Cost-aware Resource Management for Federated Clouds Using Resource Sharing Contracts

Jinlai Xu
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15213, USA
Email: jinlai.xu@pitt.edu

Balaji Palanisamy
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15213, USA
Email: bpalan@pitt.edu

***Abstract*—Cloud computing and its pay-as-you-go model continue 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 raises new challenges in terms of effective global resource sharing and management of autonomously-controlled individual datacenter resources. Earlier solutions for geo-distributed clouds have focused primarily on achieving global efficiency in resource sharing that results in significant inefficiencies in local resource allocation for individual datacenters leading to unfairness in revenue and profit earned. In this paper, we propose a new contracts-based resource sharing model for federated geo-distributed clouds that allows cloud service providers 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 cloud service providers employ a cost-aware job scheduling and provisioning algorithm that enables tasks to complete and meet their response time requirements. The proposed techniques are evaluated through extensive experiments using realistic workloads and the results demonstrate the effectiveness, scalability and resource sharing efficiency of the proposed model.**

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

I. Introduction

In the age of Big Data, we are witnessing a massive data growth in scale, volume and velocity [1]. Cloud computing and its pay-as-you-go model continue to provide significant cost benefits and a seamless service delivery model for cloud consumers. The recent growth 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]. The evolution of small-scale and large-scale geo-distributed datacenters operated and managed by individual cloud service providers raises new challenges in terms of effective global resource sharing and management of autonomously controlled individual datacenter resources. 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 [3]–[5] aims at extending the capacity of the datacenter resources by leveraging resources in remote datacenters that are underutilized or available at a reduced cost. Thus, the federation can be used to handle burstiness in workloads, fluctuations in electricity price and respond 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.

Earlier solutions for geo-distributed clouds have focused primarily on achieving global efficiency in resource sharing. Several mechanisms have been proposed to achieve higher utility and overall global profit [6]–[9]. Most geo-distributed resource allocation techniques proposed in the past [6]–[12] 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 with respect to the revenue earned before and after the federation.

In this paper, we propose a new contracts-based resource sharing model for federated geo-distributed clouds that allows cloud service providers 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 cloud service providers employ a cost-aware job scheduling and provisioning algorithm that enables tasks to complete and meet their response time requirements. We first develop an optimal contract establishment algorithm that produces the optimal design of resource sharing contracts for the proposed model considering the size and type of resources. Next, we develop an auction-based contract allocation mechanism that ensures both fairness and revenue maximization for the individual datacenter providers. Finally, our proposed techniques employ efficient job scheduling over the established contracts to minimize the resource usage cost of individual cloud service providers. We evaluate the proposed techniques and the results demonstrate the effectiveness, scalability and resource sharing efficiency of the proposed model.

The remainder of this paper is organized as follows. Section II provides a background of various resource sharing models for geo-distributed clouds and motivates the proposed contracts-based model. In Section III, we present our techniques for optimal contracts designing and briefly discusses the cost-aware resource job-scheduling techniques. Section IV evaluates the performance of the contracts-based resource allocation mechanism in comparison with conventional geo-distributed clouds using real-world datacenter workload traces.

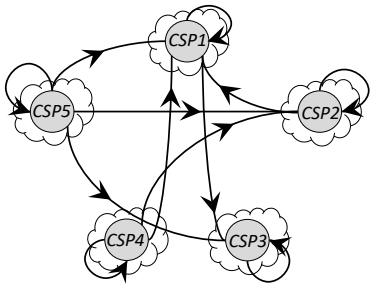


Fig. 1. Contracts-based resource sharing

Section V discusses the related work and we conclude in Section VI.

II. 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.

The conventional cloud computing model operates as "stand-alone" clouds in which every datacenter of the cloud provider operates on its own. In the "stand-alone" model, even when a cloud service provider has multiple datacenters, resources are not shared between them for optimizing the resource allocation and management. Thus, 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. However, this model may result in sub-optimal resource allocation with respect to global resource management considering all datacenter resources jointly in a federated geo-distributed cloud scenario. For instance, the dynamic electricity price data from National-Grid [13] suggests that there are significant short-term price variations even on a single day besides the notable long-term (e.g., one year) fluctuations: the highest per-day pricing on a 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 between the individual datacenters can operate very sub-optimally forcing individual datacenters to run entire workloads locally at higher electricity prices even though resources for which may be available at remote datacenters at a possibly lower electricity consumption cost.

In contrast to the "stand-alone" model, when computing resources are shared across data centers, there is an opportunity for significant improvement in terms of both resource utilization and minimizing operating cost. To achieve resource sharing efficiency among multiple geo-distributed datacenters, several mechanisms have been proposed in the past, which can be classified into two broad categories:

- Virtual Geo-distributed Clusters: Virtual geo-distributed clusters provide a mechanism for users to use computing resources across geo-distributed datacenters as a single virtual cluster. Virtual geo-distributed clusters help optimize resource management including data placement [10] [11], service latency [11] [6] [12] and electricity cost [6] [9].
- Federated Cloud: Federated clouds enable a resource sharing mechanism for CSPs to share computing resources federated in different datacenters. Here, each CSP is assumed to manage its datacenters autonomously and there is often a centralized Cloud Exchange Institution fetching all the infrastructure information from the datacenters. It provides a platform for the CSPs to discover the resources from the members of the federated cloud which can then be used to accept the dynamically arising requests [3] [4] [5].

Both the mechanisms we discussed above either enable the free use of the resources in the pool of the virtual cluster or through a centralized broker to schedule the jobs across all the datacenters based on some criteria such as resource pricing, network delay and cost. Since in this model, all of the resources in the geo-distributed datacenters can be used by all the other members or users in the system, we refer to it as Federated clouds with complete cooperation. However, this model suffers from a few key drawbacks which include (i) lack of fairness for datacenters managed 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, without participating in the federation process 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.

In this paper, we propose a new contracts-based resource sharing architecture for CSPs to share resource across the globally geo-distributed datacenters. The demerits of the complete cooperation model lead us to a more flexible and limited sharing mechanism that provides a controlled cost-aware resource sharing opportunity that finds suitable tradeoffs between traditional clouds without federation and that with the complete cooperation as illustrated in Figure 1. The figure shows the architecture with five CSPs (the nodes represent the CSPs and the edges represent the co-operation among the CSPs to share resources). Compared with the traditional "stand-alone" cloud model where each CSP vertex in the graph representation has an edge connected to itself and the complete cooperation model which can be represented as a complete graph where every vertex connects with all the other vertices, the contracts-based model can be represented by a partial graph as shown in Figure 1. Here, each CSP is not required to necessarily share resources with every other CSP in the federation. Each edge can be a resource sharing contract between the CSPs to handle the resource allocation problem. The resource sharing contracts mandate the CSPs to share the committed resources during the contract time duration at the negotiated price in the contract. For the proposed contracts-based resource sharing mechanism, the CSPs design and trade the resource sharing contracts with each other. The contract may be predetermined and established apriori before the effective time. Both the estimated workload pressure and

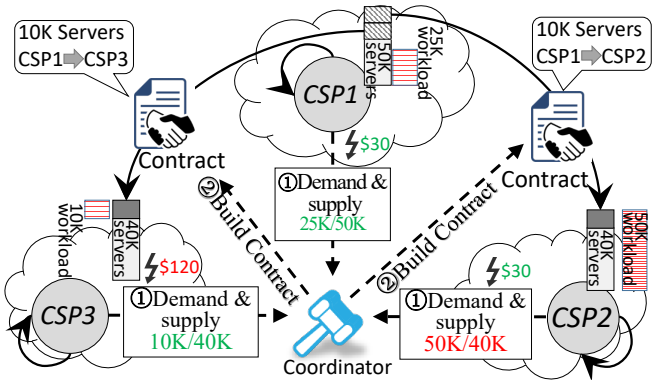


Fig. 2. Contracts-based Federated Cloud Example

the fluctuating operating cost are considered in the contract design during the trading time.

An example of the proposed contracts-based resource sharing mechanism is shown in Figure 2. Here, the negotiations are based on the demands and supplies from the CSPs and the contracts are established through a trusted coordinator as Step ① illustrated in Figure 2. The resource sharing contracts could be agreed by the CSPs with the help of the coordinator to share the committed resources during the time duration at the negotiated price in the contract as Step ② illustrated in Figure 2.

The establishment of the contracts in the proposed resource sharing mechanism involve two major challenges: the first involves the design and establishment of the contracts which can guarantee the individual profit for each CSP; the second deals with scheduling the jobs using the contracts to maximize the overall utility and minimize cost.

The design of the contract involves contract establishment decision and optimization problems as Step ① and ② illustrated in Figure 2 requiring the coordinator to decide the effective contract time, the type and amount of resources, the price of resources and the agreement on both sides in the contract.

We present an example in Figure 2 to illustrate how the contracts-based federated cloud resource sharing operates in the trading process:

1) As illustrated in Figure 2, CSP2 which has workload demands beyond its capacity needs 50K servers at least to run the workload but only has 40K servers, which means it has to either delay or drop some jobs in the workload. Alternately, within the federated cloud, it can lend resources from other CSPs to fulfill the demand. Here CSP2 lends 10K servers from CSP1 to fulfill the 50K server demand.
2) As described in Figure 2, CSP3 currently has a higher electricity cost of $120. Even though it has only 10K workloads, it also wants to outsource some workload to other CSPs to decrease the operating cost. It therefore lends 10K servers from CSP1 which has an electricity cost of $30. The contracts can save at most 75% operating cost for CSP3.

As discussed above, we find that contracts-based resource sharing provides additional opportunities and flexibility to achieve a more efficient resource allocation while minimizing the cost for each individual datacenter. However, this requires new techniques for optimal contract establishment and efficient job scheduling that can realize the full potential of the model. In the next section, we model the problem formally, analyze and develop algorithms for contract establishment and discuss our proposed job scheduling technique to efficiently and cost-effectively share resources between CSPs.

III. Contracts-based Resource Management

Our proposed contracts-based resource management comprises of two components namely (i) the model and solution for establishing contracts among the CSPs and (ii) techniques for scheduling the jobs using resources in the established contracts to minimize cost.

A. Resource Sharing Contracts Establishment

We design an auction-based mechanism to model and solve the contracts establishment problem. Intuitively, the contract establishment problem fits the essence of auctions which are efficient to match the demands and supplies in the market [14]. In addition, in situations when every participant is selfish and aims to maximize its own benefit, auction mechanisms are very suitable and perform efficiently.

We first model the contracts establishment problem as a sealed-bid double auction problem. In a sealed-bid double auction [15], 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 be both buyers and sellers based on their profiles. The coordinator of the federated cloud which is 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 [16] [17] to represent a cloud resource. The dedicated resource can be considered as a bundle 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 model the contracts establishment problem using discrete time slots. We use $\tau \in \{1, 2, ..., T\}$ to denote the contract time slots. We consider a federated cloud with N CSPs, each of which wants to maximize the profit and minimize the cost. We assume that every CSP operates only one datacenter for the sake of modeling simplicity. Each CSP $i \in \{1, 2, ..., N\}$ can be either a buyer or a seller in different time slots for different types of resources. We assume that each datacenter has several types of servers. There is 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 resources that change during each time slot under different contracts established in the time slots. 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 jobs. The job requests are sent to the CSP, which are pushed into a job queue. The jobs in the queue are processed in a FIFO (First In First Out) manner. 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 [18] or Hidden Markov Modeling (HMM) [19]. The profit earned by the CSPs is computed as the difference between the sum of the payments from the users and the operating cost and the penalty. We use ϱ_i to denote the unit price charged from the users for one unit resource. The unit resource denotes the smallest amount of resource that the users can request for running their jobs. We assume that the request for the resources can only be in multiples of unit resources. 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. The trades between the CSPs use dedicated resources such as [16] [17] as the type of resource. We use $Cost_i^k(\tau)$ to denote the operating cost of CSP i to operate and manage the type-k dedicated resource. For the operating cost, we primarily consider electricity cost as the dynamic component which varies in different locations from time to time. Other costs such as space rental and labour remain relatively fixed for a long time so they are just included as a constant in $Cost_i^k(\tau)$.

The contract establishment problem for one particular time slot τ and a particular type-k resource can be defined as a sealed-bid double auction in the following manner. Every CSP bids with a buy bid $b_i^k(\tau)$ and a sell bid $s_i^k(\tau)$ based on their bidding strategies we describe later. The auctioneer considers all the buy and sell bids, $B^k(\tau) = \{b_1, b_2, ..., b_N\}$ and $S^k(\tau) = \{s_1, s_2, ..., s_N\}$, to decide the winning buyers and sellers for a particular time slot and particular resource type-k. The results for the auction are denoted by $X_b^k(\tau) = \{x_{b_1}, x_{b_2}, ..., x_{b_N}\}$ and $X_s^k(\tau) = \{x_{s_1}, x_{s_2}, ..., x_{s_N}\}$. When $x_{b_i} = 1$, it means that $b_i^k(\tau)$ wins the auction and similarly $x_{s_i} = 1$ denotes that $s_i^k(\tau)$ wins the auction. The clear price of the auction is denoted by $\pi_s^k(\tau)$ which is the sell price that decides the payment to the sellers and $\pi_b^k(\tau)$ which is the buy price that determines the payment from the buyers. The contracts are established between the winning buyers and sellers for the time slot considered in the auction. Thus, the contracts establishment problem is modeled into an instance of the sealed-bid double auction problem with the above descriptions.

We next present the algorithm for determining the winners in the auction. As shown in Algorithm 1, the winner decision algorithm is an instance of McAfee auction mechanism [20] that guarantees both *truthfulness* and *weak budget balance*. Here, *truthfulness* guarantees that every bidder can maximize its own utility by only bidding with the true valuation of the goods and *weak budget balance* ensures that the auction result will not cause the auctioneer to subsidize in the auction. The proposed algorithm (Algorithm 1) first sorts the buy bids in the descending order and the sell bids in the ascending order. Then, it finds the break-even index h which is the index of the last profitable trade such that $b_h \geq s_h$. It then decides the winners using the McAfee mechanism [20]. 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 winner selection

Input : Type of dedicated resource : k;
Time slot: τ;
Buy bids: $B^k(\tau) = \{b_1, b_2, ..., b_N\}$;
Sell bids: $S^k(\tau) = \{s_1, s_2, ..., b_N\}$;
Output: Clearing Buy Price: $\pi_b^k(\tau)$ Clearing Sell Price: $\pi_s^k(\tau)$;
Auction decision: $X_b^k(\tau) = \{x_{b_1}, x_{b_2}, ..., x_{b_N}\}$;
$X_s^k(\tau) = \{x_{s_1}, x_{s_2}, ..., x_{s_N}\}$
Sort $B^k(\tau)$ in descending order by b_i and $S^k(\tau)$ in ascending order by s_i;
Initially, set current buy price b as the first buy bid (highest price) in $B^k(\tau)$ and current sell price s as the first sell bid (lowest price) in $S^k(\tau)$. current bid indicator $h = 0$;
while $b \geq s$ **do**
    $s = s_h$;
    $b = b_h$;
    $h = h + 1$;
    if h is larger than N break;
end
$\rho = (b_{h+1} + s_{h+1})/2$;
if $b_h \geq \rho \geq s_h$ **then**
    $X_b^k(\tau) = \{x_{b_1} = 1, ..., x_{b_h} = 1, x_{b_{h+1}} = 0, ..., x_{b_N} = 0\}$;
    $X_s^k(\tau) = \{x_{s_1} = 1, ..., x_{s_h} = 1, x_{s_{h+1}} = 0, ..., x_{s_N} = 0\}$;
    $\pi_s^(\tau) = \pi_b^(\tau) = \rho$;
end
else
    $X_b^k(\tau) = \{x_{b_1} = 1, ..., x_{b_{h-1}} = 1, x_{b_h} = 0, ..., x_{b_N} = 0\}$;
    $X_s^k(\tau) = \{x_{s_1} = 1, ..., x_{s_{h-1}} = 1, x_{s_h} = 0, ..., x_{s_N} = 0\}$;
    $\pi_s^k(\tau) = s_h$, $\pi_b^k(\tau) = b_h$;
end

Next, we present the optimal bidding strategies for the CSPs in the auction. As the McAfee auction mechanism guarantees the *truthfulness*, the optimal strategy for the sellers is to bid at the true valuation of the goods. The true valuation corresponds to the true utility of goods in the auction. Here, the utility for each provider can be defined in two aspects namely the profit it can gain from running the tasks on the resources purchased through the contracts or the profit it can gain from selling the resources to other CSPs in the auction.

First, we define the utility function based on the profit a provider i can get from renting type-k resources from other providers to execute jobs that cannot be run locally:

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

where $Res()$ is a function that estimates the resource usage for the workload. The function $Res()$ is an estimate of the resource usage which can be computed through estimation mechanisms such as the ones discussed in [21] [22].

Secondly, even if resources for servicing the workload demands are available locally but when the operating cost of the local resources is higher, the CSP may choose to participate in the auction to increase the utility by running some part

of the workload on other CSPs' resources to minimize cost. Under this circumstance, 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 CSPs would want to sell their resources to others. That is when there are idle servers. Thus, the utility for a CSP that wants to sell type-k resources to others can be represented by:

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

From the above discussion, we can get the bidding strategies for the CSPs. The provider i can set the sell bid as:

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

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.

A buyer that wants to buy a type-k dedicated resource will typically bid in two conditions namely (i) when the predicted service demand is higher than the capacity of currently available servers and (ii) another is when the expected local operation cost is relatively high in the time slot τ. The bidding strategy can be represented as follows:

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

Based on the winner decision algorithm and the bidding strategies described above, we can summarize the contracts establishment process as a sequence of four steps for each type-k dedicated resource for each time slot τ. First, the CSPs set the buy and sell bids using the bidding strategies represented in Equation (4) and (5) and send the sealed bids to the coordinator. Second, the coordinator decides the winners based on Algorithm 1. Third, the CSPs agree to the contracts based on the result of the auction. Each contract contains six attributes: a buyer CSP i, a seller CSP j, the buy price $\pi_b^k(\tau)$, the sell price $\pi_s^k(\tau)$, the effective time slot τ and the server list D_k for the type-k resource agreed in the contract. Finally, each CSP updates the list of the servers that are controlled by it, namely $M_i(\tau)$, by removing the servers that are sold to other CSPs in the auction and by adding the servers that are purchased in the auction. The capacity $C_i(\tau)$ is updated corresponding to the updated value of $M_i(\tau)$. The above four steps are iteratively performed for each time slot $\tau = 1$ to $\tau = T$ and for each resource type, type-1 to type-K. Therefore, for every time slot τ, we have k auctions, one for each of the type-k resource type in the auction market.

We note that the optimized contracts provide the CSPs with a set of available remote resources in a cost-effective manner. However, the individual CSPs need to employ intelligent job scheduling techniques that understand the cost implications of the underlying contract structure to effectively leverage the remote resources available to the CSPs. We discuss them in the next subsection.

B. Contracts-based Cost-aware scheduling

In this section, we present our proposed contracts-based cost-aware mechanism (ConBCA) to schedule jobs using the extended resources provided to the CSPs through the resource sharing contracts. We note that the contract-based scheduling problem can be reduced to a bin packing problem, which is known to be NP-hard [23]. Therefore, we adopt a heuristic solution for scheduling the jobs based on the contracts using a real-time scheduling algorithm.

Compared with an intuitive solution which schedules the jobs to the local resource first (ConBLF), ConBCA acts in a cost-aware manner: in every time slot t, the scheduler in each CSP first sorts all the contracts by their unit buy price and separates them into two sets: lower cost contracts that have lower cost than the local resource; higher cost contracts that have higher cost than the local resource. The higher cost contracts are only used when the workload resource requirement is beyond the capacity of the previous two sets of resources. Here, the provider schedules the jobs based on the priorities by using lower cost contracts before local resources and uses higher cost contracts only when other resources are entirely utilized.

IV. Evaluation

In this section, we present the results of our experimental study on the performance of our contracts-based resource management algorithms using simulations with a real-world datacenters trace.

A. Setups

1) datacenters: We consider that each provider has one datacenter in our evaluation and we use the default configuration shown in Table I for each datacenter. The default

TABLE I
DATACENTERS' DEFAULT CONFIGURATION

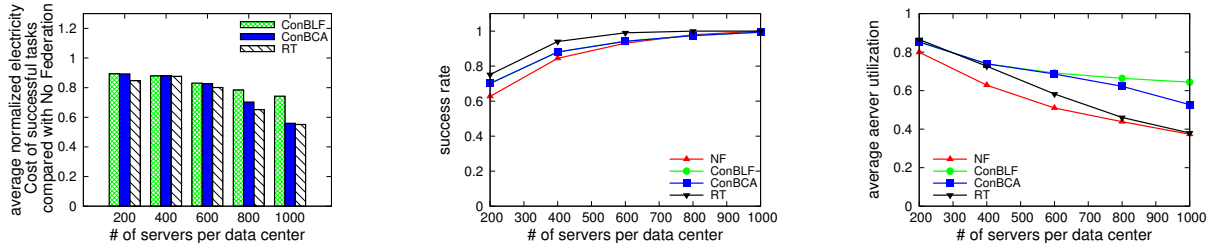
# of providers	**25**
# of servers / provider	**600**
# of cores per server	**12**
MIPS per Core	**3067**
GB of memory per Server	**16**
GB of bandwidth per Server	**1**
PUE	**1.2**
Maximal response time in SLA (s)	**600**
Prepaid ratio for reserved resource	**0.5**

server has the same performance as the IBM server x3550 (2 x [Xeon X5675 3067 MHz, 6 cores], 16GB). The different load power consumption of the server provided in [24] is shown in Table II. The locations of the datacenter are chosen from Amazon's AWS datacenters' locations with the profile of the timezone and the location.

The price model uses the AWS EC2 On-Demand price model, which can be estimated by a linear model calculated by the normal EC2 on-demand instances with the CPU and memory resources (EC2 Compute Unit (ECU) number and memory size as the input and we obtain the instance price per hour). The actually model is: $\beta_0 + \beta_1 * ECU + \beta_2 * Memory$,

TABLE II
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



(a) Electricity Cost (b) Success Rate (c) Average Server Utilization

Fig. 3. Evaluation results for different number of servers per datacenter

from the linear regression, we get $\beta_0 = 0.0005884, \beta_1 = 0.0093460, \beta_2 = 0.0076067$. One ECU equals to 1000 MIPS(Million Instructions Per Second) in our definition. The unit of the memory is gigabyte.

2) Real electricity price: The electricity price is generated based on the hourly real-time electricity price from [13]. We get the distribution of the data in 2015 from NationalGrid's hourly electricity price and use this price distribution to simulate the fluctuation of the real electricity market. In our simulation, we use the distribution of the electricity price dataset and randomly choose each day's price from it by shifting the time based on datacenters' time zone offsets.

3) Workload: We simulate the workload demands which are the replay of the Google cluster trace [25]. We choose Google trace because it provides detailed information of the CPU usage, memory usage and the duration of the tasks which are essential in simulating the allocation and scheduling algorithms. We randomly choose 40% of the tasks in the first two days' tasks of the trace to make the number of the tasks' resource requirements suit the default setting of the datacenter. In the simulator, we replay the trace to simulate the realistic workload. Each task in the trace is sent to one of the cloud service providers with the time zone offsets which are used to simulate the different peak loads for different geo-locations.

4) Dedicated resource type: The resource type which is used in the contract trading is the dedicated resource. We set the dedicated resource types with a hierarchy which respectively have 256, 128, 64, 32, 16, 8, 4 servers. The types also can be set to have another amount of resources. The dichotomous division of each dedicated resource type is for reducing the computation complexity in the simulation.

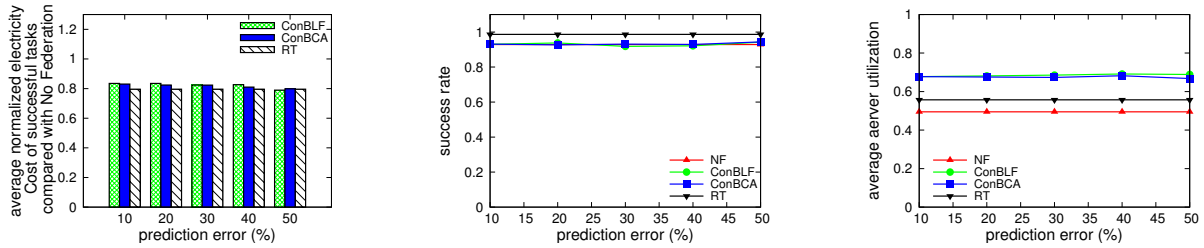
5) Algorithms: The reference algorithms include: (i) no federation (NF), which does not share any resources and workload with others and the scheduler try their best to delay the job to decrease the electricity cost; (ii) contract-based scheduling algorithm with no optimization and use the local resource first (ConBLF); (iii) contract-based scheduling algorithm with contract cost-aware (ConBCA) scheduling; (iv) the last candidate approach for comparison is a less realistic approach which optimizes the operating cost across all the datacenters without considering the individual datacenter's profits. We refer to it as real-time complete cooperation scheduling (RT).

6) Metrics: We measure the electricity cost (a significant component of the operating cost), the success rate and the utilization of the servers in the first two experimental scenarios which represents the impact of the amount of resources (number of servers) and the prediction errors. For the electricity cost, we compare ConBLF, ConBCA and RT with NF. The success rate represents the fraction of the successful tasks which excludes the number of failed tasks. The utilization is calculated as the average utilization of running servers during the evaluation time. The idle servers are not included in the calculation.

B. Experimental Results

For illustrating the performance of our contracts-based algorithms, we perform three sets of experiments: first, we study the impact of increasing the number of servers in the datacenters; second, we add different amount of errors to the prediction of the workloads and study its impact; finally, we evaluate the fairness of our algorithm compared to traditional approaches. For the first two experiments, we perform and analyze three metrics: the electricity consumption cost per successful task, the success rate and the average server utilization. For the fairness evaluation, we compare each CSP's profit which is normalized by the profit it can earn when running the service without cloud federation.

1) Impact of Number of Servers: We test the performance of our mechanisms with a various number of servers per datacenter first. The number of servers increases from 200 to 1000 per datacenter in the evaluation. As shown in Figure 3a, the y-axis is the normalized electricity cost per successful task compared with no federation. The x-axis is the number of servers per datacenter. We can get that with the ability to share the resources in the Cloud Federation, the electricity cost compared with no federation has been optimized from about 10% to 40% with increasing the number of servers. Our cost-aware mechanism (ConBCA) gets the result which is near the result of real-time complete cooperation mechanism (RT). Compared with ConBLF, ConBCA increases the utilization of the low-cost contracts which can potentially decrease the operating cost per successful task. As shown in Figure 3b, the



(a) Electricity Cost (b) Success Rate (c) Average Server Utilization

Fig. 4. Evaluation results for different average errors of the workload predictions

success rate increases with increasing the number of servers. The priority of sharing the resources are mainly reflected when the resources are scarce. The difference can be as much as more than 10%. If the resource is scarce, the result can be better. The RT mechanism gets the best result, which is about 4% better than our mechanism. As shown in Figure 3c, we can get that the contracts-based mechanisms (ConBLF and ConBCA) and the RT mechanism get better results than NF, because the sharing mechanisms make the workload more balanced among the datacenters. The RT mechanism is worse than our contracts-based mechanisms, because in our mechanism, the prediction of the workload takes effect in the contracts, so the utilization of the resource in the contracts are increased so as the overall average utilization.

When ConBLF and ConBCA schemes are compared, we can see that ConBLF performs better as it uses the local resources which is more available than the resources in the contract. Here, larger number of resources increases the possibility to get a better match in the first-fit provisioning mechanism.

From the above experiments with increasing number of servers, we can see that contracts-based algorithms perform significantly better than NF in terms of operating cost, the success rate and server utilization. The performance is close to RT in the three observations.

2) Impact of Prediction Errors: We then test the performance of our mechanisms with prediction error which is added to the workload demand prediction. The distribution of the added error is white noise. The average amount of error is increased from 10% to 50%. As shown in Figure 4a, the y-axis is the same as the previous evaluation. The x-axis is the number of prediction errors. We can get that with the ability to share the resources in the federated cloud, the electricity cost compared with NF has been optimized from about 18% to 20% regardless of the prediction errors. The result shows that the prediction errors do not influence the result significantly (2% with 50% added error) because the error only influences the predicted volume of the workload. As the resource is traded between the providers with the true value, the error in the predicted volume does not influence the true value evaluation in the bid. As shown in Figure 4b, the success rate is also not influenced significantly by the prediction error. We also notice in Figure 4c that the influence on utilization is not visibly significant as well. From the above experiments with different prediction errors, we can see that our contracts-based mechanisms perform better than NF overall.

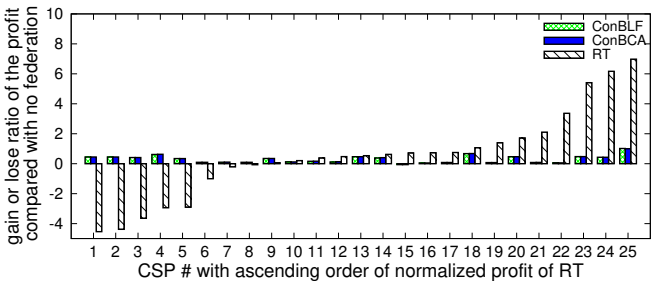


Fig. 5. The gain or lose ratio of the profit for each individual CSP

3) Fairness: In this set of experiments, we evaluate the fairness of the approaches by comparing the individual profit of each CSP with different mechanisms. The result is observed with the setup of 200 servers per datacenters. The other settings are the default. As shown in Figure 5, the y-axis is the gain or loss ratio of the normalized profit which is the difference between the profit which can be earned with federated cloud and the profit which can be earned without cloud federation. When the number is larger than 0, it means the provider earns more with cloud federation compared to not using cloud federation. The x-axis represents the index for each CSP which is sorted by the normalized profit in RT. From the figure, we can see that, when the federated cloud is operated by RT, there are eight CSPs (CSP1-8) of the total 25 CSPs losing profits. The mechanisms which are contracts-based perform better except for CSP15 which gets a little less (less than 8%) than the profit it can earn without cloud federation. From the observations above, we can see that the RT optimizes the operating cost but makes some of the CSPs lose profits. It demonstrates that our proposed contracts-based mechanisms perform better than the RT scheme with respect to fairness.

V. Related Work

Cloud Federation has gained significant focus from the cloud computing research community in the recent past. Most of the work related to cloud federation primarily focuses on two aspects. The first set of research efforts focus on the architecture and the system model for enabling and deploying federated clouds and the second class of existing works optimize the performance of federated cloud through efficient job scheduling, task migration and resource allocation. Rochwerger et al. [4] proposed an architecture which is called RESERVOIR to enable cloud providers to deal with each other in a P2P manner. Buyya et al. [5] proposed a centralized

architecture named InterCloud which provides a market for the CSPs or cloud brokers to share their resources. Carlini et al. [3] proposed a centralized architecture providing single sign-on for building the federated cloud.

Li et al. [26] proposed a model which makes it possible for a cloud federation to consider both the workload and the electricity price and maximize the profit through an auction mechanism. Xu et al. [6] proposed a technique that combines the alternating direction method of multipliers (ADMM) method with the problem of how to place 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 obtained by the provider, which is not applicable for generic federated clouds with CSPs having competing interests, such as the one considered in our work. To the best of our knowledge, the work proposed in this paper is the first research effort aimed at developing a contracts-based approach to allocating resources in a federated cloud environment. As shown in the experimental evaluation, the proposed model achieves a high degree of fairness compared to existing models while simultaneously achieving high performance in terms of success rate and resource utilization.

VI. Conclusions and Future Work

In this paper, we propose a contracts-based mechanism for resource sharing between CSPs in a federated cloud. We develop an auction-based mechanism for contract establishment and a cost-aware scheduling technique that maximizes the local profits of the CSPs while meeting the individual job requirements. We evaluate 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 while achieving similar operational costs and success rate properties as existing methods.

In the future, we plan to address a few limitations of our current work. First, in addition to computationally intensive tasks considered in this work, we plan to consider data intensive workloads in the resource allocation model. With data-intensive workloads, job migration across datacenters will become expensive. Such scenarios demand a migration cost-aware approach to workload offloading and scheduling over the core contracts establishment framework presented in this work. Another direction of our future work will focus on considering additional SLA requirements in the resource allocation framework including additional performance, reliability and fault-tolerant requirements.

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] K.-y. Chen, Y. Xu, K. Xi, and H. J. Chao, "Intelligent virtual machine placement for cost efficiency in geo-distributed cloud systems," in *Communications (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3498–3503.

[8] C. Jing, Y. Zhu, and M. Li, "Customer satisfaction-aware scheduling for utility maximization on geo-distributed data centers," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 5, pp. 1334–1354, 2015.

[9] 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.

[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] 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.

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

[13] N. Grid, "Large general tou," https://www.nationalgridus.com/niagaramohawk/business/rates/5_hour_charge.asp, accessed Jan. 4, 2016.

[14] V. Krishna, *Auction theory*. Academic press, 2009.

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

[16] Amazon, "Amazon ec2 dedicated instances," https://aws.amazon.com/ec2/purchasing-options/dedicated-instances/, accessed Nov. 11, 2016.

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

[18] 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.

[19] 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.

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

[21] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[22] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 5.

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

[24] 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.

[25] Google, "Google cluster's trace," https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md, accessed Jan. 4, 2016.

[26] H. Li, C. Wu, Z. Li, and F. C. Lau, "Profit-maximizing virtual machine trading in a federation of selfish clouds," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 25–29.