2012 IEEE Fifth International Conference on Cloud Computing

A General and Practical Datacenter Selection Framework for Cloud Services

Hong Xu, Baochun Li
henryxu, bli@eecg.toronto.edu
Department of Electrical and Computer Engineering
University of Toronto

***Abstract*—Many cloud services nowadays are running on top of geographically distributed infrastructures for better reliability and performance. They need an effective way to direct the user requests to a suitable datacenter, depending on factors including performance, cost, etc. Previous work focused on efficiency and invariably considered the simple objective of maximizing aggregated utility. These approaches favor users closer to the infrastructure. In this paper, we argue that fairness should be considered to ensure users at disadvantageous locations also enjoy reasonable performance, and performance is balanced across the entire system. We adopt a general fairness criterion based on Nash bargaining solutions, and present a general optimization framework that models the realistic environment and practical constraints that a cloud faces. We develop an efficient distributed algorithm based on dual decomposition and the subgradient method, and evaluate its effectiveness and practicality using real-world traffic traces and electricity prices.**

I. Introduction

Internet-scale online services are becoming increasingly essential to our everyday life, with important applications including web search, video streaming, and online gaming. The burgeoning of cloud computing platforms, such as Amazon AWS, further enables small enterprises to rapidly deploy cloud services at scale. Almost all of these cloud services are built atop geographically distributed infrastructures, i.e. datacenters located in different regions as shown in Fig. 1, to provide reliability and performance. They need an effective way to direct clients across the wide area to an appropriate datacenter. Usually, cloud services handle datacenter selection by deploying mapping nodes, which are typically DNS servers, to customize the IP address(es) returned to different clients. Alternatively, they can also outsource datacenter selection to third-parties [1], [2] or the cloud provider [3].

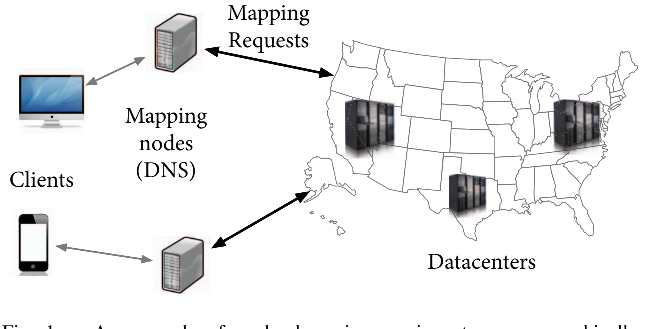

Fig. 1. An example of a cloud service running atop a geographically distributed cloud infrastructure.

An efficient datacenter selection algorithm is imperative to the operation of cloud services. Many previous works exist in this area. The problem can be cast as an optimization that maximizes the total utility or minimize the total cost, both of which can be defined in different ways depending on how the cloud services define performance. The formulated problems and their solutions are focused on the efficiency issue. These approaches thus tend to favor clients closer to the infrastructure, because prioritizing them in assigning datacenter capacity improves system utility. This results in poor performance for disadvantaged users far away from the infrastructure, and can potentially lead to substantial revenue losses. For instance, Amazon reports that every 100 ms delay in page load time decreases sales by 1 percent [4].

Many fairness criteria have been considered in the traditional context of traffic engineering in wired networks, and resource allocation in cellular networks. Max-min and proportional fairness models are arguably the most widely used in the literature [5]–[7]. However, in the context of cloud computing, cloud services usually have distinct Service Level Agreements (SLAs) that need to be satisfied. This requirement cannot be accommodated by neither the max-min nor the proportional fairness model. Moreover, the max-min approach deals with the worse-case scenario and penalizes clients with better conditions, thus unnecessarily reducing the system efficiency.

In this paper, our main contribution is a general optimization framework for datacenter selection based on the fairness criterion stemming from the concept of Nash bargaining games in game theory [8]. Essentially, we can view the problem as a cooperative bargaining game amongst the mapping nodes. Each node, serving requests aggregated from a specific area, has a minimum utility requirement for clients it serves based on the SLA, and they compete ("bargain") *cooperatively* for the cloud resources. The solution of this game, called the Nash bargaining solutions (NBS), is a unique Nash equilibrium point with NBS fairness among mapping nodes, which is a generalized proportional fairness notion. It strives to satisfy the minimum utility requirements first, and allocates the remaining datacenter capacity proportionally among mapping nodes according to their conditions. Therefore it is able to achieve high system efficiency with good fairness.

Our framework is general in the sense that it models practical environments that cloud operators face. The utility abstraction encompasses many possible performance considerations, including throughput and latency, as well as policy preferences, such as datacenter load, user locality, etc. It is a function of user-datacenter tuples in order to model the location diversity of performance. We also consider the cost of serving requests and model the price diversity of datacenters, since recent works have recognized the benefit of location-dependent electricity price in terms of minimizing the energy bill of datacenters [9].

By using dual decomposition, our optimization formulation can be decentralized to the mapping node level. Specifically, the problem can be decomposed into many subproblems, each solvable by an individual mapping node itself. This enables us to develop efficient distributed implementations of our datacenter selection algorithm to find the optimal node-datacenter assignment, based on the subgradient method. We also point out that our algorithms remain relevant in and are applicable to other request direction scenarios, such as a commercial CDN.

We evaluate the effectiveness and practicality of our decentralized implementation using the real-world traffic traces collected from UUSee [10], a commercial Video-on-Demand provider in China, as well as real-world electricity prices collected from the U.S. Federal Energy Regulatory Commission [11]. Results show that our algorithm achieves better fairness with satisfactory performance in terms of total utility and cost, and is amenable to practical implementations, since it converges within 20 iterations.

The rest of the paper is structured as follows. In Sec. II we introduce the concept of NBS and present our formulation of the datacenter selection problem. In Sec. III we develop distributed algorithms to solve the optimization problem based on dual decomposition. Numerical results are provided in Sec. IV. In Sec. V we summarize related work. Finally, we conclude the paper in Sec. VI.

II. An Optimization Framework based on NBS

In this section, we present our optimization framework based on NBS.

A. System Model

We start by introducing the system model. We consider a cloud infrastructure with M geographically distributed datacenters. The cloud deploys N *mapping nodes* (e.g. DNS servers) at different locations to direct client requests to the appropriate datacenters as determined by the datacenter selection algorithm. Since the request traffic fluctuates dynamically, the datacenter selection algorithm has to be run periodically to optimize performance.

We assume that the cloud operator employs learning techniques [12], [13] to predict the traffic demand of each node D_i in each epoch with satisfactory accuracy. We also assume that the electricity price at each datacenter W_d is available at the beginning of an epoch, and remains static throughout the entire epoch. This is a practical assumption in today's electricity market. If the local electricity market of datacenter d is a regulated utility region, the electricity price is fixed. If on the other hand the datacenter is in a deregulated market region, such as California and Texas, there is a forward market with settlements of various kinds, such as day-ahead and hour-ahead, for customers to lock in the price [11]. $\boldsymbol{W} = [W_d]$ is called the cost matrix.

We use an abstract *utility notion* U_{id} to capture the performance of the cloud service, when a request from node i is directed to datacenter d. This notion allows us a considerable amount of expressiveness. For example, if the cloud service is an interactive application and seeks minimal latency, U_{id} can be a decreasing function of the round trip time (RTT), directly measured or estimated by various means. If the cloud service is a bulk transfer application and seeks good throughput, U_{id} can be a decreasing function of the network congestion level or the link utilization. This utility function can be any shape—e.g. a convex function of the latency or throughput. For more discussion of the generality of the utility notion one can refer to [2]. Finally, $\boldsymbol{U} = [U_{id}]$ is called the performance matrix in the following.

B. Basics of Nash bargaining solutions

We present the salient concepts and results from Nash bargaining solutions here, which are used in the sequel. For details we refer readers to [8].

The basic setting is as follows: Let $\mathcal{N}$ be the set of mapping nodes. Let $\mathcal{S}$ be a closed and convex subset of $\mathcal{R}^N$ to represent the set of feasible utility allocations. Let $U_i^{\min}$ be the minimal utility that the i-th node require. In our datacenter selection problem, this is obtained from clients' SLAs. Suppose $\{U_i \in \mathcal{S} | U_i \geq U_i^{\min}, \forall i \in \mathcal{N}\}$ is a nonempty bounded set. Define $\boldsymbol{U}^{\min} = (U_1^{\min}, \ldots, U_N^{\min})$, then the pair $(\mathcal{S}, \boldsymbol{U}^{\min})$ is called a N-node bargaining problem.

Within the feasible set $\mathcal{S}$, we first define the notion of Pareto optimality as a selection criterion in a typical game.

Definition 1: The point $(U_1, \ldots, U_N)$ is said to be *Pareto optimal* if and only if there is no other allocation U_i' such that $U_i' \geq U_i, \forall i \in \mathcal{N}$, and $U_i' > U_i, \exists i \in \mathcal{N}$.
That is, there exists no other allocation that leads to superior performance for some node without inferior performance for some other node.

Our next selection criterion is the fairness of resource sharing. In this paper, we use the NBS fairness axioms from game theory.

Definition 2: $\bar{\mathbf{r}}$ is a **NBS**, *i.e.* $\bar{\mathbf{r}} = \phi(\mathbf{S}, \mathbf{R}^{\min})$, if the following axioms are satisfied: *Feasibility, Pareto Optimality, Independence of Irrelevant Alternatives, Independence of Linear Transformations*, and *Symmetry* [8].

Theorem 1: There is a unique solution function $\phi(\mathcal{S}, \boldsymbol{U}^{\min})$ that satisfies all axioms in Definition 2 such that [8]

$$\phi(\mathcal{S}, \boldsymbol{U}^{\min}) \in \underset{\boldsymbol{U} \in \mathcal{S}, \boldsymbol{U} \succeq \boldsymbol{U}^{\min}}{\operatorname{argmax}} \prod_{i \in \mathcal{N}} \left(U_i - U_i^{\min}\right). \qquad (1)$$

NBS fairness naturally fits to the datacenter selection problem. $U_i^{\min}$ correspond to the specific SLA of the mapping node to guarantee performance of clients. After the minimal requirements are met for all nodes, the rest of the resources are allocated *proportionally* to nodes according to their conditions, so that every node obtains a fair share. When $U_i^{\min} = 0$ for all i, NBS fairness reduces to proportional fairness [7].

C. An Optimization Framework based on NBS

Now we formally introduce our optimization framework. For our datacenter selection problem, we wish to consider *long-term* NBS fairness which depends on the average utility. Long-term fairness not only faithfully reflects clients' performance, but also gives more flexibility to exploit location diversity of the cloud infrastructure. As discussed, the datacenter selection problem can be viewed as a bargaining game. Each mapping node has the average utility $\bar{U}_i$ as its objective. The goal is to maximize all $\bar{U}_i$ cooperatively. Each node also has $U_i^{\min}$ that represents the minimal utility requirement based on the SLAs with the cloud. In the general case, different nodes can have different SLAs depending on their locations and thus different $U_i^{\min}$. Note that the SLAs have to be satisfied in each epoch, instead of statistically over time. The problem at epoch t, then, is to find the NBS, i.e. to solve for the optimization problem

$$\max_{\boldsymbol{U}(t)\succeq\boldsymbol{U}^{\min}} \prod_i^N \left(\bar{U}_i(t) - U_i^{\min}\right). \tag{2}$$

Mathematically it is equivalent to the following objective:

$$\sum_i^N \ln\left(\bar{U}_i(t) - U_i^{\min}\right). \tag{3}$$

Note that $\bar{U}_i(t)$ is a function of the instantaneous utility $U_i(t)$, typically obtained by using an exponentially weighted low-pass time window:

$$\bar{U}_i(t) = (1-\rho_w)\bar{U}_i(t-1) + \rho_w U_i(t).$$

$\rho_w = (T_s/T_w)$ where T_s is the slot length and T_w is the length of the time window to calculate the average.

This is difficult to solve because the logarithm function is not linear. It has been shown in the seminal work [14] that maximizing the total marginal gain of $\sum_i V_i'(\bar{X}_i(t-1))X_i(t)$ at each t achieves long-term maximization of $\sum_i V_i(\bar{X}_i(t))$ asymptotically. Thus, we can greatly reduce the computational complexity by transforming (3) to the following linear objective:

$$\sum_i^N \frac{U_i(t) - U_i^{\min}}{\bar{U}_i(t-1) - U_i^{\min}}. \tag{4}$$

Note that without considering long-term performance, the optimization must guarantee fairness in each epoch. However, when a time window is used, the fairness requirement is relaxed to the time window length. This provides more flexibility to improve the system efficiency, by making the current datacenter selection related to previous ones. The term $\bar{U}_i(t-1) - U_i^{\min}$ in the denominator of (4) serves as a weight factor to adjust the priority of node i. If the node has an unfairly large utility gain from previous epochs, its priority is lower in obtaining a good datacenter in the current epoch. Therefore the long-term fairness model encourages nodes to share the cloud resources cooperatively, and in turn gain more when it needs more help. In general it helps to achieve better system performance while enforcing the fairness notion over long run.

Without loss of generality, we use $\bar{U}_i^-$ to denote $\bar{U}_i(t-1)$, and drop the time index t in all notations and the constant term $U_i^{\min}$ in the numerator of the objective function (4). The optimization problem at each epoch then can be formulated as

$$\text{DC-FAIR:} \quad \max_{\boldsymbol{U}} \sum_i^N \frac{U_i}{\bar{U}_i^- - U_i^{\min}} \tag{5}$$

$$\text{s.t. } U_i = \sum_d^M p_{id}U_{id}, \forall i, \tag{6}$$

$$\sum_d^M p_{id}U_{id} \geq U_i^{\min}, \forall i, \tag{7}$$

$$\sum_d^M p_{id} = 1, \forall i, \quad p_{id} \geq 0, \forall i, d, \tag{8}$$

$$\sum_i^N p_{id} \geq P_d, \forall d, \tag{9}$$

$$\sum_i^N p_{id}D_i \leq C_d, \forall d, \tag{10}$$

$$\sum_d^M \sum_i^N W_d p_{id} D_i \leq B. \tag{11}$$

The decision variables are p_{id}, i.e. the proportion of requests directed to datacenter d from node i. The equality (6) calculates the total utility U_i given by the datacenter selection p_{id} and the performance matrix U_{id}. Constraint (7) is the basic SLA requirement for each node i. Constraint (8) corresponds to the simple facts that all the requests at node i should be served and that the decision variable p_{id} is non-negative. Constraint (9) models the possible load balancing requirement of the cloud service such that datacenter d should at least handle $P_d \in [0,1]$ out of the total requests. (10) is the capacity constraint at d, and (11) captures the cost constraint that the total cost of serving all the requests should not exceed the budget B.

Some may argue that some QoS parameters defined in a SLA, such as fee, responsibility, and security level, cannot be captured by utility. We comment that though this is the case, its effect on our framework is minimal. Typical cloud services do not have QoS guarantees on security and privacy, which cannot be feasibly realized currently and still remains an active research topic. Even when these QoS parameters do need to be considered, they can be easily incorporated as additional constraints on the request direction matrix, because the requests of a mapping node can only be directed to datacenters that satisfy the responsibility and security requirements.

III. A Decentralized Implementation

The optimization problem DC-FAIR is essentially a LP, and can be solved in polynomial time. However, this requires a central coordinator which introduces a single point of failure and is vulnerable to attacks. Further, the complexity of solving the LP also increases significantly when the problem size scales up, since the number of variables is NM and the number of constraints is $2N+3M$. A centralized solution also makes it less adaptive to sudden changes in traffic demand in a flash crowd scenario. Thus, for reasons of reliability, security, scalability, and performance, we are motivated to develop distributed solutions in which the mapping nodes iteratively solve the DC-FAIR problem.

A. Dual Decomposition

Substituting (6) into the objective function (5), and relax the constraints (9)–(11), we can obtain the Lagrangian of DC-FAIR:

$$L(\boldsymbol{p}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \nu) = \sum_i \sum_d \frac{p_{id} U_{id}}{\bar{U}_i^- - U_i^{\min}} + \sum_d \lambda_d \left(\sum_i p_{id} - P_d \right)$$
$$+ \sum_d \mu_d \left(C_d - \sum_i p_{id} D_i \right) + \nu \left(B - \sum_d \sum_i W_d p_{id} D_i \right),$$

where $\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu} \succeq \mathbf{0}$ are the Lagrange multipliers associated with the load balancing, capacity, and cost constraints, respectively. The dual function is then

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}, \nu) = \begin{cases} \max\limits_{\boldsymbol{p}} & L(\boldsymbol{p}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \nu) \\ \text{s.t.} & \boldsymbol{p} \succeq \mathbf{0},\ \boldsymbol{U}_i \succeq \boldsymbol{U}_i^{\min} \end{cases} \tag{12}$$

To solve $g(\boldsymbol{\lambda}, \boldsymbol{\mu}, \nu)$, it is equivalent to solving the problem with the following objective

$$\sum_i \sum_d p_{id} \left(\frac{U_{id}}{\bar{U}_i^- - U_i^{\min}} + \lambda_d - \mu_d D_i - \nu W_d D_i \right)$$

where the constant terms in $L(\boldsymbol{p}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \nu)$ can be safely removed. The key observation here is that it can be decomposed into N per-node maximization sub-problems

$$\begin{aligned} \max_{p_{id} \geq 0} \quad & \sum_d p_{id} \left(\frac{U_{id}}{\bar{U}_i^- - U_i^{\min}} + \lambda_d - \mu_d D_i - \nu W_d D_i \right) \\ \text{s.t.} \quad & \sum_d p_{id} = 1,\ \sum_d p_{id} U_{id} \geq U_i^{\min}, \end{aligned} \tag{13}$$

The per-node sub-problem naturally embodies an economic interpretation. Each mapping node i strives to maximize the total utility of serving the requests, discounted by the costs of violating the load balancing, capacity, and budget constraints, as priced by the Lagrange multipliers $\boldsymbol{\lambda}, \boldsymbol{\mu}, \nu$. The SLA constraint prevents the node from directing all its requests to the most "economical" datacenter, and forces it to distribute the requests among datacenters with good performance. This balances the cloud operator's interest in minimizing its costs and the client's interest in maximizing its performance.

Note that the decomposition greatly reduces the complexity of the optimization. The per-user sub-problem has only M variables and 2 constraints. In a typical production cloud, the number of datacenters M is much smaller than the number of mapping nodes N, which can be hundreds. It is essentially a small-scale linear program that can be solved efficiently by standard optimization solvers.

B. A Distributed Algorithm

We have shown that the dual function of DC-FAIR can be decomposed into N per-node maximization problem, which is a simple linear program. Now we need to solve the dual problem

$$\begin{aligned} \min_{\boldsymbol{\lambda}, \boldsymbol{\mu}, \nu} \quad & g(\boldsymbol{\lambda}, \boldsymbol{\mu}, \nu) \\ \text{s.t.} \quad & \boldsymbol{\lambda}, \boldsymbol{\mu} \succeq \mathbf{0}, \nu \geq 0. \end{aligned} \tag{14}$$

Subgradient method [15] can be used to solve the dual problem. The updating rules for the dual variables are as follows:

$$\lambda_d^{(l+1)} = \left[\lambda_d^{(l)} + \delta_d^{(l)} \left(P_d - \sum_i p_{id} \right) \right]^+, \forall d, \tag{15}$$

$$\mu_d^{(l+1)} = \left[\mu_d^{(l)} + \epsilon_d^{(l)} \left(\sum_i p_{id} D_i - C_d \right) \right]^+, \forall d, \tag{16}$$

$$\nu^{(l+1)} = \left[\nu^{(l)} + \sigma^{(l)} \left(\sum_d \sum_i W_d p_{id} D_i - B \right) \right]^+, \tag{17}$$

where $[x]^+$ represents $\max\{0, x\}$, and $\boldsymbol{\delta}, \boldsymbol{\epsilon}, \sigma$ are the *step sizes*. According to [15], the above procedure is guaranteed to converge as long as the following condition is satisfied.

Proposition 1: The subgradient updates as in (15)–(17) converge to the optimal dual variables if a diminishing step size rule is followed for choosing $\boldsymbol{\delta}, \boldsymbol{\epsilon}, \sigma$ [15].

Observe that, because of the dual decomposition, dual optimization by subgradient method can be done in a *distributed* fashion. First, in each iteration, the per-node problems (13) can be solved simultaneously by the mapping nodes, with $\boldsymbol{\lambda}, \boldsymbol{\mu}, \nu$ given by datacenters. Second, subgradient updates can also be distributively performed by each mapping node. The algorithm can be perceived as an iterative bargaining process. The dual variables $\boldsymbol{\lambda}, \boldsymbol{\mu}$ are transmitted from datacenters to all nodes. They serve as price signals to coordinate the resource consumption and load balancing. For example, when the total traffic routed to datacenter d exceeds its capacity, i.e. $\sum_i p_{id} D_i > C_d$, d increases its price μ_d for the next round of bargaining to suppress the excessive demand. Similarly, if d has not reached its minimum load P_d, i.e. $\sum_i p_{id} < P_d$, it increases the reward price λ_d to attract more traffic and therefore balance the load[1]. The process continues until it converges to the optimal resource allocation.

The update method of the other dual variable ν, i.e. the budget price, is also worth mentioning. While λ_d and μ_d can be independently updated by each datacenter with only local information, ν needs to be updated with global information

[1]Note that λ is a positive term in (13).

from all datacenters. This can be done in a distributed way as follows. Initially, the previous $\nu^{(l)}$ is made common knowledge among the datacenters. First, a datacenter is randomly chosen and given a token with the total budget B. It calculates its own monetary cost of serving the requests $\sum_i W_d p_{id} D_i$, and deduct this amount from B. It puts a mark in the token, and pass it on to the next datacenter, who also updates the remaining budget, marks the token, and passes it further down. A datacenter determines it is the last one in the loop by examining that except itself, everyone else has marked the token. It thus updates the remaining budget, calculates the updated budget price $\nu^{(l+1)}$, and broadcasts to every mapping node and datacenter.

Algorithm 1 *Optimal Distributed Datacenter Selection*

```
1. Each datacenter initializes $\lambda_d^{(0)}, \mu_d^{(0)}$. $\nu^{(0)}$ is initialized to 0.
2. Each mapping node collects $\boldsymbol{\lambda^{(l)}}, \boldsymbol{\mu^{(l)}}, \nu^{(l)}$, and independently solves the per-node problem (13) using standard optimization solvers and obtain $p_{id}$.
3. Each datacenter $d$ collects its load $p_{id}$ from all nodes, and perform a subgradient update for the load and capacity price $\lambda_d^{(l)}, \mu_d^{(l)}$ as in (15) and (16). The updated $\lambda_d^{(l+1)}, \mu_d^{(l+1)}$ are broadcast to every mapping node.
4. A datacenter is randomly chosen and given a token with the budget $B$.
5. The datacenter deducts its cost $\sum_i W_d p_{id} D_i$ from the remaining budget in the token, marks it, and passes it down.
6. Repeat step 5 until the last datacenter calculates the final remaining budget, updates $\nu^{(l)}$ as in (17), and broadcasts to every datacenter and mapping node.
7. Return to step 2 until convergence.
```

The complete bargaining algorithm is shown in Algorithm 1. Since it optimally solves the dual problem (14), it optimally solves the primal problem DC-FAIR because the duality gap for linear programs is zero.

Theorem 2: The distributed bargaining algorithm as shown in Algorithm 1 always converges, and when it converges its solution *optimally* solves the datacenter selection problem DC-FAIR.

C. Discussions

We discuss some implementation issues related to our distributed datacenter selection algorithm.

First, step (2) and (3) of Algorithm 1 need to be performed in a synchronized fashion across the mapping nodes, which may be of concern to some readers for practical implementation. We comment that this can actually be done efficiently. Note that the synchronization requirement is loose in the sense that we only require each node to have the complete dual variables before solving (13) in step (2), and that step (3) to be performed after step (2) is completed for each node. This can be readily achieved without any need of explicit or implicit time synchronization, by having each node to wait for all the updated dual variables $\boldsymbol{\lambda^{(l)}}, \boldsymbol{\mu^{(l)}}, \nu^{(l)}$ at round l to be received completely before attempting to solve the per-node optimization, and by having each datacenter to wait for all the updated solutions $p_{id}^{(l)}$ to be received completely before updating its own dual variables. Therefore the entire procedure is naturally a self-synchronizing solution. The concept of round does not need any synchronization or central coordination either, since each node and datacenter knows for how many times it has solved the per-node problem or updated the dual variables.

Second, we examine the message exchanging overhead of Algorithm 1. Each datacenter needs to share its load and capacity price to all nodes, which implies $O(MN)$ messages in total, each of size 2. Each node needs to share its selection decision of size M with all datacenters, which implies $O(MN)$ messages, each of size $O(M)$. Thus in each round, $O(MN)$ messages are exchanged each of size $O(M)$. The overhead scales linearly with the number of nodes and datacenters, which is practical.

IV. Evaluation

We present our simulation studies in this section.

A. Setup

1) Demand matrix: To represent the request traffic for a cloud service, we use the real-world traces collected from UUSee Inc. [10], a major commercial Video-on-Demand provider with servers distributed geographically in China. The dataset contains, among other things, the bandwidth demand for UUSee video programs sampled every 10 minutes, in a 12-day period during the 2008 Beijing Olympics, from 14:51:58, Friday, August 8, 2008 (GMT+8) to 23:43:56, Tuesday, August 19, 2008 (GMT+8). Although the scale of the UUSee infrastructure is not as large as that of a cloud provider, we believe the traces faithfully reflect the traffic demand distribution for a cloud service, and it is appropriate to use them for the purpose of benchmarking the performance of our datacenter selection algorithm.

The prediction of traffic demand can be done accurately as demonstrated by previous work [12], [13], and in the simulation we simply adopt the measured traffic demand as the predicted demand matrix $\boldsymbol{D}$. We use the traffic demand of distinct video channels to represent demand of distinct mapping nodes. We simulate a cloud with 100 mapping nodes. Fig. 2 shows a sample of traffic demand for 3 mapping nodes. Since the data is collected every 10 minutes, the optimization epoch is also set to 10 minutes.

2) Datacenter placement and cost matrix: To capture the location diversity of the cloud infrastructure and electricity market, we assume the datacenters are deployed across the continental U.S. According to the Federal Energy Regulatory Commission (FERC), the U.S. electricity market is consisted of multiple regional markets as shown in Fig. 3 [11]. Each regional market has several hubs with their own pricing. Therefore for the ease of exploration, we assume that there is one datacenter deployed in a randomly chosen hub in each

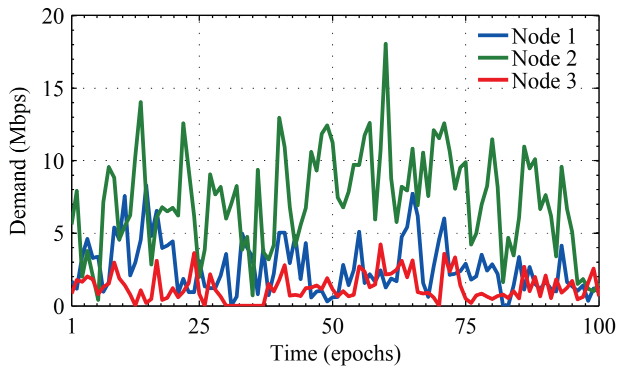

Fig. 2. A 100-epoch sample of traffic demand for 3 mapping nodes.

regional market. We use the 2011 annual average day-ahead on peak price (\$/MWh) published online by FERC as the electricity price for each datacenter, i.e. W_d, as summarized in Table I [11]. The capacity of each datacenter is randomly set such that their total capacity is 15000 Mbps.

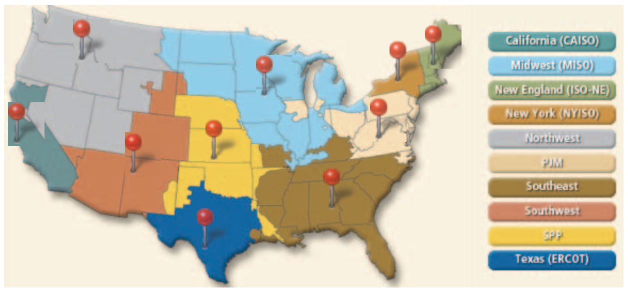

Fig. 3. The U.S. electricity market and our cloud datacenter map. Source: FERC [11].

TABLE I
2011 ANNUAL AVERAGE DAY AHEAD ON PEAK PRICE (\$/MWH) IN DIFFERENT REGIONAL MARKETS. SOURCE: FERC [11].

Region	Hub	Price
California	NP15	\$35.83
Midwest	Michigan Hub	\$42.73
New England	Mass Hub	\$52.64
New York	NY Zone J	\$62.71
Northwest	California-Oregon Border (COB)	\$32.57
PJM	PJM West	\$51.99
Southeast	VACAR	\$44.44
Southwest	Four Corners	\$36.36
SPP	SPP North	\$36.41
Texas	ERCOT North	\$61.55

3) Performance matrix: Finally, we consider a latency-critical cloud service, whose utility can be defined by the negative Euclidean distance between the mapping nodes and the datacenters. To calculate the performance matrix, we first obtain the longitude and latitude of ten randomly chosen counties in the area covered by each of the ten hubs as the exact locations of our datacenters in the U.S. We then randomly choose another 100 counties as the locations of the 100 mapping nodes. All the location information is obtained from [16]. The Euclidean distance between any given pair of mapping node and datacenter then can be readily calculated, which constitutes the performance matrix $\boldsymbol{U}$. Without loss of generality, we assume that serving 1 Gbps per epoch, i.e. 10 minutes, consumes 10 kWh electricity. The SLA constraint for each node $U_i^{\min}$ is set to be 30% lower than the best utility i may obtain among all datacenters.

B. Effectiveness

We first evaluate the effectiveness of our distributed datacenter selection algorithm. The budget B is set to \$5 per epoch. Fig. 4 shows the per-node average utility with total demand for a 100-epoch period of time. We observe that when the average utility stands at -520 most of the time, i.e. on average the request is directed to a datacenter 520 km away, unless the total demand shoots beyond around 1200 Mbps. This demonstrates that when demand is low, our algorithm matches mapping nodes to their closet datacenters, thus directing requests to the best available datacenters.

The consistent performance can be better explained by Fig. 5 that shows the total cost of serving requests versus total demand under the same setting. Clearly the total cost figure closely follows the total demand, and is below the \$5 budget limit all the time. Thus, the budget constraint is inactive, which corresponds to a scenario where the cloud operator has ample financial resources to optimize performance without having to consider the extra cost involved in doing so. This results in a constant average utility curve amid fluctuating demand. When the demand becomes significant, it becomes necessary to direct some requests to distant datacenters to conform the budget constraint, which explains the performance degradation in epochs 74–84.

To see the effectiveness of our algorithm on guaranteeing SLAs, we plot $U_i - U_i^{\min}$ for each mapping node at three sampled epochs, 10, 75, and 77 in Fig. 6. Epoch 10 corresponds to a low demand epoch, and epoch 75 and 77 correspond to extremely high demand periods. We observe that no matter the demand, the SLA is *always* satisfied for each node because the curves stay above 0. It can also be seen that each node enjoys a better utility when demand is low in epoch 10, which is intuitive to understand.

Now to see the effectiveness of our algorithm on guaranteeing the budget, we evaluate the effect of budget on the performance of datacenter selection. We run our algorithm with a reduced budget of \$4, while keeping the other settings unchanged. Compared to Fig. 4, Fig. 7 shows that now the performance swings widely along with the demand curve, and degrades to less than -600 when demand pikes over 1000 Mbps. Fig. 8 also shows that the total cost is reduced compared to Fig. 5. It is thus evident that a tighter budget negatively affects the performance, but helps reducing the operating costs of the datacenters.

Readers may notice that during the epochs from 74 to 84 with a high volume of demand, total cost is actually beyond the \$4 budget. The stringent budget constraint causes the problem

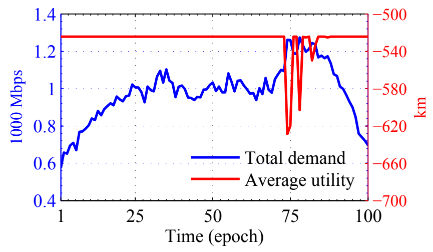

Fig. 4. Average utility and total demand, $B = 5$.

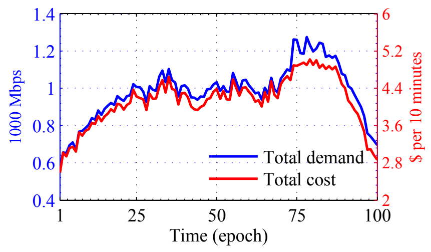

Fig. 5. Total cost and total demand, $B = 5$.

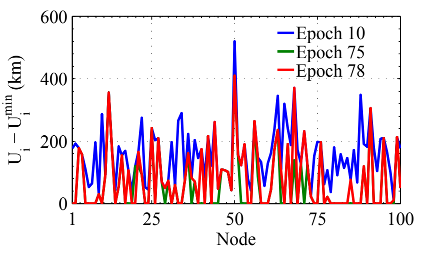

Fig. 6. $U_i - U_i^{\min}$ at sampled epochs, $B = 5$.

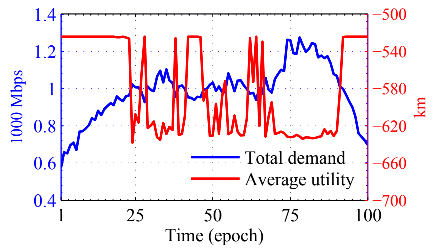

Fig. 7. Average utility and total demand, $B = 4$.

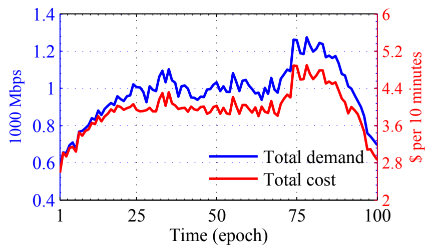

Fig. 8. Total cost and total demand, $B = 4$.

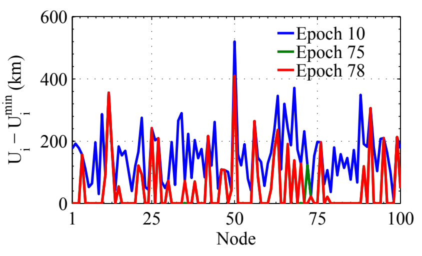

Fig. 9. $U_i - U_i^{\min}$ at sampled epochs, $B = 4$.

to be infeasible for those epochs with extremely high demand. In our implementation, we set a stopping criterion to be related to the absolute change of the objective values, so that our algorithm satisfies the budget constraint whenever it is feasible to do so, and still produces sensible results with an infeasible budget constraint. The SLA constraint is always satisfied as shown in Fig. 9, however, because it is not relaxed in the per-node problem (13).

C. Efficiency and fairness

To demonstrate the efficiency and fairness achieved with our algorithm, we use a simple LP, called DC-OPT, that shares all the same constraints with our DC-FAIR problem, but uses the total utility $\sum_i U_i$ as the objective function instead of (5) as the benchmark. Therefore it does not consider fairness. DC-OPT can also be solved by the same method of distributed subgradient updates.

Fig. 10 first shows the fairness comparison of DC-FAIR and the simple DC-OPT. We use the standard deviation of $\bar{U}_i(t) - U_i^{\min}$, i.e. the time average surplus utility, across nodes at each epoch t as the fairness measure. A fair selection algorithm yields a proportional surplus utility allocation among nodes with a smaller standard deviation, whereas a poor algorithm without considering fairness produces an allocation with a larger standard deviation. We can observe from Fig. 10 that in most of the time, DC-FAIR achieves a much smaller standard deviation for the surplus utility, which translates to much better fairness among nodes. Fairness is generally improved over time except for the high demand epochs, which demonstrates the effectiveness of our long-term NBS based fairness model.

The performance and cost comparisons between our DC-FAIR and the DC-OPT algorithms are shown in Fig. 11 and Fig. 12. DC-OPT performs better than our DC-FAIR in terms of per-node average utility along the time line, which is expected since the sole objective of DC-OPT is to maximize the aggregated utility without considering fairness. In terms of cost, our DC-FAIR algorithm achieves slightly better results.

Overall, DC-FAIR attains a different performance-fairness trade-off with much better fairness among nodes, at the cost of system-wide performance. As we discussed, fairness is a critical requirement of the datacenter selection problem, since clients served by a particular mapping node wish to obtain a fair share of the available cloud resources after the minimum SLAs are satisfied. Our algorithm thus achieves the fairness requirement with satisfactory performance.

D. Speed of convergence

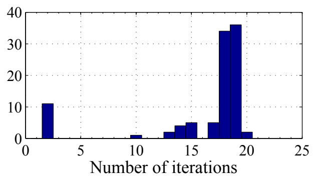

Fig. 13. Histogram of the convergence speed, $B = 5$.

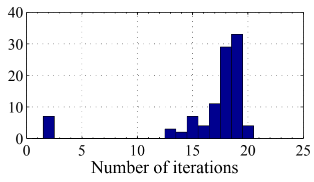

Fig. 14. Histogram of the convergence speed, $B = 4$.

Finally, we evaluate the convergence speed of the DC-FAIR algorithm. Fig. 13 and Fig. 14 show the histograms of the number of iterations the algorithm takes to converge to the optimal solution for different budgets. Clearly, we observe that it usually takes 19–20 iterations, and never takes more than 20 iterations to finish. For a problem with 10 datacenters and 100 nodes, the speed of convergence should be considered

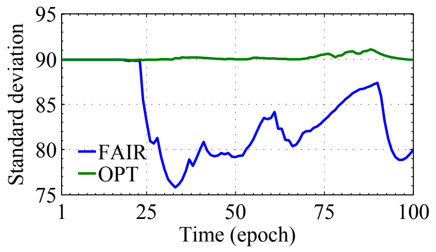

Fig. 10. Fairness comparison, $B = 4$.

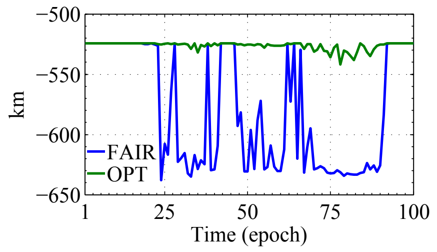

Fig. 11. Average utility comparison, $B = 4$.

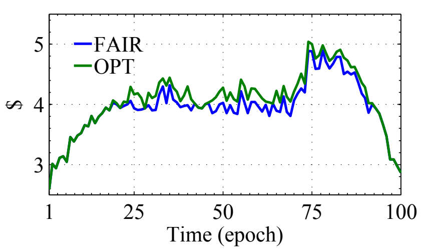

Fig. 12. Total cost comparison, $B = 4$.

very satisfactory. Thus we believe that our algorithm is also practical for real-world problems with larger scales.

V. Related Work

The topic of datacenter selection and load direction for a geo-distributed cloud has started to gain attention in the research community. Qureshi et al. [9] introduced the idea of utilizing the location diversity of electricity spot price to intelligently direct requests to datacenters with lower prices. Wendell et al. [2] developed a decentralized datacenter selection algorithm for cloud services, and evaluated its performance using a prototype and realistic traffic traces. Rao et al. [17] considered a joint load balancing and power control problem for Internet datacenters to exploit the time and location diversity of electricity price. [18] specifically considered the effect of geographical load balancing on providing environmental gains by encouraging the use of green energy. [19] studied a complementary problem of data placement in a geo-distributed cloud, considering the data locality, WAN bandwidth costs, and datacenter capacity. These works, however, do not consider fairness in their problem formulation.

The concept of Nash bargaining games has been applied to networking problems in other domains. [20] applies it to ensure fairness in a network flow control problem. Kelly in [6] has shown that Nash bargaining ensures proportional fairness in a TCP setting. [7] studied a fair multiuser channel allocation scheme for OFDMA networks based on Nash bargaining solutions and coalitions. [21] applied NBS to a resource allocation problem in cooperative cognitive radio networks. We consider a new context of cloud computing, the specifics of which are not captured in these works.

VI. Concluding Remarks

In this paper, we presented a general optimization framework to solve the datacenter selection problem for cloud services. Our framework is based on the concept of NBS fairness to proportionally allocate the available cloud resources across the mapping nodes, after the minimum SLA requirements are met. We adopted a dual decomposition approach to develop a distributed algorithm based on the subgradient method. Our work can be extended in many directions. One possible direction is to consider the online datacenter selection with fairness consideration, which is more difficult than the offline problem we solved here. The other direction is to take into account the stochastic nature of the request traffic, and provide stochastic performance guarantee amid such random traffic.

References

[1] "DynDNS," http://dyn.com/dns/.

[2] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "DONAR: Decentralized server selection for cloud services," in *Proc. ACM SIGCOMM*, 2010.

[3] "Amazon AWS elastic load balancing," http://aws.amazon.com/elasticloadbalancing/.

[4] R. Kohavi and R. Longbotham, "Online experiments: Lessons learned," *IEEE Computer*, vol. 40, no. 9, pp. 85–87, September 2007.

[5] D. K. Goldenberg, L. Qiu, H. Xie, Y. R. Yang, and Y. Zhang, "Optimizing cost and performance for multihoming," in *Proc. ACM SIGCOMM*, 2004.

[6] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Operat. Res. Soc.*, vol. 49, no. 3, pp. 237–252, March 1998.

[7] Z. Han, Z. J. Ji, and K. J. R. Liu, "Fair multiuser channel allocation for OFDMA networks using Nash bargaining solutions and coalitions," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1366–1376, August 2005.

[8] G. Owen, *Game Theory*. New York: Academic, 2001.

[9] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electricity bill for Internet-scale systems," in *Proc. ACM SIGCOMM*, 2009.

[10] "UUSee Inc." http://www.uusee.com/.

[11] Federal Energy Regulatory Commission, "U.S. electric power markets," http://www.ferc.gov/market-oversight/mkt-electric/overview.asp, 2011.

[12] D. Niu, H. Xu, B. Li, and S. Zhao, "Risk management for video-on-demand servers leveraging demand forecast," in *Proc. ACM Multimedia*, 2011.

[13] ——, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proc. IEEE INFOCOM*, 2012.

[14] G. Song and Y. G. Li, "Cross-layer optimization for OFDM wireless network—Part II: Algorithm development," *IEEE Trans. Wireless Commun.*, vol. 4, no. 2, pp. 625–634, March 2005.

[15] S. Boyd and A. Mutapcic, "Subgradient methods," Lecture notes of EE364b, Stanford University, Winter Quarter 2006-2007. http://www.stanford.edu/class/ee364b/notes/subgrad_method_notes.pdf.

[16] "WebGIS," http://www.webgis.com/.

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

[18] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening geographical load balancing," in *Proc. ACM Sigmetrics*, 2011.

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

[20] R. Mazumdar, L. G. Mason, and C. Doulgligeris, "Fairness in network optimal flow control: Optimality of product forms," *IEEE Trans. Commun.*, vol. 39, no. 5, pp. 775–782, May 1991.

[21] H. Xu and B. Li, "Efficient resource allocation with flexible channel cooperation in OFDMA cognitive radio networks," in *Proc. IEEE INFOCOM*, 2010.