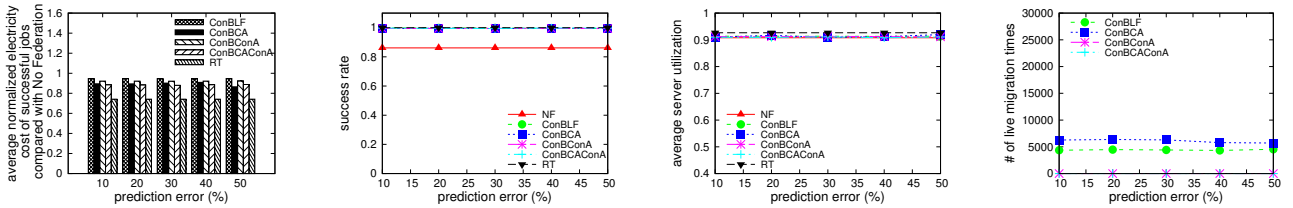
Overall, we can deduce that contracts-based algorithm performs significantly better than the NF scheme with respect to operating cost and success rate. Live migration is avoided when using contracts duration-aware mechanisms (ConBConA and ConBCAConA) and the performance of our contracts-based mechanisms is close to that of the RT method in most of the measurements.

6.2.2 Impact of Number of Providers

We next evaluate the performance of our mechanisms to study the impact of different number of service providers in the federated cloud. The number of providers is increased from 10 to 50. As shown in Figure 5a, the y-axis is the normalized electricity cost per successful job compared with no federation. The x-axis is the number of providers which remains unchanged in this set of evaluation. We can observe that the electricity cost compared with no federation has been optimized from about 10% to 20% in the proposed schemes. This result is similar to the previous experiment. Here, ConBCA optimizes the electricity cost very significantly compared to the other contracts-based algorithms when the resources are sufficiently available in the federation. As shown in Figure 5b, the success rate also increases with increase in the number of providers. Thus, all contracts-based algorithms perform better than NF with more than 20% improvements. Also, the contracts-based algorithms perform similarly to RT when the number of providers is more than 20. The utilization levels measured in Figure 5c also show a similar trend as previous experiments. We can see that when the number of providers is increased, the number of live migrations of ConBLF and ConBCA are decreased. The reason is that when the resources are available and when the number of jobs submitted to each datacenter is decreased, the number of live migrations also decrease. *From the above experiments, we can conclude that contracts-based algorithm performs better than the NF in operating cost and success rate. Live migration is avoided when using contracts duration-aware mechanisms (ConBConA and ConBCAConA). Most of the measurements of our contracts-based mechanisms are close to the RT.*

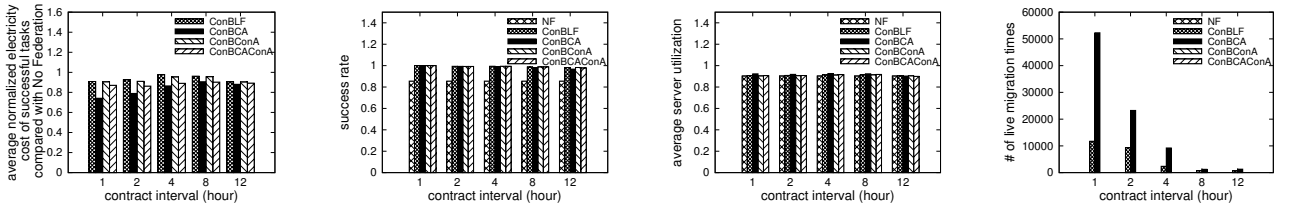
6.2.3 Impact of Prediction Errors

We next evaluate the performance of our mechanisms using different proportions of prediction error added to the demand and workload prediction. We use a white noise [42] to introduce the error in the predicted values. The amount of error introduced is increased from 10% to 50% in the experiment. All the other settings are kept as shown in the default configuration Table 3. As shown in Figure 6a, the y-axis is the same as the previous evaluations. The x-axis is the average percentage of prediction errors. The electricity cost compared with no federation has been optimized from about 10% to 15% regardless of the prediction errors. The result shows that the prediction errors do not influence the result significantly(2% with 50% added error) as the error only influences the contract trading volume. The resources are traded between the providers with the true value. As only the volume is influenced insignificantly by the added error, it does not have a significant impact on the outcome.



(a) Electricity Cost (b) Success Rate (c) Average Server Utilization (d) Number of Live Migrations

Fig. 6. Evaluation results for different average errors of the workload predictions



(a) Electricity Cost (b) Success Rate (c) Average Server Utilization (d) Number of Live Migrations

Fig. 7. Evaluation results for different inner intervals of the contracts

In Figure 6b, we also observe that the success rate is not influenced much (less than 1% variance) by the prediction error. We find a similar trend with the measurements on utilization and live migrations in Figure 6c and Figure 6d respectively. *From the above experiments we can deduce that the prediction error does not influence our algorithm significantly and hence the proposed techniques are robust under a wide range of errors in the workload prediction.*

6.2.4 Impact of Contract Intervals

Next, we test the performance of our mechanisms with different duration of contract intervals. The contract interval is set to five values (1, 2, 4, 8 and 12 hours). As shown in Figure 7a, the x-axis is the contract interval. We can see that the contract interval does not influence the operating cost significantly. But when the contract interval increases, the normalized electricity price is also increased considerably between 3% to 5% except for ConBCA. ConBCA is influenced more and has an increase of 16%. This is because with a longer interval, the evaluation accuracy of the true values for each time slot will decrease which will influence the contracts establishment process. However, the influence is not very significant. In Figure 7b, we observe that the success rate is not influenced significantly by the contract intervals and in Figure 7c, we note that the average utilization of the running servers also does not change significantly. As shown in Figure 7d, the number of live migrations decreases when the contract interval increases(for LFConB, from 12K to 0.6K; for ConBCA, from 52K to 1K) as the longer effective time decreases the probability of live migrations. The contracts duration-aware mechanisms (ConBConA and ConBCAConA) also perform better here.

Overall, from the above experiment, we can see that the contract interval influences the number of live migration and electricity cost. With an increase in the contract interval, the normalized electricity cost is slightly increased and the number of live migrations is decreased.

6.2.5 Fairness

In this set of experiment, we evaluate the fairness of the proposed schemes by comparing the individual profit of each CSP. The result is observed with the setup of 100 servers per datacenters. We use the default setting for the other experiment parameters. As shown in Figure 8, the y-axis is the gain or loss ratio of the normalized profit which is the difference between the profit that can be earned using the federated cloud and the profit that can be earned otherwise when operating alone. When the number is larger than 0, it means that the provider earns more using the federated cloud than when it operates alone and vice versa. The x-axis represents the index for each CSP. From the figure, we can see that, when the federated cloud is operated using the real-time complete cooperation mechanism, there are six CSPs (CSP4, 5, 6, 8, 13, 22) of the total 25 CSPs losing profits compared with the profits they can earn when operating alone. The contracts-based mechanisms perform significantly better than operating alone except for CSP10 which gets a very minimal decrease (less than 4%) compared to the profit that it can earn from operating alone.

From the observations above, we can see that the real-time complete cooperation mechanism globally optimizes the operating cost but results in several CSPs losing profits. In contrast, the proposed contracts-based mechanisms perform better than the real-time complete cooperation mechanism and achieve higher fairness with most of the CSPs obtaining higher profits when participating in the federation.

7 Related Work

Existing literature [10]–[16] have focused on optimizing the performance of cloud services in the Geo-distributed Cloud environment. This class of techniques builds Virtual Machines (VMs) for users to use computing resources across geo-distributed datacenters as a single logical virtual cluster. These techniques primarily ptimize the data placement [10] [12], the latency of the services [12] [13] [14] , the Quality of Service(QoS) [11] [15], the electricity cost [13] [16] across multiple datacenters.

In the recent past, cloud Federation has gained significant focus from the cloud computing research community. Most of the works related to Cloud Federation primarily focus on two aspects: the first kind of research efforts focus on the architecture and the system model for enabling and

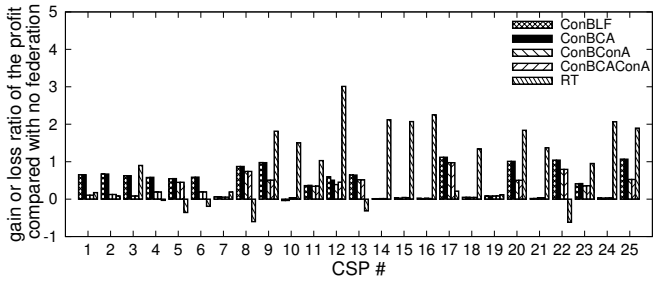


Fig. 8. The gain or loss ratio of the profit for each individual CSP

deploying federated clouds [5]–[7], [43], [44]; the second class of existing work optimizes the performance of federated cloud through efficient job scheduling, job migration and resource allocation [8], [13], [45]–[47]. Rochwerger et al. [6] proposed an architecture called RESERVOIR to enable cloud providers to deal with each other in a P2P manner. Buyya et al. [7] proposed a centralized architecture named InterCloud which provides a market for the CSPs or cloud brokers to share their resources. In [5], Carlini et al. proposed a centralized architecture for achieving federated cloud which provides single sign-on and both centralized and decentralized architecture for building the federated cloud. Moreno et al. [44] proposed a cloud-broker-based architecture to support the management of the federated cloud. In [43], Ferrer et al. proposed OPTIMIS which is a toolkit for implementing peer-to-peer Cloud Federation provisioning.

McGough et al. [45], [46] analyzed and optimized the power consumption in a commercial framework for establishing Cloud Federations. In [8], Li et al. proposed a model which makes it possible for one Cloud Federation to consider both the workload and the electricity price and maximize the profit through an auction mechanism. Xu et al. [13] proposed a technique that combines the alternating direction method of multipliers (ADMM) method with the problem of how to place the cloud services with minimized electricity cost and latency to the client. However, the work is based on the geo-distributed cloud assumption which assumes that all the services' and users' information can be accessed by the provider, which is not applicable for generic federated clouds with competing CSPs, such as the one considered in our work. In [47], Breitgand et al. proposed a method that uses policy similar to contracts to optimize the service placement problem in federated clouds. However, this work primarily addresses the architectural issues of the policy enforcement but not the combination of the contracts-based solution with optimization based on contracts, which is the key focus of our work.

8 Conclusions and Future work

In this paper, we proposed a contracts-based mechanism for resource sharing between CSPs in a federated cloud. Compared with previous work in this area, our proposed approach considers both the global cost minimization as well as the local profit maximization of each individual datacenters participating in the federation process. We developed an auction-based mechanism for contract establishment and a suite of contracts cost-aware and duration-aware scheduling techniques that maximize the local profits of the CSPs while meeting the individual job requirements. We evaluated the performance of the proposed approach using a trace-driven simulation study with realistic workload traces and electricity pricing. The contracts-based solution achieves good performance and performs significantly better than the traditional model in terms of fairness in local profits while achieving similar operational costs and success rate properties as existing methods.

In the future, we plan to address three limitations of our current work. First, the auctioneer in our present model represents a single point of failure. One direction of future work may focus on developing the auctioneer in a decentralized way so that single point of control and trust can be eliminated. Second, in our current work, we do not account for data-intensive workloads in the resource allocation model. The second direction may extend our resource allocation and job scheduling techniques to take into consideration data-locality and data-affinity constraints so that both data-intensive and computationally intensive workloads can be supported in the model. Finally, another direction of our future work may focus on considering additional SLA requirements in the resource allocation framework including job reliability and fault-tolerant requirements to further enhance the resource allocation model.

Acknowledgements

The workload logs on which it is based are available on-line from the Parallel Workloads Archive [39].

References

[1] D. Laney, "3d data management: Controlling data volume, velocity and variety," *META Group Research Note*, vol. 6, p. 70, 2001.

[2] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: The next frontier for innovation, competition, and productivity," 2011.

[3] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of big data on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98–115, 2015.

[4] D. Agrawal, S. Das, and A. El Abbadi, "Big data and cloud computing: current state and future opportunities," in *Proceedings of the 14th International Conference on Extending Database Technology*. ACM, 2011, pp. 530–533.

[5] E. Carlini, M. Coppola, P. Dazzi, L. Ricci, and G. Righetti, "Cloud federations in contrail," in *European Conference on Parallel Processing*. Springer, 2011, pp. 159–168.

[6] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 4–1, 2009.

[7] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2010, pp. 13–31.

[8] H. Li, C. Wu, Z. Li, and F. C. M. Lau, "Profit-Maximizing Virtual Machine Trading in a Federation of Selfish Clouds," *IEEE/ACM Transactions on Networking*, pp. 1–1, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7124534

[9] S. Ren, Y. He, and F. Xu, "Provably-efficient job scheduling for energy and fairness in geographically distributed data centers," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012, pp. 22–31.

[10] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services." in *NSDI*, 2010, pp. 17–32.

[11] H. Roh, C. Jung, W. Lee, and D.-Z. Du, "Resource pricing game in geo-distributed clouds," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 1519–1527.

[12] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 2015, pp. 421–434.

[13] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 854–862.

[14] B. Yu and J. Pan, "Location-aware Associated Data Placement for Geo-distributed Data-intensive Applications."

[15] L. Gu, D. Zeng, P. Li, and S. Guo, "Cost minimization for big data processing in geo-distributed data centers," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 314–323, 2014.

[16] Z. Zhou, F. Liu, Y. Xu, R. Zou, H. Xu, J. C. Lui, and H. Jin, "Carbon-aware load balancing for geo-distributed cloud services," in *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 2013, pp. 232–241.

[17] Amazon, "AWS Amazon EC2," accessed Jan. 4, 2016. [Online]. Available: https://aws.amazon.com

[18] "Google Cloud," accessed Aug. 4, 2016. [Online]. Available: https://cloud.google.com

[19] National Grid, "Large General TOU," accessed Jan. 4, 2016. [Online]. Available: https://www.nationalgridus.com/niagaramohawk/business/rates/5_hour_charge.asp

[20] P. Mell and T. Grance, "The nist definition of cloud computing," 2011.

[21] C. Camerer, *Behavioral game theory: Experiments in strategic interaction*. Princeton University Press, 2003.

[22] IBM, "Bluemix Dedicated," accessed Nov. 7, 2016. [Online]. Available: https://www.ibm.com/cloud-computing/bluemix/dedicated

[23] Amazon, "Amazon EC2 Dedicated Instances," accessed Nov. 11, 2016. [Online]. Available: https://aws.amazon.com/ec2/purchasing-options/dedicated-instances/

[24] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press Cambridge, 2007, vol. 1.

[25] R. P. McAfee, "A dominant strategy double auction," *Journal of economic Theory*, vol. 56, no. 2, pp. 434–450, 1992.

[26] N. Nisan *et al.*, "Introduction to mechanism design (for computer scientists)," *Algorithmic game theory*, vol. 9, pp. 209–242, 2007.

[27] "Double auction," accessed Aug. 29, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Double_auction

[28] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications qos," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449–458, 2015.

[29] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 1287–1294.

[30] "AWS Regional Data Centers mapping," accessed Jan. 4, 2016. [Online]. Available: https://github.com/turnkeylinux/aws-datacenters

[31] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[32] M. R. Gary and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," 1979.

[33] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 273–286.

[34] H. Liu, H. Jin, C.-Z. Xu, and X. Liao, "Performance and energy modeling for live migration of virtual machines," *Cluster computing*, vol. 16, no. 2, pp. 249–264, 2013.

[35] S. Akoush, R. Sohan, A. Rice, A. W. Moore, and A. Hopper, "Predicting the performance of virtual machine migration," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 37–46.

[36] D. Breitgand, G. Kutiel, and D. Raz, "Cost-aware live migration of services in the cloud." in *SYSTOR*, 2010.

[37] T. Wood, K. Ramakrishnan, P. Shenoy, and J. Van der Merwe, "Cloudnet: dynamic pooling of cloud resources by live wan migration of virtual machines," in *ACM Sigplan Notices*, vol. 46, no. 7. ACM, 2011, pp. 121–132.

[38] Amazon, "AWS Amazon EC2 On-demand Pricing," accessed Jan. 4, 2016. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/

[39] "Logs of real parallel workloads," accessed Aug. 29, 2017. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/

[40] D. Klusáček and H. Rudová, "Alea 2: job scheduling simulator," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010, p. 61.

[41] I. Rodero, F. Guim, and J. Corbalan, "Evaluation of coordinated grid scheduling strategies," in *High Performance Computing and Communications, 2009. HPCC'09. 11th IEEE International Conference on*. IEEE, 2009, pp. 1–10.

[42] Y. Wu, K. Hwang, Y. Yuan, and W. Zheng, "Adaptive workload prediction of grid performance in confidence windows," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 7, pp. 925–938, 2010.

[43] A. J. Ferrer, F. HernáNdez, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame *et al.*, "Optimis: A holistic approach to cloud service provisioning," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, 2012.

[44] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures," *Computer*, no. 12, pp. 65–72, 2012.

[45] A. S. McGough, C. Gerrard, P. Haldane, D. Sharples, D. Swan, P. Robinson, S. Hamlander, and S. Wheater, "Intelligent power management over large clusters," in *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*. IEEE Computer Society, 2010, pp. 88–95.

[46] A. S. McGough, C. Gerrard, J. Noble, P. Robinson, and S. Wheater, "Analysis of power-saving techniques over a large multi-use cluster," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*. IEEE, 2011, pp. 364–371.

[47] D. Breitgand, a. Marashini, and J. Tordsson, "Policy-driven service placement optimization in federated clouds," *IBM Research Division, Tech. Rep*, vol. 0299, 2011.

Jinlai Xu is currently a PhD student in the School of Computing and Information at University of Pittsburgh He received the BE and ME degrees from China University of Geosciences, China, in 2012 and 2015, respectively. His research interests are in the areas of Edge/Fog and Cloud Computing, with a current focus on contracts-based resource sharing and allocation in edge/fog and cloud computing. He is a student member of the IEEE.

Balaji Palanisamy is an Assistant Professor in the School of Information Science in University of Pittsburgh. He received his M.S and Ph.D. degrees in Computer Science from the college of Computing at Georgia Tech in 2009 and 2013 respectively. His primary research interests lie in scalable and privacy-conscious resource management for large-scale Distributed and Mobile Systems. At University of Pittsburgh, he co-directs research in the Laboratory of Research and Education on Security Assured Information Systems (LERSAIS), which is one of the first group of NSA/DHS designated Centers of Academic Excellence in Information Assurance Education and Research (CAE &CAE-R). He is a member of the IEEE.