2012 32nd IEEE International Conference on Distributed Computing Systems

Dynamic Service Placement in Geographically Distributed Clouds

Qi Zhang*, Quanyan Zhu†, Mohamed Faten Zhani*, Raouf Boutaba*‡
*David R. Cheriton School of Computer Science, University of Waterloo, Canada.
Email: {q8zhang, mfzhani, rboutaba}@uwaterloo.ca
† Coordinated Science Laboratory and Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, USA. Email: zhu31@illinois.edu
‡ Division of IT Convergence Engineering, POSTECH, Pohang, KB 790-784, Korea

***Abstract*—Large-scale online service providers have been increasingly relying on geographically distributed cloud infrastructures for service hosting and delivery. In this context, a key challenge faced by service providers is to determine the locations where service applications should be placed such that the hosting cost is minimized while key performance requirements (e.g. response time) are assured. Furthermore, the dynamic nature of both demand pattern and infrastructure cost favors a dynamic solution to this problem. Currently most of the existing solutions for service placement have either ignored dynamics, or provided inadequate solutions that achieve both objectives at the same time. In this paper, we present a framework for dynamic service placement problems based on control- and game- theoretic models. In particular, we present a solution that optimizes the desired objective dynamically over time according to both demand and resource price fluctuations. We further consider the case where multiple service providers compete for resource in a dynamic manner, and show that there is a Nash equilibrium solution which is socially optimal. Using simulations based on realistic topologies, demand and resource prices, we demonstrate the effectiveness of our solution in realistic settings.**

I. Introduction

Cloud computing has recently gained significant popularity as a cost-effective model for delivering large-scale services over the Internet. In a Cloud computing environment, infrastructure providers (namely, cloud providers) build large data centers in geographically distributed locations to achieve reliability while minimizing operational cost [1]. The service providers (SPs), on the other hand, leverage geo-diversity of data centers to serve customers from multiple geographical regions. Today, large companies like Google, Yahoo and Microsoft have already adopted this model in their private clouds, offering a wide range of services to millions of users world-wide. As Cloud computing technologies become mature, an increasing number of companies are expected to adopt this model by moving into clouds.

A key technique of each SP in cloud service management is to distribute servers in multiple data centers in order to meet the performance requirements specified in Service Level Agreements (SLA), while reducing operational costs by optimizing the placement of servers in multiple data centers. This typically involves solving two problems jointly: (1) deciding on the number of servers placed in each data center, and (2) routing each request to appropriate servers to minimize response time. As infrastructure

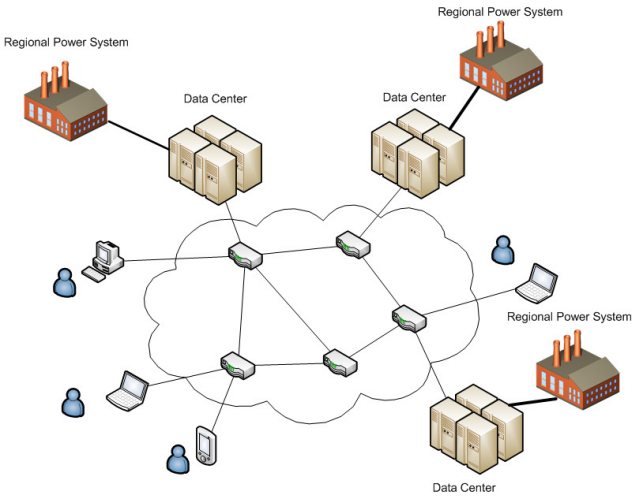

Figure 1. Model of service placement in geographically distributed data centers

providers typically offer on-demand and elastic resource access, it is possible to adjust the number of servers to match service demand in a dynamic way. Furthermore, the cost of reconfiguration (i.e. the cost of adding and removing servers) must be taken into account. The consideration for reconfiguration cost is important for ensuring the system stability and minimum management overhead and costs. For instance, these operations have costs for setup (e.g., VM image distribution) and tear-down (e.g., data / state transfer). Thus, it is in the interest of SPs to reduce such reconfiguration cost.

On the other hand, the price of resources offered by infrastructure providers are also subject to change. In particular, energy consumption is a major contributor to the operation cost of a data center. In many regions of the U.S., the electricity grid of each region is managed independently by a Regional Transmission Organization (RTO) which operates wholesales electricity markets in order to match supply and demand for electricity, as illustrated in Figure 1. As a result, electricity prices in each region can vary independently over time. Based on this fact, recently there have been several studies on dynamic server placement [2], [3] and request dispatching [4] in private clouds, taking into account fluctuating energy costs. The same benefit can be achieved in public clouds by introducing some degree of dynamic pricing, such as the one being used by Amazon EC2 [5].

1063-6927/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDCS.2012.74

Combining the above observations, a SP is facing the problem of dynamically controlling the number of servers placed in each data center to minimize the total resource cost while satisfying SLA requirements, taking into consideration the fluctuation of both demand and resource price. We call this problem *dynamic service placement problem (DSPP)*. This problem shares many similarities with traditional replica placement problem in [6], [7], [8]; however, the price fluctuation is often neglected in the existing literature. Recently, there are several papers that have studied this problem (e.g., [3]). However, the dynamic aspect of the problem, particularly the cost of dynamically starting and shutting down servers, is still largely unaddressed.

In this paper, we study the DSPP problem using both control and game theoretic methods. Specifically, we first propose a control framework based on Model Predictive Control (MPC) approach to provide an online adaptive control mechanism which aims at reducing service provider costs, namely, resource allocation and reconfiguration costs. We further extend this framework to a game-theoretic model to consider the competition among multiple SPs, taking into consideration the capacity constraint of each data center. This model is realistic for several reasons: (1) on-demand resource allocation mechanisms can often lead to situations where resource demand exceeds the capacity available in a data center (e.g., during holiday seasons). (2) Recently, there are numerous proposals that advocate for small-scale data centers (e.g., [9], [10]). In both cases, limited data center capacity can result in some SPs not getting the resources they desire. In this case, we analyze the outcome of resource competition and show that there exists an optimal outcome, and provide algorithms to compute this outcome. Finally, using simulations based on realistic topologies, demand and prices, we demonstrate the effectiveness of our proposed approach and analyze various properties of the resource competition game.

The remainder of the paper is organized as follows: Section II surveys the related work. Section III presents the proposed framework for a single SP. Section IV describes the problem formulation of DSPP for a single SP. The design of our controller for DSPP is provided in Section V. In Section VI, we extend our framework to a multi-provider scenario and analyze the outcome of the resource competition game. Section VII presents our experimental results. Finally, Section VIII concludes the paper.

II. Related Work

Service placement in large-scale shared service hosting infrastructures has been studied in many contexts in the past. Early works on this problem primarily have focused on placing content replicas in the context of content delivery networks (CDNs) [11], [12], [8]. However, most of the work in that context have addressed the centralized cases where the demand profile and network topology are static or time invariant. More recent studies have also investigated dynamic cases [7], [13] where iterative improvement algorithms are proposed. In the context of application placement in distributed systems, Oppenheimer et. al. [14] have studied the problem of placing applications on PlanetLab. As resource usage can be heterogenous and time-varying among the PlanetLab nodes. They have suggested that dynamically adjusting service placement can potentially improve the system performance. Laoutaris et. al. have formulated the service placement as an uncapacitated facility location problem (UFLP) [15], and have presented a local search heuristic algorithm for improving the quality of service placement solutions over time. However, the objective of these studies is to ensure the algorithm converges to a near optimal solution for a static topology in finite number of iterations, instead of optimizing the overall system performance in the presence of demand and resource dynamics. Furthermore, the cost of dynamic reconfiguration is not considered in these studies. More recently, Arora et. al. [16] have presented an approximation algorithm for the DSPP in the context of virtual networks, however, the approximation guarantee is inadequate to guarantee the quality of the solution in general cases.

With the growth of large-scale data center infrastructures, energy consumption has recently become an acute problem. Not only does energy consumption account for a significant fraction of data center operating cost, it also raises concerns regarding environmental impact and sustainability of these infrastructures. As a result, there is much effort in the research community to improve energy efficiency of data centers (e.g., [17], [18]) Driven by the fact that electricity grids are independently managed in different geographical regions, several recent works have started exploiting geo-diversity for minimizing energy costs. For instance, Qureshi et. al. [4] have provided a detailed analysis of regional electricity markets and have shown that energy-aware request routing can achieve significant cost savings for large CDNs. Following this line of research, Rao et. al. [2] have studied the problem of server placement in a multi-electricity market environments with the goal of minimizing electricity cost. More recently, Liu et. al. [3] have presented a distributed solution for the same problem, taking into consideration both request response time and energy cost. However, both of them have only studied static cases. As both service and resource price can change independently over time, it is necessary to find a solution that dynamically adjusts the placement strategies to optimize both service performance and cost, while minimizing the overhead of reconfiguration.

Lastly, the application of control theory to capacity provisioning in compute clusters has been extensively studied, primarily in the context of autonomic computing. Earlier work in this area primarily focused on performance objectives, specifically, dynamic control of the number of server replicas to meet a specific performance criterion. For example, Diao et. al. [19] have studied the problem of dynamically adjusting memory pool sizes for multiple agents

Table I
TABLE OF NOTATIONS

Symbol	Meaning
x_k^l	Num. of servers at DC l at time k
x_k^{lv}	Num. of servers at l serving demand from v at time k
D_k^v	Avg. demand arrival rate originated from v
σ_k^{lv}	Avg. arrival rate of demand from v to DC l at time k
u_k^{lv}	Change in the number of servers at DC l at time k
λ_k^{lv}	Avg. arrival rate to each server from v to l at time k
d_{lv}	Network latency between location v and data center l
μ	Request process rate of a single server
p^l	Price of each server at DC l
r	Reservation ratio
C^l	Capacity of DC l
H_k	Resource allocation cost at time k
G_k	Reconfiguration cost at time k
J	Total operational cost

in a database server with the goal of minimizing worst-case response time. More recently, energy consumption has become a consideration in these studies. For instance, Kusic et. al. [20] have presented a control-theoretic framework for reducing energy consumption while satisfying SLA constraints. However, most of existing solutions only apply to intra-data center environments (i.e. inside a data center), while the impact of geographical location have not been considered in these studies.

III. SYSTEM ARCHITECTURE AND DESIGN

We consider a multi-regional cloud environment that consists of multiple data centers situated at different geographical locations. Our system architecture consists of 4 components as depicted in Figure 2: (1) request routers, (2) monitoring module, (3) analysis and prediction module, and (4) the resource controller. Both the request router and the monitoring module can be directly owned by the SP, or leased from other service providers who offer them as services. In particular, the service provider controls *request routers* (a.k.a. redirectors) which are responsible for redirecting the requests to the appropriate servers [21], [22]. In practice, request redirection can play a key role in improving server accessibility through load balancing, latency minimization and content replication. For instance, Amazon EC2 Elastic Load Balancing service [23] is an example of a simplified request router. More sophisticated designs (e.g. [22]) have also been studied in the literature. The *monitoring module* is responsible for collecting statistics, including the amount of requests received (i.e. the demand) at the different request routers and the prices offered by each data center. The *analysis and prediction module* models the dynamics of demand and price fluctuations, and forecasts the future values of both demand and resource prices. In practice, it has been shown that both demand and price in production data centers generally show daily fluctuation patterns [3], [4]. In this case, the demand can be reasonably predicted using historical traces. However, there are occasions where both demand and resource price can behave in an unexpectedly manner, e.g., flash-crowd effect or system failure. Alternative prediction models such as autoregressive (AR) models [24], and demand characterization models [25] may be used. It is important to point out that our control-theoretic model is generic and can work with any demand prediction techniques.

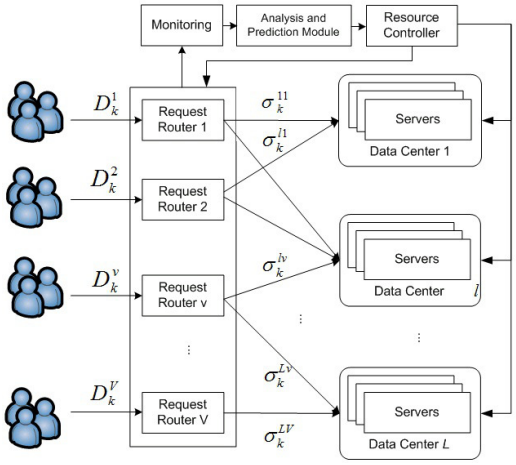

Figure 2. System architecture for a single service provider

Finally, the *resource controller* is responsible for solving the DSPP and making online control decisions at run time. It dynamically adjusts the number of servers leased in each data center in order to satisfy the SLA requirements (in terms of latency), while minimizing the resource rental cost. Furthermore, it informs the request routers about the number of servers allocated in each data center. The request routers must then find appropriate assignment of demand to the allocated servers. In our system architecture, request router adopts a simple strategy which is to split demand proportionally among the servers that satisfy the SLA requirements. We will formally define the demand our assignment model in Section 13. In the next section, we will first provide a mathematical model of DSPP, and then use the model to describe the request assignment policy employed by each request router. The solution to DSPP will then be described in the subsequent sections.

IV. PROBLEM FORMULATION

We model the network as a bipartite graph $G = (L \cup V, E)$, where L denotes the set of data centers, V denotes the location of customers. For instance, V can be the set of access networks which customers are connected to. Denote by $E \subseteq L \times V$ the communication paths between customers and data centers. We also assign constant weights d_{lv} to denote the network latency between a data center location $l \in L$ and a client location $v \in V$.

In our framework, we consider a discrete-time system model where time is divided into multiple time periods corresponding to the timescale at which server placement and routing decisions are made. We assume that there is an interval of interest $\mathcal{K} = \{0, 1, 2, ..., K\}$ that consists of

$K+1$ periods. Let $N = \{1, 2, ..., n\}$ denote the set of SPs. We assume that at time $k \in \mathcal{K}$, each customer location $v \in V$ has demand D_k^v in terms of average arrival rate of requests from location v at time k. We assume that D_k^v is real valued since each request is typically small compared to the aggregated demand.

For simplicity, we assume that all the servers leased by each SP have identical size and functionality. For instance, a server can be a virtual machine (VM) that runs a specific application image. We define the state variable $x_k^l \in \mathbb{R}_+$ as the number of servers owned by the SP at location $l \in L$ at time k. To simplify the model, we assume that x_k^l can take continuous values rather than discrete values. This assumption is reasonable for large-scale services that require tens or hundreds of servers, where the weight of each individual server in the overall solution is small. In this case, we can always obtain a feasible solution by rounding up the continuous values to the nearest integer values. Based on this assumption, we can further decouple x_k^l by defining $x_k^{lv} \in \mathbb{R}_+$ as the number of servers at location l serving certain amount of demand from $v \in V$:

$$x_k^l = \sum_{v \in V} x_k^{lv}, \qquad \forall l \in L, 0 \leq k \leq K. \tag{1}$$

Let $u_k^{lv} \in \mathbb{R}$ denote the change in the number of servers in x_k^{lv} at time k. Therefore, the state equation for x_k^{lv} is:

$$x_{k+1}^{lv} = x_k^{lv} + u_k^{lv}, \qquad \forall l \in L, v \in V, 0 \leq k \leq K. \tag{2}$$

A. Modeling the Server Allocation and Reconfiguration Cost

To model the cost of server allocation, We assume that there is a price p_k^l for running a server at data center $l \in L$ at time k. The total resource cost H_k for service hosting at time k is

$$H_k = \sum_{l \in L} x_k^l p_k^l = \sum_{l \in L} \sum_{v \in V} x_k^{lv} p_k^l, \qquad \forall 0 \leq k \leq K \tag{3}$$

We also assume that there is a reconfiguration cost function $S : \mathbb{R} \to \mathbb{R}_+$ that computes the cost of reconfiguration. Specifically, we want to penalize rapid change in system state, as it often leads to system instability and to additional costs (costs of setup/release of resources). A typical reconfiguration cost that satisfies our goal is a quadratic function $S(u_k^{lv}) = c^l (u_k^{lv})^2$, where $c^l \in \mathbb{R}_+$ is a predefined constant [26]. Quadratic penalty functions are widely used in control theory literature, as it penalizes rapid reconfiguration of system states. In this case, the total reconfiguration cost is given by:

$$G_k = \sum_{l \in L} \sum_{v \in V} S(u_k^{lv}) = \sum_{l \in L} \sum_{v \in V} c^l (u_k^{lv})^2, \forall\ 0 \leq k \leq K. \tag{4}$$

B. Modeling the Constraints

While minimizing the total operational cost, the allocation of servers and demand assignment must satisfy a set of constraints, including (1) demand constraint, (2) data center capacity constraint and (3) SLA performance constraint. Define σ_k^{lv} as the demand arrival rate from v assigned to data center l at time k, the demand constraint ensures that all demands are satisfied, i.e.,

$$\sum_{l \in L} \sigma_k^{lv} = D_k^v, \qquad \forall v \in V,\ l \in L,\ 0 \leq k \leq K. \tag{5}$$

The data center capacity constraint specifies an upper-bound C^l the denotes the number of servers that can be allocated in each data center $l \in L$.

$$\sum_{v \in V} x_k^{lv} \leq C^l, \qquad \forall l \in L,\ 0 \leq k \leq K. \tag{6}$$

Finally, there is a SLA performance constraint that specifies a maximum latency $\bar{d}_{lv}$ that the SP tries to achieve between a location v and a data center l. We focus on modeling this constraint in the rest of the this subsection.

For data center $l \in L$, we assume that the demand σ_k^{lv} arriving from location v is equally split among the local servers x_k^{lv}. Let $\lambda = \frac{\sigma_k^{lv}}{x_k^{lv}}$ denote the arrival rate of requests for each server. For model simplicity, we model the queuing delay using the standard $M/M/1$ queueing model. But we believe it is straightforward to adapt our framework to other queueing models as well. In our case, The queueing delay for a demand at location $v \in V$ to a server at $l \in L$ can be computed as:

$$q(x_k^{lv}, \sigma_k^{lv}) = \frac{1}{\mu - \lambda} = \frac{1}{\mu - \frac{\sigma_k^{lv}}{x_k^{lv}}}. \tag{7}$$

where μ is the service rate of each server. For a request from location $v \in V$ to server at $l \in L$, we want to ensure that for any $(v, l) \in E$ with $\sigma_k^{lv} > 0$, the average delay is upper-bounded by $\bar{d}_{lv}$:

$$d_{lv} + q(x_k^{lv}, \sigma_k^{lv}) \leq \bar{d}_{lv}, \quad \forall v \in V, l \in L, 0 \leq k \leq K. \tag{8}$$

The constraint (8) can be rewritten as

$$x_k^{lv} \geq \frac{\sigma_k^{lv}}{\mu - \frac{1}{\bar{d}_{lv} - d_{lv}}}, \quad \forall v \in V, l \in L, 0 \leq k \leq K. \tag{9}$$

By defining

$$a^{lv} = \begin{cases} \frac{1}{\mu - (\bar{d}_{lv} - d_{lv})^{-1}}, & \text{if } \bar{d}_{lv} - d_{lv} > 0, \\ \infty, & \text{otherwise}, \end{cases} \tag{10}$$

as a known constant, we can rewrite the constraint (9) as:

$$x_k^{lv} \geq a^{lv} \sigma_k^{lv}, \qquad \forall v \in V, l \in L. \tag{11}$$

We would like to point out that even though our model focuses on bounding the average delay, it is straightforward to extend it to handle more general cases, such as ϕ-percentile delay (where ϕ is typically 95%) by multiplying

$q(x_k^{lv}, \sigma_k^{lv})$ by a constant factor $\ln\left(\frac{1}{1-\phi}\right)$. Furthermore, it is straightforward to extend our model to consider additional requirements and constraints. For example, some SPs may wish to over-provision capacities (e.g., capacity cushion) to account for temporary demand fluctuation. This factor can be modeled by introducing a variable $r \in \mathbb{R}_+$ to represent the ratio between the actual number of servers and the number of servers required to satisfy the SLA performance constraint, taking into consideration the over-provisioning of capacities. In this case, we only need to change the definition of a^{lv} to $a^{lv} = \frac{1}{\mu - (d_{lv} - d_{lv})^{-1}}$ for the case $d_{lv} - d_{lv} > 0$.

C. Modeling the Demand Assignment

We can combine constraints (5) and (11) to eliminate demand assignment variable σ_k^{lv} as follows:

$$\sum_{l \in L} \frac{x_k^{lv}}{a^{lv}} \geq D_k^v, \qquad \forall v \in V, 0 \leq k \leq K. \tag{12}$$

If (12) is satisfied, we can define the demand assignment policy that specifies σ_k^{lv} for each request router as:

$$\sigma_k^{lv} = D_k^v \cdot \frac{\frac{x_k^{lv}}{a^{lv}}}{\sum_{l \in L} \frac{x_k^{lv}}{a^{lv}}}. \tag{13}$$

This ensures that SLA requirement is met by all request routers. In practice, each request router for location $v \in V$ can implement the policy by splitting the demand D_k^v proportionally according to $\frac{\frac{x_k^{lv}}{a^{lv}}}{\sum_{l \in L} \frac{x_k^{lv}}{a^{lv}}}$ for each $l \in L$ using any standard load balancing techniques.

D. DSPP formulation

Given the system model described above, the goal of DSPP is to minimize the total cost of server allocation and reconfiguration cost. Based on (3) and (4), the goal of DSPP is to minimize the following objective function:

$$J := \sum_{k=0}^{K} H_k + G_k = \sum_{k=0}^{K} \sum_{v \in V} \sum_{l \in L} x_k^{lv} p_k^{lv} + c^l (u_k^{lv})^2.$$

Formally, DSPP can be represented as the following optimization problem:

$$\begin{aligned}
\min \quad & \sum_{k=0}^{K} \sum_{v \in V} \sum_{l \in L} x_k^{lv} p_k^{lv} + c^l (u_k^{lv})^2 & \\
\text{s.t.} \quad & \sum_{l \in L} \frac{x_k^{lv}}{a^{lv}} \geq D_k^v, & \forall v \in V, 0 \leq k \leq K \\
& x_{k+1}^{lv} = x_k^{lv} + u_k^{lv}, & \forall l \in L, v \in V, 0 \leq k < K \\
& \sum_{v \in V} x_k^{lv} \leq C^l, & \forall l \in L, k \in K, \\
& x_k^{lv} \in \mathbb{R}_+, u_k^{lv} \in \mathbb{R}, & \forall l \in L, v \in V, 0 \leq k \leq K
\end{aligned}$$

Define $\mathbf{x}_k = [x_k^{11}, ...x^{L1}, ..., x_k^{lv}, ..., x_k^{LV}]^\top \in \mathbb{R}_+^{LV}$, $\mathbf{p}_k^{'} = [p_k^1, p_k^2, ..., p_k^L] \in \mathbb{R}_+^L$, $\mathbf{p}_k = [\mathbf{p}_k^{'}, \mathbf{p}_k^{'}, ...\mathbf{p}_k^{'}]^\top \in \mathbb{R}_+^{LV}$, $\mathbf{u}_k = [u_k^{11}, ...u^{L1}, ..., u_k^{lv}, ..., u_k^{LV}]^\top \in \mathbb{R}^{LV}$, $\mathbf{a}_k^v = [\frac{1}{a_k^{1v}}, \frac{1}{a_k^{2v}}, ..., \frac{1}{a_k^{Lv}}]^\top$, $\mathbf{a}_k = \text{diag}^{-1}\{\mathbf{a}_k^1, ..., \mathbf{a}_k^V\} \in \mathbb{R}_+^{LV \times V}$, $\mathbf{R}^{'} = [c^1, c^2, ..., c^L] \in \mathbb{R}_+^L$, $\mathbf{R} = \text{diag}\{[\mathbf{R}^{'}, \mathbf{R}^{'}, ..., \mathbf{R}^{'}]\} \in \mathbb{R}_+^{LV \times LV}$, $\mathbf{D}_k = [D_k^1, ..., D_k^V]^\top$, $\mathbf{C} = [C^1, C^2, ...C^L]^\top \in \mathbb{R}_+^L$, $\mathbf{s} = [\mathbf{I}^{L \times L}, ..\mathbf{I}^{L \times L}]^\top \in \mathbb{R}_+^{LV \times L}$, we can rewrite DSPP as follows:

$$\begin{aligned}
\min_{\{\mathbf{u}_0, .., \mathbf{u}_{K-1}\}} \quad & J = \sum_{k=0}^{K} \mathbf{p}_k^\top \mathbf{x}_k + \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k & \\
\text{s.t.} \quad & \mathbf{a}_k^\top \mathbf{x}_k \geq \mathbf{D}_k, & \forall 0 \leq k \leq K, \\
& \mathbf{s}^\top \mathbf{x}_k \leq \mathbf{C}, & \forall 0 \leq k \leq K, \\
& \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k, & \forall 0 \leq k \leq K-1, \\
& \mathbf{x}_k \in \mathbb{R}_+^{LV}, \mathbf{u}_k \in \mathbb{R}^{LV}, & \forall 0 \leq k \leq K-1.
\end{aligned}$$

This problem is a linear-quadratic program that can be solved optimally using standard methods [27].

Algorithm 1 MPC Algorithm for DSPP

1: Provide initial state $\mathbf{x}_0$, $k \leftarrow 0$
2: **loop**
3: At beginning of control period k:
4: Predict $\mathbf{D}_{k+i|k}^l$ for horizons $i = 1, \cdots, K$ using a demand prediction model
5: Solve DSPP to obtain $\mathbf{u}_{k+t|k}$ for $t = 0, \cdots, W-1$
6: Change the resource allocation according to $\mathbf{u}_{k|k}$
7: Update demand assignment policy of request routers according to equation (13)
8: $k \leftarrow k+1$
9: **end loop**

V. Controller design for DSPP

Even though DSPP can be solved optimally, in practice, the resource controller must solve this problem in an online setting where future demand is unknown. In this case, we use the Model Predictive Control (MPC) framework that is widely used for solving online control problems. Algorithm 1 is our MPC algorithm used by the resource controller for solving DSPP online. It can be described as follows. At time k, the resource controller predicts the future demand $\mathbf{D}_k^v$ for multiple periods $[k+1, ..., k+W]$. using the demand predicted by the analysis and prediction module, where W is the prediction horizon. Denote by $\mathbf{D}_{k+t|k}^v$ the demand predicted for time $k+t$ at time k. The controller then solves the optimization problem for the horizon $[k, ..., k+W]$, starting with the initial state $\mathbf{x}_{k|k}^{lv} = \mathbf{x}_k^{lv}$. Even though the solution of the optimization problem will contain a set of values $\mathbf{u}_{k|k}^{lv}, ..., \mathbf{u}_{k+W-1|k}^{lv}$ The controller will only execute the first step in sequence $\mathbf{u}_{k|k}^{lv}$. When the next control period $k+1$ starts, the same procedure is performed again by the controller. Using this MPC algorithm, the controller can effectively adjust the number of servers in each data center.

VI. Competition Among Multiple Providers

In this section, we extend our previous model and consider the case where multiple SPs share the cloud platform in terms of resources in data centers. The goal of each SP is to minimize its operational costs with respect to the SLA performance requirements and the data center capacity constraints. In our model, we assume that the placement configuration of each SP is kept private from other SPs. In this scenario, strategic interactions may arise as each SP makes decisions independently. Therefore, we can model the system as a multi-person non-cooperative game. Our objective is to analyze the equilibrium outcome of this game and discuss its properties.

One of the key challenges in defining the resource competition game is the modeling of the data center capacity constraints. As there are multiple SPs using the resources in data centers, it is necessary to ensure that total capacity constraint of each data center is not violated. To make our analysis tractable, we assume that this capacity is *exact*, namely, the resources in each data center can be optimally allocated to servers without loss in wastage. Even though this assumption may not hold in general (Optimally scheduling virtualized servers with heterogenous resource requirements generalizes the $\mathcal{NP}$-hard bin-packing problem), we believe it is a reasonable approximation of the real world scenario, and is sufficient to illustrate the characteristics of the resource competition game. Furthermore, cloud infrastructure providers typically design VM sizes to match physical machine capacities to reduce resource wastage. An example is the GoGrid public cloud service, which offers VMs in 6 different types. Arranged from the smallest to the largest, each type of VMs has exactly twice the size of previous type in terms of CPU, memory and disk capacity. When VM sizes are multiples of each other, bin-packing can be solved optimally using First-Fit-Decrease (FFD) policy [28], and no resource is wasted during the process. In this case, our competition model can be applied exactly to the case of GoGrid.

A. Problem Formulation

We formally define the resource competition game in this section. For each $v \in V$, define $\mathcal{N} = \{1, 2, ..., N\}$ as the set of SPs, and let $i \in \mathcal{N}$ represent the index of each SP. Let $\mathcal{K} = \{0, 1, 2, ...K\}$ denote the set of stages (i.e. time indices) of the game. At time k, $0 \leq k \leq K$, each SP has a state $\mathbf{x}_k^{iv} = [x_k^{i1v}, ..., x^{iLv}]^\top \in \mathbb{R}_+^L$ that describes the number of servers allocated to demand from $v \in V$ at data center $l \in L$. Each SP i also makes a control decision $\mathbf{u}_k^{iv} = [u_k^{i1v}, ..., u^{iLv}]^\top \in \mathbb{R}^L$ at time $k \in \mathcal{K}$, where u_k^{ilv} denotes the change in the number of servers serving $v \in V$ at data center $l \in L$ at time k. Given an initial system state $\mathbf{x}_0^i$, the system dynamics for each SP is captured by the following state equation:

$$\mathbf{x}_{k+1}^{iv} = \mathbf{x}_k^{iv} + \mathbf{u}_k^{iv} \quad \forall i \in \mathcal{N}, v \in V, 0 \leq k \leq K-1, \quad (14)$$

At time $k \in \mathcal{K}$, we assume each SP i has a demand $\mathbf{D}_k^i = [D_k^{i1}, D_k^{i2}, ..., D_k^{iv}]^\top$, where D_k^{iv} represents the demand for SP i originated from $v \in V$ at time $k \in \mathcal{K}$. The total resource allocation must satisfy the demand constraint, which states that the total resource allocation must be sufficient to handle all demands without violating the SLA policy. This demand constraint can be formally stated as:

$$\mathbf{a}_k^{i\top} \mathbf{x}_k^{iv} \geq D_k^{iv}, \quad \forall i \in \mathcal{N},\ v \in V,\ 0 \leq k \leq K, \quad (15)$$

where $\mathbf{a}_k^i$ is defined as in Section IV-D for each SP i. We also require the total allocated resources to satisfy the data center capacity constraint:

$$\sum_{i \in \mathcal{N}} \sum_{v \in V} x_k^{ilv} s^i \leq C^l, \ \ \forall i \in \mathcal{N},\ l \in L,\ 0 \leq k \leq K, \quad (16)$$

where $s^i \in \mathbb{R}_+$ is the "size" of a server owned by SP i in terms of resource requirement, such as CPU or memory. Define $\mathbf{s}^i = s^i \cdot \mathbf{I}^{L \times L} \in \mathbb{R}_+^{L \times L}$ and $\mathbf{C} = [C^1, C^2, ...C^L]^\top \in \mathbb{R}_+^L$, we can rewrite equation (16) in vector form as follows:

$$\sum_{i \in \mathcal{N}} \sum_{v \in V} \mathbf{s}^i \mathbf{x}_k^{iv} \leq \mathbf{C}, \quad \forall l \in L, 0 \leq k \leq K. \quad (17)$$

Lastly, we define $\mathbf{u}^{iv} = \{\mathbf{u}_0^{iv}, \mathbf{u}_1^{iv}, ..., \mathbf{u}_{K-1}^{iv}\}$ $\mathbf{x}^{iv} = \{\mathbf{x}_0^{iv}, \mathbf{x}_1^{iv}, ..., \mathbf{x}_{K-1}^{iv}\}$, $\mathbf{u}^i = \{\mathbf{u}^{i1}, \mathbf{u}_1^{i2}, ..., \mathbf{u}_{K-1}^{iV}\}$. Furthermore, let $\mathbf{u}^{-i} = \{\mathbf{u}^1, ..., \mathbf{u}^{(i-1)}, \mathbf{u}^{(i+1)}, ..., \mathbf{u}^N\}$ represent the control decisions of the other SPs, the objective of SP i is to minimize its cost function:

$$J^i(\mathbf{u}^i, \mathbf{u}^{-i}) = \sum_{k=0}^{K} \sum_{v \in V} \mathbf{p}_k \mathbf{x}_k^{iv} + \mathbf{u}_k^{iv\top} \mathbf{R}^i \mathbf{u}_k^{iv}$$

$$\begin{aligned}
\text{s.t.} \quad & \mathbf{a}_k^{i\top} \mathbf{x}_k^{iv} \geq D_k^{iv}, && \forall i \in \mathcal{N}, v \in V, 0 \leq k \leq K, \\
& \sum_{i \in \mathcal{N}} \sum_{v \in V} \mathbf{s}^i \mathbf{x}_k^{iv} \leq \mathbf{C}, && 0 \leq k \leq K, \\
& \mathbf{x}_{k+1}^{iv} = \mathbf{x}_k^{iv} + \mathbf{u}_k^{iv}, && \forall i \in \mathcal{N}, v \in V, 0 \leq k \leq K-1, \\
& \mathbf{x}_k^{iv} \in \mathbb{R}_+^L \mathbf{u}_k^{iv} \in \mathbb{R}^L, && \forall i \in \mathcal{N}, v \in V, 0 \leq k \leq K.
\end{aligned}$$

where $\mathbf{R}^i = [c^{i1}, c^{i2}, ..., c^{iL}] \in \mathbb{R}_+^L$ which captures the weight factor for reconfiguration cost for SP i in every data center $l \in L$.

B. Resource Competition Game Analysis

In this section, we characterize the Nash equilibrium (NE) of the resource competition game. The Nash equilibrium refers to the stable outcome of the competition, where no SP can improve its cost by unilaterally changing the its server allocation over time. Formally, the resource competition game can be represented as a N-player dynamic non-cooperative game Ξ. Notice that as our controller relies on the MPC framework for dynamic resource allocation, we need to introduce a new version of Nash equilibrium for control strategies using the MPC framework. We first start with the following general definitions:

Definition 1 (η-Nash Equilibrium [29]). *Let $\mathcal{I}_k^i$ be the information set of a SP i at time k under a given information*

structure η^i, and Γ^i is the set of all admissible policies of SP i under η^i. The policy $\{\gamma^{i*}, i \in \mathcal{N}\}$ is an $\eta-$Nash equilibrium of the game Ξ, where $\mathbf{u}^i = \gamma^{i*}(\mathcal{I}_k^i)$ and $\eta = \{\eta^i, i \in \mathcal{N}\}$ if $J^i(\gamma^{i*}, \gamma^{-i*}) \leq J^i(\gamma^i, \gamma^{-i*})$, for all admissible policies $\gamma^i \in \Gamma^i$ and for all $i \in \mathcal{N}$, where $\gamma^{-i*} = \{\gamma^j, j \neq i, j \in \mathcal{N}\}$.

Definition 1 provides a general description of NE under a given information structure (IS) η^i. The dynamic game Ξ can admit different NEs under different information structures η. Typical information structures are, for example, open-loop IS, where the policy is only dependent on the initial conditions, and the perfect-state feedback IS, where the policy depends on the perfect measurement of the system state. The IS under MPC algorithms in Algorithm 1 can be deemed a special mixture between open-loop IS and feedback IS since at each stage each SP computes within a window in an open-loop manner but the initial condition of the computation is the current state known to SPs. With this special IS, we can define NE under MPC-type computations for our resource competition game.

Definition 2 ($\mathbf{W}-$MPC Nash Equilibrium)**.** *Let W^i be the prediction window of SP i and every SP adopts MPC as outlined in Algorithm 1. The dynamic non-cooperative game Ξ admits $\mathbf{W}-$MPC Nash Equilibrium, $\mathbf{W} = \{W^i, i \in \mathcal{N}\}$, if the sequences $\mathbf{u}^{iv*} := \{\mathbf{u}_k^{iv*}, 0 \leq k \leq K\}$ obtained under MPC algorithms satisfy $J^i(\mathbf{u}^{i*}, \mathbf{u}^{-i*}) \leq J^i(\mathbf{u}^i, \mathbf{u}^{-i*})$, for all admissible sequences $\mathbf{u}^i \in \mathcal{U}^i$ and for all $i \in \mathcal{N}$, where $\mathcal{U}^i$ is the set of admissible control sequences under MPC algorithms, and $\mathbf{u}^{-i*} = \{\mathbf{u}^j, j \neq i, j \in \mathcal{N}\}$.*

Note that Nash equilibrium solutions may not be unique, and hence we let $\mathcal{U}^*$ to denote the set of Nash equilibrium solutions $\mathbf{u}^* := \{\mathbf{u}^i, \mathbf{u}^{-i}\}$ that satisfy Definition 2. The $\mathbf{W}-$MPC Nash equilibrium $\{\mathbf{u}^{i*}, i \in \mathcal{N}\}$ can be used to compare with the optimal MPC solution $\{\mathbf{u}^{i\circ}, i \in \mathcal{N}\}$ to the social welfare problem (SWP), which is described as follows.

$$
\begin{aligned}
\min_{\{\mathbf{u}^1,\ldots,\mathbf{u}^N\}} \quad & \sum_{i\in\mathcal{N}} J^i(\mathbf{u}^1, ..., \mathbf{u}^N) \\
\text{s.t.} \quad & \mathbf{a}_k^{i\top}\mathbf{x}_k^{iv} \geq D_k^{iv}, && \forall i \in \mathcal{N}, v \in V, 0 \leq k \leq K \\
& \sum_{i\in\mathcal{N}}\sum_{v\in V} \mathbf{s}^i \mathbf{x}_k^{iv} \leq \mathbf{C}, && 0 \leq k \leq K, \\
& \mathbf{x}_{k+1}^{iv} = \mathbf{x}_k^{iv} + \mathbf{u}_k^{iv}, && \forall i \in \mathcal{N}, v \in V, 0 \leq k < K, \\
& \mathbf{x}_k^{iv} \in \mathbb{R}_+^L \mathbf{u}_k^{iv} \in \mathbb{R}^L, && \forall i \in \mathcal{N}, v \in V, 0 \leq k \leq K.
\end{aligned}
\tag{SWP}
$$

The optimal MPC solution to (SWP) can be compared with the $\mathbf{W}-$MPC Nash equilibrium using price of anarchy (PoA) and price of stability (PoS), [30], [31], defined as follows.

Definition 3. *The price of anarchy (PoA) ρ_{MPC} and the price of stability (PoS) ξ_{MPC} of the dynamic non-cooperative game Ξ under centralized MPC Algorithm 1 and distributed MPC Algorithm 2 are defined by*

$$\rho_{MPC} := \inf_{\mathbf{u}^*\in\mathcal{U}^*} \frac{\sum_{i\in\mathcal{N}}\sum_{v\in V} J_v^i(\mathbf{u}^{i\circ})}{\sum_{i\in\mathcal{N}}\sum_{v\in V} J_v^i(\mathbf{u}^{i*})}, \tag{18}$$

$$\xi_{MPC} = \sup_{\mathbf{u}^*\in\mathcal{U}^*} \frac{\sum_{i\in\mathcal{N}}\sum_{v\in V} J_v^i(\mathbf{u}^{i\circ})}{\sum_{i\in\mathcal{N}}\sum_{v\in V} J_v^i(\mathbf{u}^{i*})}, \tag{19}$$

where $\{\mathbf{u}^{i\circ}, i \in \mathcal{N}\}$ is the optimal solution to (SWP) obtained by MPC algorithm 1, and $\{\mathbf{u}^{i*}, i \in \mathcal{N}\}$ is the $\mathbf{W}-$MPC Nash equilibrium of the game Ξ.

The metric ρ_{MPC} is a measure of worse-case efficiency loss of the game, while ξ_{MPC} provides a best-case measure of loss of efficiency. It is easy to observe that both ρ_{MPC} and ξ_{MPC} are always greater or equal to 1.

Theorem 1. *Assume that the prediction horizon of each SP $i, i \in \mathcal{N}$, is the same, i.e., $W^i = \bar{W}$ and $\bar{W}$ is also the prediction window used for (SWP). Then, the price of stability ξ_{MPC} of the game Ξ is always equal to 1, i.e., there exists a Nash equilibrium solution yields no efficiency loss under the common knowledge of the capacity constraint.*

The proof of Theorem 1 relies on the fact that the utility functions of the SPs are not coupled and the interdependence among the players enters in the constraints. The details of the proof can be found in [32]. One direct implication of Theorem 1 is that there exists a Nash equilibrium that achieves optimal social outcome for all SPs given that the demand and resource prices follow a stable patten. This result also motivates us to design algorithms to converge to the equilibrium solution which has this property. We propose a simple best-response algorithm (See Algorithm 2) based on MPC Algorithm 1 to compute this Nash equilibrium, based on dual decomposition technique [27]. We can interpret this algorithm as follows. Cloud infrastructure providers are responsible for coordinating the allocation of resources when demand exceeds capacity. Specifically, the algorithm performs an iterative process. In each iteration, every SP first requests resource quota $C^{il} \in \mathbb{R}_+$ for DC $l \in L$, and computes its allocation strategy over time, based on the received quota for every data enter. Each SP i then solves DSPP using Algorithm 1, and reports the optimal dual variable λ^{il} to each DC. Then each infrastructure provider will adjust quota for each SP based on the received dual variable. This process repeats until no SP can significantly improve its total cost. Algorithm 2 essentially provides an online computation algorithm of Nash equilibrium solutions.

VII. Experiments

We have implemented our distributed algorithm and conducted several simulation studies. To make the experiment realistic, we have used a real Internet topology graph from the *Rocketfuel project* [33], which contains link latency information. However, as the data set only contains topologies for several tier-1 Internet Service Providers (ISPs), we have

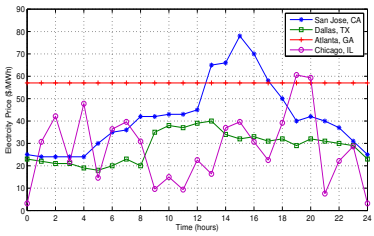

Figure 3. Prices of electricity used in the experiments

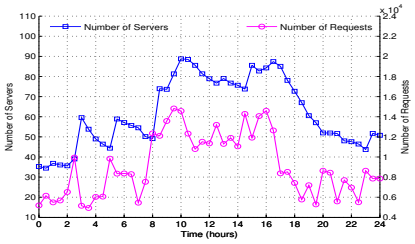

Figure 4. Impact of demand change on resource allocation

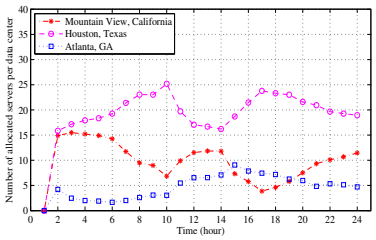

Figure 5. Impact of price on resource allocation

Algorithm 2 Iterative Algorithm for Computing the Best Nash Equilibrium

Provide initial state $\mathbf{x}_0$, $k \leftarrow 0$, Initialize $\mathbf{C}^i \in \mathbb{R}_+^L$, $\bar{\mathbf{x}}_k^i \leftarrow 0$, $\forall k \in \mathcal{K}$ $\bar{J}(\mathbf{u}^1, ..., \mathbf{u}^N) \leftarrow \infty$, $converged \leftarrow$ **false**
repeat
  for $i = 1 \rightarrow N$ **do**
    $\mathbf{u}^i \leftarrow$ solution of DSPP^i with capacity vector $\mathbf{C}_i, \forall k \in \mathcal{K}$
    $\lambda^{il} \leftarrow$ the dual variable of capacity constraint for DSPP^i of DC $l \in L$, $\forall k \in \mathcal{K}$
  end for
  $\tilde{\mathbf{C}}^i := (\mathbf{C^i} + \alpha[\lambda^{i1}, ..., \lambda^{iL}]^\top)$
  $\mathbf{C}^i := \tilde{\mathbf{C}}^i \cdot \frac{\mathbf{C}}{\sum_{i \in \mathcal{N}} \tilde{\mathbf{C}}^i}$
  $J(\mathbf{u}^1, ..., \mathbf{u}^N) = \sum_{i \in \mathcal{N}} J^i(\mathbf{u}^1, ..., \mathbf{u}^N)$
  if $|J(\mathbf{u}^1, ..., \mathbf{u}^N) - \bar{J}(\mathbf{u}^1, ..., \mathbf{u}^N)| \leq \epsilon \bar{J}(\mathbf{u}^1, ..., \mathbf{u}^N)$ **then**
    $converged \leftarrow$ **true**
  end if
  $\bar{J}(\mathbf{u}^1, ..., \mathbf{u}^N) \leftarrow J(\mathbf{u}^1, ..., \mathbf{u}^N)$
until $converged =$ **true**

augmented the topology graph by introducing intermediary ISP and access networks, similar to the procedure for generating transit-stub networks in the GT-ITM network topology generator [34]. We specify the communication latency at intra-transit, stub-transit and intra-stub domain links to be $20ms$, $5ms$ and $2ms$, respectively [35].

Based on our knowledge about Google's data centers, in our experiment, we have created 5 large data centers located in San Jose, CA, Houston, TX, Atlanta, GA and Chicago, IL. As for the access networks that originate service requests, we have studied the geographical locations of the routers and created 24 access networks in major cities across the U.S. The requests are generated from a non-homogenous Poisson process that considers both the population of each cities as well as the time of day. Generally speaking, requests from the same location follow an on-off stochastic process that has high arrival rate during working hours (8am-5pm) and low arrival rate at night.

In our experiment, the price of resources in each data center is set to the electricity price of each VM. Figure 3 shows the price of electricity of all 4 data centers during different times of each day. We assume that there are 3 types of VMs: small, medium and large. The electricity consumption of each VM type is set to 30 watts, 70 watts and 140 watts, respectively. The capacity of data centers are set to 2000 machines each.

A. The Case for a Single Service Provider

In this subsection, we report our experiment results for a single service provider. To demonstrate how our controller adjusts resource allocation to handle fluctuating requests, in the first experiment, we consider the simplest case where there is a single data center responsible for requests from a single access network. Figure 4 shows the experiment result. It can be observed that the controller always tries to adjust the resource allocation dynamically to match the demand, while minimizing the change of number of servers at each time step. This indeed is what we expect to achieve in our design. We have also analyzed the effect of prediction horizon K on the outcome of dynamic resource allocation. Figure 6 shows that the change in the number of servers tends to be less as K increases.

To demonstrate how the resource controller reacts to dynamic resource pricing, we have conducted an experiment where multiple data centers are used to serve demand from different locations with constant arrival rate. Our experiment result is shown in Figure 5. It can be observed from Figure 3 that the electricity price is generally higher in Mountain View than in Houston. The difference reaches its maximum around 5pm in the afternoon. Consequently, Figure 5 shows that our controller allocates less sources in the Mountain View data center in the afternoon. This suggests that our algorithm indeed achieves its objective of load balancing according to price change.

B. The Case for Multiple Service Providers

In this section, we analyze the outcome when multiple SPs are competing for resources in data centers. In our simulation, we generate the input parameters (μ^i, $\mathbf{D}_k^i$, s^i, c^{il},$\bar{d}^i$) for each SP $i \in \mathcal{N}$ randomly. We have implemented an iterative algorithm that computes the NE solution as described in Section VI.

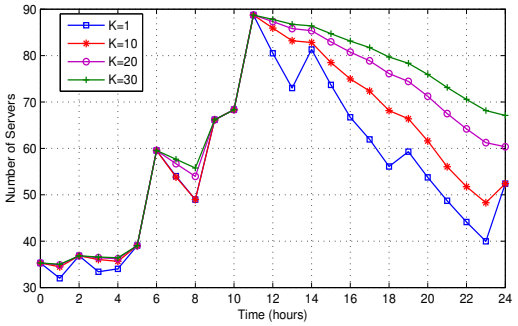

Figure 6. Effect of prediction horizon on the number of servers

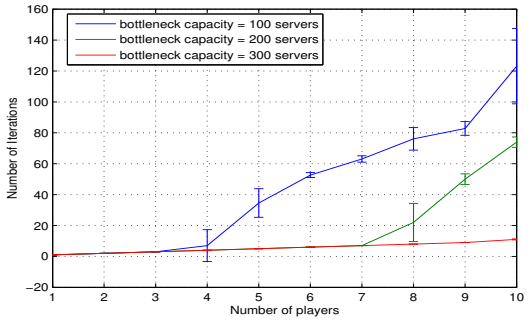

Figure 7. Impact of number of players on the convergence rate

We start with the analysis of the convergence rate of the algorithm. To produce a competition scenario, we set the number of servers in the data center with the cheapest cost (i.e. Data center in Dallas, TX) to 100, 200 and 300, respectively, and record the number of iterations the algorithm takes to produce an approximately stable outcome. Thus we call an outcome relatively stable if the difference in the solution cost is less than a small constant factor ϵ, i.e., $|\bar{J}(\mathbf{u}^1, ..., \mathbf{u}^N) - J(\mathbf{u}^1, ..., \mathbf{u}^N)| \leq \epsilon \bar{J}(\mathbf{u}^1, ..., \mathbf{u}^N))$, where $\bar{J}(\mathbf{u}^1, ..., \mathbf{u}^N)$ is the cost of the solution in the previous iteration. In our experiments, we have set $\epsilon = 0.05$. The result is shown in Figure 7. It is clear that the number of iterations to obtain a stable outcome grows with number of players and the tightness of data center capacity constraints. For instance, the case where the bottleneck data center has only 100 servers requires many more rounds to converge compared to the other two cases. It can be seen that the solution takes moderate number of steps to converge. However, we believe it is possible to improve the running time by computing approximate solutions using faster algorithms.

Finally, we have also conducted experiments to examine the impact of prediction horizon K on the solution optimality and convergence rate. Figure 8 suggests that longer prediction horizon can improve convergence rate. However, it does not imply that longer prediction horizon is desirable. Figure 9 shows that long prediction horizon can worsen the solution quality. In particular, setting $K = 2$ achieves lowest cost for this scenario. Through many experiments, we have found that the optimal prediction horizon length is highly dependent on the accuracy of prediction model for both demand and resource price. For instance, we have simulated a scenario where the both demand and price are constant over time (See Figure 10), which is easy to predict. In this case, indeed solution quality improves with the length of prediction horizon. On the other hand, when both demand and resource prices are highly volatile, a simple prediction scheme (AR in our case) is not accurate and hence a long prediction horizon will actually hurt the algorithm performance. It is part of our future work to find more accurate demand and price prediction model for cloud prediction environments.

VIII. Conclusion

Large-scale online service providers have been increasingly relying on geographically distributed cloud infrastructures for service hosting and delivery. In this paper, we have presented a framework for the dynamic service placement problem based control- and game- theoretic models. In particular, we have presented a solution that optimizes the desired objective dynamically over time according to both demand and resource price fluctuations. We have considered the case where multiple service provider compete for revenue in a dynamic manner, and have shown that there exists a socially optimal Nash equilibrium solution. In addition, we have analyzed the impact of various factors on the quality of the Nash equilibrium solution.

There are several directions that we would like pursue in the future. First, we would like to extend our framework to find a practical resource allocation mechanism that also takes into account the goal of the infrastructure providers. Second, we would like to study more realistic scenarios where number of servers can only take integer values. This is particularly important for small scale data centers. In this case, the MPC control framework would involve mixed integer programming (MIP) at each stage, which can be $\mathcal{NP}$-hard to solve. Finding an efficient approximation algorithm for this problem would be an interesting direction to pursue in the future. Another direction of future work is to design practical learning algorithms for SPs to find the best-case Nash equilibrium solution. We need to take into account the differences in rationality, intelligence and information among the players and show how these practical factors lead to different equilibrium outcomes [36].

Acknowledgment

This research was partially supported by the Natural Science and Engineering Council of Canada (NSERC) under its discovery program and partially by WCU (World Class University) program through the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0).

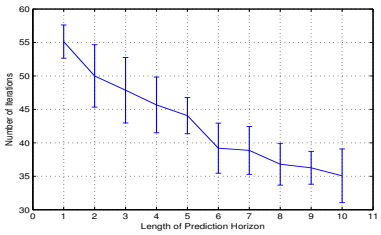

Figure 8. Impact of prediction horizon length on the speed of convergence

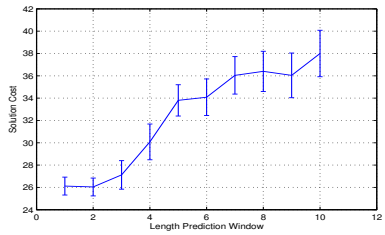

Figure 9. Impact of prediction horizon length on the cost

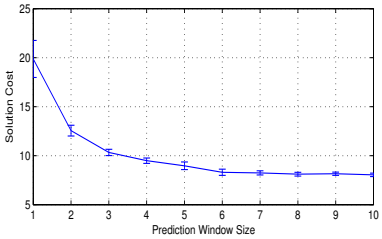

Figure 10. Impact of prediction horizon length when price and demand are both constant

REFERENCES

[1] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, 2008.

[2] L. Rao, X. Liu, and W. Liu, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *Proceedings of IEEE INFOCOM*, 2010.

[3] Z. Liu, M. Lin, A. Wierman, S. Low, and L. Andrew, "Greening geographical load balancing," in *Proceedings of ACM SIGMETRICS*, 2011.

[4] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *Proceedings of ACM SIGCOMM*, 2009.

[5] "Amazon ec2 spot instances," http://aws.amazon.com/ec2/spot-instances/.

[6] L. Qiu, V. Padmandabhan, and V. Geoffrey, "On the placement of web server replicas," *IEEE INFOCOM*, 2001.

[7] C. Vicari, C. Petrioli, and F. Presti, "Dynamic replica placement and traffic redirection in content delivery networks," in *Proceeding of IEEE (MASCOTS)*, 2007.

[8] M. Karlsson and C. Karamanolis, "Choosing replica placement heuristics for wide-area systems," in *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2004.

[9] K. Church, A. Greenberg, and J. Hamilton, "On delivering embarrassingly distributed cloud services," in *Proceedings of ACM HotNets*, 2008.

[10] S. Islam and J. Grégoire, "Network edge intelligence for the emerging next-generation internet," *Future Internet*, 2010.

[11] X. Tang and X. Jianliang, "On replica placement for qos-aware content distribution," *Proc. of IEEE INFOCOM*, 2004.

[12] Y. Chen, R. Katz, and J. Kubiatowicz, "Dynamic replica placement for scalable content delivery," in *Proceedings of Internl. workshop on Peer-To-Peer Systems (IPTPS)*, 2002.

[13]

[14] D. Openheimer, B. Chun, D. Patterson, and A. Snoeren, "Service placement in a shared wide-area platform," *Proc. of USENIX*, 2006.

[15] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, "Distributed placement of service facilities in large-scale networks," in *IEEE INFOCOM*, 2007.

[16] D. Arora, M. Bienkowski, A. Feldmann, G. Schaffrath, and S. Schmid, "Online strategies for intra and inter provider service migration in virtual networks," *Technical Report, Arxiv preprint arXiv:1103.0966*, 2011.

[17] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No power struggles: Coordinated multi-level power management for the data center," in *ACM SIGARCH Computer Architecture News*, 2008.

[18] L. Parolini, B. Sinopoli, and B. Krogh, "Reducing data center energy consumption via coordinated cooling and load management," in *Proceedings of the conference on Power aware computing and systems*, 2008.

[19] Y. Diao, J. Hellerstein, A. Storm, M. Surendra, S. Lightstone, S. Parekh, and C. Garcia-Arellano, "Using mimo linear control for load balancing in computing systems," in *Proceedings of American Control Conference (ACC)*, 2004.

[20] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," in *IEEE International Conference on Autonomic Computing (ICAC)*, 2009.

[21] A.-M. Pathan and R. Buyya, "A taxonomy and survey of content delivery networks," *Technical Report, University of Melbourne, Australia*, 2006.

[22] P. Wendell, J. Jiang, M. Freedman, and J. Rexford, "Donar: decentralized server selection for cloud services," in *Proceedings of ACM SIGCOMM*, 2010.

[23] "Amazon elastic computing cloud (amazon ec2)," http://aws.amazon.com/ec2/.

[24] G. Box, G. Jenkins, and G. Reinsel, *Time series analysis*. Holden-day San Francisco, 1970.

[25] P. Bodik, A. Fox, M. J. Franklin, M. I. Jordan, and D. A. Patterson, "Characterizing, modeling, and generating workload spikes for stateful services," in *Proceedings of ACM symposium on Cloud computing (SOCC)*, 2010.

[26] E. Camacho and C. Bordons, *Model predictive control*. Springer Verlag, 2004.

[27] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[28] W. Fernandez de La Vega and G. Lueker, "Bin packing can be solved within 1+ ε in linear time," *Combinatorica*, 1981.

[29] T. Başar and G. Olsder, *Dynamic noncooperative game theory*. Society for Industrial Mathematics, 1999.

[30] Q. Zhu and T. Basar, "Price of anarchy and price of information in n-person linear-quadratic differential games," in *IEEE American Control Conference (ACC)*, 2010.

[31] T. Başar and Q. Zhu, "Prices of anarchy, information, and cooperation in differential games," *Dynamic Games and Applications*, 2011.

[32] "Dynamic service placement in geographically distributed clouds," http://netlab.cs.uwaterloo.ca/cloud/publications?action=AttachFile\&do=get\&target=serviceplacement_tr.pdf.

[33] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Transactions on Networking (TON)*, 2009.

[34] "GTITM homepage," http://www.cc.gatech.edu/projects/gtitm/.

[35] S. Ratnasamy, M. Handley, R. Karp, and S. Scott, "Topologically-aware overlay construction and server selection," *IEEE INFOCOM*, 2002.

[36] Q. Zhu, H. Tembine, and T. Başar, "Heterogeneous learning in zero-sum stochastic games with incomplete information," in *IEEE Conference on Decision and Control (CDC),*, 2010.