Joint Optimization of Bandwidth for Provider and Delay for User in Software Defined Data Centers

Wenxin Li, Heng Qi, Keqiu Li, Ivan Stojmenovic, and Julong Lan

Abstract—In large-scale Internet applications running on geographically distributed datacenters, such as video streaming, it is important to efficiently allocate requests among datacenters. To the best of our knowledge, existing approaches, however, either solely focus on minimizing total cost for provider, or guaranteeing QoS for end-users. In this paper, we apply the software defined network (SDN) controller to enable the central control of the entire network, and propose a joint optimization model to consider high bandwidth utilization for provider and low delay for users. We present the Nash bargaining solution (NBS) based method to model both requirements of provider's high bandwidth utilization and end-users' low delay. Specifically, we formulate the design of request allocation under those requirements as an optimization problem, which is NP-hard. To solve such hard optimization problem, we develop an efficient algorithm blending the advantages of Logarithmic Smoothing technique and the auxiliary variable method. According to the theoretical analysis, we verify the existence and uniqueness of our solution and the convergence of our algorithm. We conduct a large amount of experiments based on real-world workload traces and demonstrate the efficiency of our algorithm compared to both greedy and locality algorithms.

Index Terms—Geo-distributed Datacenters, Request Allocation, NBS, Logarithmic Smoothing

1 Introduction

Large-scale Internet applications, such as video streaming (Netflix), web search (Google), and social network (Facebook), provide service to hundreds of millions of end-users. The enormous, and growing demand of these applications has motivated service providers to deploy geographically distributed datacenters for both reliability and performance reasons. In particular, Netflix is using the Amazon Simple Storage Service (S3) [1] for storing all of its video masters, which are further transcoded to a number of formats, and are then distributed to Content Distribution Networks (CDNs), ready to be served to end-users [2], [3].

The resulting deployment of application leads to a particular important *request allocation* problem, which means that massive end-users' requests across the wide area must be directed to an appropriate datacenter. Intuitively, such request allocation is able to be completed with a simple method that allocates each request to the closest datacenter [4]. Nevertheless, such a naive method can lead to two main consequences from the perspectives of both provider and end-users, respectively.

On the one hand, due to the oversubscription [5], a datacenter may be overloaded when it comes to the peak workload time. Moreover, different datacenters may reach their peak workload at different times [6]. As a consequence, some datacenters may become overloaded, while some datacenters may experience extremely low bandwidth utilization at a particular moment. For datacenters with low bandwidth utilization, it is a waste of both investment and energy for the provider [7]. Meanwhile, the overloading are vulnerable to failures [8], and can further lead to poor application performance. This makes end-users unwilling to use provider's service and further impacts the revenue of provider.

On the other hand, as business increases, there is a rising demand for end-users' requirements for low delay [9]. Unfortunately, with the naive localized allocation strategy, users close to the datacenters experience low delay, while users far away from the datacenters are likely to suffer high delay. Furthermore, as users are increasing to select the closest datacenters, the phenomenon of high delay will become worse when datacenters come to the peak time.

Motivated by these, we focus on the request allocation in geographically distributed datacenters, and jointly optimize bandwidth for provider and delay for end-users. More precisely, some workload can be shifted from overloaded datacenters to the datacenters that have low bandwidth utilization, so as to increase the bandwidth utilization with the overall system per-

- *W. Li, H. Qi, and K. Li are with the School of Computer Science and Technology, Dalian University of Technology, No 2, Linggong Road, Dalian 116023, China. E-mail: liwenxin@mail.dlut.edu.cn, {hengqi, keqiu}@dlut.edu.cn. K. Li is the corresponding author.*
- *I. Stojmenovic is with the School of Electrical Engineering and Computer Science, University of Ottawa, 800 King Edward, Ottawa K1N6N5, ON, Canada. E-mail: ivan@site.uottawa.ca.*
- *J. Lan is with the National Digital Switching System Engineering & Technological Research Center, Zhengzhou, Henan, China.*

spective of the provider. Additionally, the user delay is usually consisting of the transport delay outside datacenters and the response time inside datacenters. This implies that end-users can choose a litter farther datacenter that provides low response time, so as to decrease the delay. Unfortunately, the current request allocation on geo-distributed services, either solely focus on minimizing total cost for provider, i.e., energy cost [10], bandwidth cost [11], or guaranteeing the quality of service (QoS) [12] for end-users. However, such cost savings may not necessarily imply QoS guaranteeing. Moreover, the resource over-provisioning for users' QoS will in turn increase the provider's cost. Hence, this results in a significantly challenge to the joint optimization for provider and end-users.

To make an optimal request allocation, we believe that the global information controlling is needed. Fortunately, such central network control is becoming feasible due to the success of SDN controller [13]. Inspired by this, we construct our multiple datacenters model based on the SDN controller. We propose NBS based method to model provider's requirement of high bandwidth utilization at all datacenters and end-users' requirements of low delay. Specifically, the delay contains the transport delay outside datacenters and the response time inside datacenters. We model the response time based on the G/G/1 queue. Then, we formulate an optimization problem, which is NP-hard. To solve it, we further propose an efficient request allocation algorithm by blending the advantages of the Logarithmic Smoothing technique and the auxiliary variable method. We theoretically prove the existence and uniqueness of our solution and the convergence of our algorithm. Extensive simulation experiments based on real-world workload traces demonstrate that our request allocation algorithm can efficiently improve the bandwidth utilization for provider and reduce the delay for end-users, and outperforms both greedy and locality algorithms.

To sum up, our main contributions of this paper are as follows:

- We focus on the emerging request allocation problem in geographically distributed datacenters, and propose a joint optimization model to consider high bandwidth utilization for provider and low delay for end-users. Specifically, we present Nash bargaining solution based method to model both the requirement of provider's high bandwidth utilization at all datacenters and end-users' low delay.
- We formulate the *request allocation* under those requirements as an optimization problem. Such optimization can be NP-hard. To solve it, we propose an efficient request allocation algorithm by introducing the auxiliary variable method to eliminate inequality, rather than directly applying the Logarithmic Smoothing technique. We perform theoretical analysis to prove the existence and uniqueness of our solution, and the convergence of our algorithm.
- We conduct a large amount of experiments based on real-world workload traces. With the simulation results, we show that our algorithm outperforms the conventional greedy and locality algorithms, and can efficiently improve the bandwidth utilization for provider and reduce the delay for end-users.

The rest of this paper is organized as follows. In Section II, we present the related work. In Section III, we show our model and notation. In Section IV, we present our Logarithmic Smoothing based request allocation algorithm. In Section V, we discuss the experiment evaluation. Finally, in Section VI we conclude this paper.

2 Related Work

Recently, request allocation problem has gained considerable research interest over the past few years. However, existing solutions solely focus on the benefit of either provider or end-users. We review first solutions considering the benefit of end-users. Xu et al. [14] 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. Zhang et al. [15] proposed to optimize traffic engineering across all upstream ISPs, assuming requests are simply allocated to the closest ingress point. Wong et al. [4] supported locality policies based on on-demand network probing. Nevertheless, this locality policy can benefit end-user close to the infrastructure while can lead to poor performance for users far away from the infrastructure. Wu et at. developed a prototype generic workflow system for scientific workflows, and formulate a task scheduling problem to minimize the end-to-end delay under a user-specified financial cost constraint [16]. Their work mainly targets in a single datacenter. This is different from our work since we focus on the request allocation in geographically distributed datacenters. Other solutions consider the benefits for provider. Wendell et al. [12] developed a decentralized request allocation algorithm for cloud services with minimum network costs and balancing considered. Qureshi et al. [10] proposed a simple cost-aware request allocation policy that utilizes geographical diversity of electricity price to preferentially allocate request to datacenters where energy is cheaper. They showed that this sort of optimization can reduce the electricity bill by 13% when assuming power-proportional datacenters. Mathew et al. [17] focused on load balancing for the provider. Mohsenian-Rad et al. [18] considered how request allocation can be used to help with load balancing in the electric grid. Liu et al. [19] considered the effect of request allocation on

providing environment gains by using green energy. Xu et al. [11] developed an efficient request allocation algorithm that considers both bandwidth and electricity cost. Boloor et al. [20] proposed a novel approach of data-oriented dynamic service-request allocation with gi-FIFO scheduling to globally increase the profit charged by cloud computing system. Le et al. [21] considered the joint optimization problem of minimizing carbon emission and electricity cost while their user assignment algorithm does not attempt to minimize the distance between users and the datacenters.

Polverini et al. explore the benefit of electricity price variations across time and locations, and propose an online algorithm for batch jobs scheduling in geographically distributed datacenters [22]. They focused on the energy cost, and do not optimize the network resource for the provider. In addition, they consider the fairness with respect to the queuing delay, which is different from our delay. Xu et al. [23] developed a request allocation algorithm based on the alternating direction method of multipliers (ADMM), which efficiently balance the trade-off between performance and cost. The difference from our scheme is that they take advantage of the cost diversity, and allocate request to a datacenter with low bandwidth price as well as energy price, so as to reduce the cost for the provider. Our focus is shifting workload from overloaded datacenters to datacenters with low utilization. In addition, their delay does not contain the response time inside the datacenters. Gao et al. designed FORTE with access latency, electricity cost, and carbon footprint considered in [24]. Our study differs in the user delay. The user delay we considered consists of both the transport delay outside of datacenters and the response time inside of datacenters, which is different from their access latency in [24].

3 Model and Notation

3.1 The Multiple Datacenters Model

Traditionally, *request allocation* is handled by deploying mapping nodes, which are typically HTTP or authoritative ingress proxies that route end-user requests from a given locale to the appropriate datacenter, or authoritative DNS servers [25] that resolve local queries for the same names of Web sites. Nowadays, with the success of software defined networks, such request allocation is easily and practically handled by the SDN controller that enables the central control of the entire network [13]. In other words, information (i.e., bandwidth utilization, congestion level) of each datacenter can be gathered at the SDN controller. Once massive requests come to the controller, the controller is able to make appropriate and even optimal request allocation strategies with the global information.

Motivated by this, we construct our multiple datacenters model based on the SDN controller. Our model consists of a provider and a set $\mathcal{S} = \{1, 2, \cdots, s\}$ of end-users, which is shown in Fig.1. To handle massive requests, the provider deploys a central SDN controller. We assume that the provider hosts a set of geographically distributed datacenters $\mathcal{M} = \{1, 2, \cdots, m\}$ for better reliability and performance, and provides a set $\mathcal{N} = \{1, 2, \cdots, n\}$ of application instances. Generally, for each application instance $i \in \mathcal{N}$, provider will make several copies. For example, the same encoded content is delivered from multiple CDNs (i.e., Akamai, LimeLight, and Level-3) for the same video with the same quality level in Netflix [3]. Without loss of generality, we assume that the same collection $\mathcal{N}$ is hosted by each datacenter. $c(j)$ is the fixed bandwidth capacity of datacenter j. Let $b(i)$ be the amount of bandwidth to handle one request when serving $i \in \mathcal{N}$. $\forall i \in \mathcal{N}$, we assume that each end-user $k \in \mathcal{S}$ can only request once for a given moment. Let $r(k, i)$ be the binary variable that indicates whether k requests i:

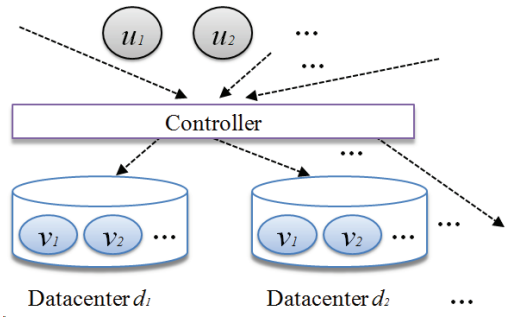


Fig. 1. The multiple datacenters model, where the controller handles request allocation.

$$r(k,i) = \begin{cases} 1 & \text{if } k \text{ requests } i, \\ 0 & \text{otherwise.} \end{cases}$$

Let $x(j, k, i)$ be a binary variable that denotes whether j gets the request from k to i:

$$x(j,k,i) = \begin{cases} 1 & \text{if } r(k,i) \neq 0 \text{ and } j \text{ gets the} \\ & \text{request from } k \text{ to } i, \\ 0 & \text{otherwise.} \end{cases}$$

Important notations used throughout this paper are listed in Table I.

3.2 The NBS Based Model

The above multiple datacenters model contains two entities: the provider and users. The provider has a strong desire for high bandwidth utilization at all datacenters, and users aim to experience low delay. In the following, we detailed study different entities, and apply the Nash bargaining solution [26] to model the benefit of both providers and end-users. The Nash bargaining solution is known as non-zero-sum game,

where players cooperate in this game to achieve a win-win solution.

3.2.1 High bandwidth utilization for provider

Since the service provider has cost a lot to build $\mathcal{M}$ datacenters, the question becomes how the bandwidth resource can be better utilized. Recall that due to the over-subscribed phenomenon and different peak workload in datacenters, it is likely to come to the case where some datacenters are overloaded, and some datacenters experience extremely low bandwidth utilization. An efficient request allocation should shift some workload from overloaded datacenters to datacenters with low bandwidth utilization, so as to increase the bandwidth utilization at all datacenters for the provider.

To achieve this goal, we relate the entire multiple datacenters model to a bargaining market. Requests issued by end-users are considered as commodities in this market. Each datacenter is viewed as a player, and each player makes its decision by bargaining for its desired commodities, so as to maximize its own utility. Note that the bandwidth utilization is the only metric for the player ("datacenter"), and we simply use the bandwidth utilization $p(j)$ as the utility for player j. The bandwidth utilization of datacenter j can be defined as the the total amount of utilized bandwidth divided by the bandwidth capacity of datacenter j. The definition is as follow

$$p(j) = \frac{\sum_{k,i} r(k,i)b(i)x(j,k,i)}{c(j)}.$$

So, in this Nash bargaining game, each player enters this game with a utility function, and aims to maximize its own utility. Players cooperative in this game to achieve a win-win solution, in which the social utility products are maximized. This corresponds to the requirements of the provider, which aims to achieve high bandwidth utilization at all datacenters. Consequently, we model the benefit of provider as the Nash product in the Nash bargaining game, i.e., $\prod_j p(j)$.

3.2.2 Low delay for end-users

Delay is arguably the most important performance metric for end-users. Existing approaches, however, tend to favor end-users closer to the infrastructure. As a consequence, this will result in poor performance for disadvantaged users far away from the infrastructure, and can potentially lead to substantial revenue losses for the provider. In particular, Amazon reports that every 100 ms delay in page load time decreases sales by 1 percent [27]. In this paper, the delay contains two parts. The first part is the *transport delay* between end-users and datacenters. The second part is the *response time* inside of datacenters.

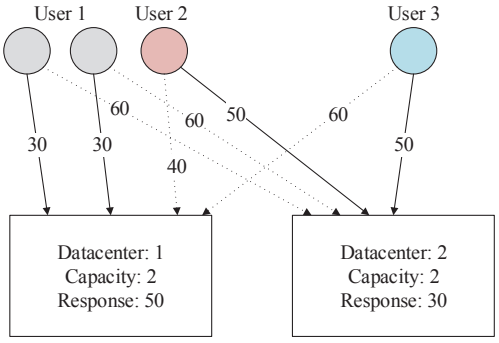


Fig. 2. An illustrative request allocation with two datacenters and three users.

Let $l(j,k)$ denote the *transport delay* between datacenter j and end-user k. To model the *response time*, let's build a set of parallel queues for each application instance inside of datacenters, such as $queue(j,i)$. Requests arrive one by one into its corresponding infinite capacity queue. $\theta(j,i)$ is the mean arrival rate of requests and $\tau(j,i)$ represents the inter-arrival time. Let $\varphi(j,i)$ denote the mean service time. Hence, $\varrho(j,i) = \theta(j,i)\varphi(j,i)$ represents the traffic offered corresponding to the fraction of the time in which the server is busy if each application instance is served by a single server. Let $\sigma^2_{\varphi(j,i)}$ and $\sigma^2_{\tau(j,i)}$ represent the squared coefficient of variation of service time and request inter-arrival time, respectively. Based on G/G/1 queue and the method in Bloch et al. [28], we have the delay of the link between datacenter j and end-user k for application instance i as follows:

$$\delta(j,k,i) = l(j,k) + \varphi(j,i) + \varphi(j,i)\frac{\varrho(j,i)}{1-\varrho(j,i)}\left(\frac{\sigma^2_{\varphi(j,i)} + \sigma^2_{\tau(j,i)}}{2}\right),$$

where the third term is the average waiting time in $queue(j,i)$. Then, the average delay experienced by user k is calculated as follows:

$$t(k) = \frac{\sum_{j,i} r(k,i)x(j,k,i)\delta(j,k,i)}{\sum_i r(k,i)}.$$

Note that there are multiple end-users in the above multiple datacenters model. Each users aims to achieve minimum delay. Clearly, one user's delay is likely to affect other one's. This is because that the traffic in those parallel queues can become heavier as requests arrive one by one. So, we are interested in a Nash bargaining game. In this game, end-users are viewed as players and datacenters are viewed as commodities. Since the only metric in the benefit of end-user is delay, we can simply model the utility of end-user k as $\frac{1}{t(k)}$. Rational players will seek an appropriate datacenter to maximize itself utility. To this end, players cooperate in this game to achieve a win-win solution, where each user can enjoy low

TABLE 1
Notations and Definitions

Notation	Definition
$\mathcal{M}$	the set of datacenters
$\mathcal{S}$	the set of end-users
$\mathcal{N}$	the set of application instances
j	the jth datacenter, $j = 1, 2, \cdots, m$
k	the kth end-user, $k = 1, 2, \cdots, s$
i	the ith application instance, $i = 1, 2, \cdots, n$
$c(j)$	the bandwidth capacity of datacenter j
$b(i)$	the amount of bandwidth to handle one request when serving i
$r(k, i)$	whether end-user k requests i
$x(j, k, i)$	whether datacenter j gets the request from end-user k to i
$p(j)$	the bandwidth utilization of j
$l(j, k)$	the transport delay between datacenter j and end-user k
$\theta(j, i)$	mean arrival rate of requests in $queue(j, i)$
$\varphi(j, i)$	mean service time in $queue(j, i)$
$\sigma^2_{\varphi(j,i)}$	squared coefficient of variation of service time $\varphi(j, i)$
$\sigma^2_{\tau(j,i)}$	squared coefficient of variation of request inter-arrival time $\tau(j, i)$
$\delta(j, k, i)$	the latency of the link between datacenter j and end-user k for instance i
$t(k)$	the average latency experienced by user k
e	$[1, 1, \cdots, 1]^T$
$Diag(x)$	diagonal matrix with m diagonal elements being vector x
$vec(X)$	a row vector of length sn formed by matrix $X \in \mathbb{R}^{s \times n}$
I_n	a $(n \times n)$ identity matrix
μ	the smoothing parameter
θ_μ	reduction ratio for smoothing parameter
γ	the penalty parameter
θ_γ	reduction ratio for penalty parameter

delay. In summary, we can model the benefit of end-users as the social utility products, which are defined by $\prod_k \frac{1}{t(k)}$.

3.3 Joint Optimization Model

Given the above models, the goal of joint optimization is to choose the request allocation policy $x(j, k, i)$ such that provider gains high bandwidth utilization at its datacenters, and each user is enjoying low delay. This is captured by the following optimization problem:

$$\begin{aligned} \max_{x(j,k,i)} \quad & \prod_j p(j) + \prod_k \frac{1}{t(k)} \\ \text{s.t.} \quad & \forall j, \sum_{k,i} r(k,i) b(i) x(j,k,i) \leq c(j), \\ & \forall k, \forall i, \sum_j x(j,k,i) = 1, \\ & \forall j, \forall k, \forall i, x(j,k,i) \in \{0, 1\}. \end{aligned} \tag{1}$$

The optimization problem maximizes its objective function, which represents the sum of benefits of provider and end-users. The first constraint enforces the total workload in datacenter j not to exceed its bandwidth capacity $c(j)$ and the second denotes that each request can only be allocated to one datacenter. We ignore the case $r(k, i) = 0$ while $x(j, k, i) = 1$, since it has no influence on the objective. The last constraint enforces that $x(j, k, i)$ can only take on $\{0, 1\}$.

To have a roundly understanding of our optimization model, we show an illustrative request allocation in Fig. 2. The provider hosts two datacenters. Each datacenter has a capacity for holding two requests. The response time in datacenter 1 is 50, while the response time in datacenter 2 is 30. There are three users. User 1 has two requests and both user 2 and user 3 have one request. The number on each line is the transport delay between users and datacenters. Simply applying localized request allocation, three requests are allocated to datacenter 1 and only one request is directed to datacenter 2. It is clear that one request will be dropped due to the capacity of datacenter 1, while datacenter 2 still has room for one request. Hence, if we choose to move one request from datacenter 1 to datacenter 2, the problem can be resolved. However, one question is that which one should be moved. Based on our optimization model, the optimal solution is to direct the request issued by user 2 to datacenter 2. Such that the utilization

is increased, and each user obtains a 80 transport delay. So, we need to develop an efficient algorithm to compute such an optimal request allocation.

Note that the optimization problem in Equation (1) is a nonlinear integer optimization problem whose variables can only take on integer quantities or discrete values, and is NP-hard [29]. Since there is a finite set of possible solutions, one possibility is simply to examine all such solutions, i.e, perform an exhaustive enumeration of all solutions. However, we might have to perform up to 2^{msn} function evaluations to determine the optimal solution by enumeration. It would seem attractive if the third constraint is dropped, thus Equation (1) could be replaced by a continuous problem. In the following, we present our algorithm based on Logarithmic Smoothing [30] which is a continuous approach that has recently received renewed interest in solving nonlinear integer programming problem.

4 Logarithmic Smoothing Based Request Allocation Algorithm

We first provide a brief primer on Logarithmic Smoothing, which is the corner stone of our algorithm design.

4.1 A Primer on Logarithmic Smoothing

Logarithmic Smoothing [30] has recently received renewed interest in solving nonlinear integer programming problem. The smoothing algorithm solves problem in the form:

$$\begin{aligned} \min \quad & f(z) \\ \text{s.t.} \quad & \mathcal{A}z = \flat, \\ & z \in \{0,1\}^n, \end{aligned}$$

where $z \in \mathbb{R}^n$, $\mathcal{A} \in \mathbb{R}^{p \times n}$ and $\flat \in \mathbb{R}^p$. Logarithmic Smoothing aims to eliminate all the integer constraints. The smoothing function can be defined as:

$$-\sum_{i=1}^{n} \ln z_j - \sum_{i=1}^{n} \ln(1 - z_j).$$

The barrier function is well-defined when $0 < z < e$ and it can efficiently eliminate constraints $z \in \{0,1\}^n$. When the decision variable z_i is 0 or 1, the smoothing function tends to ∞. This implies that a logarithmic barrier function is insufficient to achieve desirable rounding. When the variables are not close to zero or one, extra penalty terms must be added to the objective to ensure a $0-1$ solution. We introduce the penalty term

$$\sum_{i=1}^{n} z_i(1 - z_i).$$

Thus, the actual problem associated with the barrier function and the penalty term becomes:

$$\begin{aligned} \min \quad & f(z) - \mu \left(\sum_{i=1}^{n} \ln z_j + \sum_{i=1}^{n} \ln(1 - z_j) \right) \\ & + \gamma \sum_{i=1}^{n} z_i(1 - z_i) \\ \text{s.t.} \quad & \mathcal{A}z = \flat, \end{aligned}$$

where $\mu > 0$ is the smoothing parameter, and $\gamma > 0$ is the penalty parameter. The coordination of smoothing function and penalty term can efficiently eliminate the constraint $z \in \{0,1\}^n$.

Let's consider a simple example, as follows:

$$\begin{aligned} \min \quad & z^2 \\ \text{s.t.} \quad & z \in \{0,1\}. \end{aligned}$$

The optimal solution for this simple example is $z = 0$. If we solve this problem by Logarithmic Smoothing, the smoothed problem becomes:

$$\min \quad z^2 - \mu \left(\ln z + \ln(1 - z) \right) + \gamma z(1 - z).$$

Fig. 3 shows the original function and smoothed function for different values of smoothing parameter μ and penalty parameter γ. For $\mu = 0.1$ and $\gamma = 1$, the smoothed function closely follows the original function. When μ decreases to 0.1 and γ increases to 10, the smoothed function becomes a concave function and the optimal solution is $z = 0$ which is the same as for the original optimal solution. This implies that an increase in γ and a decrease in μ makes the optimal solution of the smoothed optimization closer to the original solution. Thus, we only need to find suitable μ and γ, such that the nonlinear integer programming can be transformed to an equivalent continuous optimization. Based on the guidelines, we now formally give the design of our algorithm.

4.2 Problem transformation

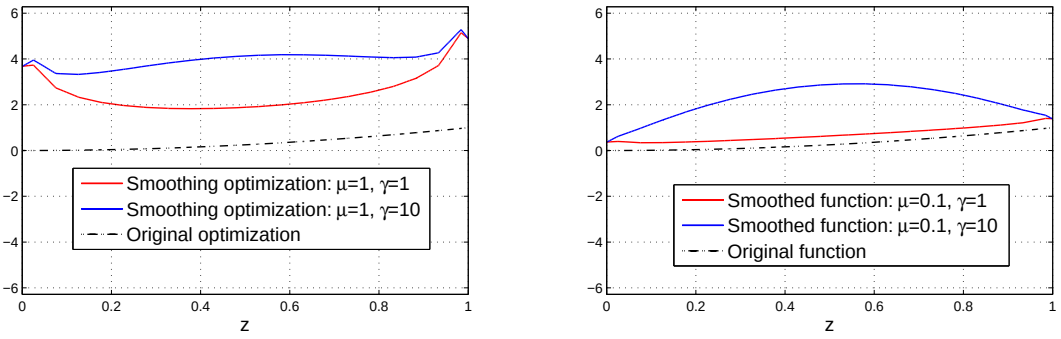
Our joint optimization problem (1) cannot be readily solved using Logarithmic Smoothing, due to the following facts. First, the decision variable $x(j,k,i)$ is constrained by an inequality, rather than an equality constraint required by the Logarithmic Smoothing. Second, the smoothing function should be modified if the inequality constraint is successfully transformed to equality constraint.

In response to these challenges, we introduce a new set of auxiliary variables

$$y = [y(1), \cdots, y(m)]^T$$

which represent the residual bandwidth usage, and thus to transform the inequality constraint to the equality constraint

$$\forall j, \quad \sum_{k,i} r(k,i)b(i)x(j,k,i) + y(j) = c(j).$$



(a) $\mu = 1$ (b) $\mu = 0.1$

Fig. 3. The effect of smoothing the original optimization with varying smoothing parameter μ and penalty parameter γ.

To easy presentation, we write our problem in the matrix form. Let

$$x = [x(1,1,1), \cdots, x(m,s,n)]^T,$$
$$c = [c(1), \cdots, c(m)]^T,$$
$$A_1 = \text{Diag}(vec((r(k,i)b(i))_{s\times n})),$$
$$A_2 = [e_1, \cdots, e_{sn}, e_1, \cdots, e_{sn}, \cdots, e_1, \cdots, e_{sn}],$$

where e_l is the l-th column of I_{sn}. A_1 is a $(m \times msn)$ matrix and A_2 is a $(sn \times msn)$ matrix. Then, we get the matrix form optimization problem with equality constraints only, which is shown as follows:

$$\begin{aligned} & \min \ -\mathcal{G}(x) \\ \text{s.t.} \ & A_1 x + y = c, \\ & A_2 x = e, \\ & x \in \{0,1\}^{msn}, \end{aligned} \tag{2}$$

where $\mathcal{G}(x) = \prod_j p(j) + \prod_k \frac{1}{t(k)}$.

Since the inequality constraints are transformed into equality constraints, the smoothing function to be added should be a modification of that in the primer Logarithmic Smoothing to take into account the bounds on y. Consequently, the new smoothing function is

$$\Phi(x,y) = \sum_j \ln y(j) - \sum_{j,k,i} \left(\ln x(j,k,i) + \ln(1 - x(j,k,i))\right).$$

Again, an optional penalty term can be added to force the binary variables to their bounds. The penalty term is defined as

$$\Psi(x) = \sum_{j,k,i} x(j,k,i)\left(1 - x(j,k,i)\right).$$

Hence, the actual problem we are solving is:

$$\begin{aligned} & \min \ \ \mathcal{G}(x,y) \\ \text{s.t.} \ & A_1 x + y = c, \\ & A_2 x = e, \end{aligned} \tag{3}$$

where $\mathcal{G}(x,y) = -\mathcal{G}(x) + \mu\Phi(x,y) + \gamma\Psi(x)$. $\mu > 0$ is the smoothing parameter and $\gamma > 0$ is a penalty parameter. In general, it is possible to show that the penalty function introduced this way is "exact" in the sense that there exists an unique solution to Equation (3). For completeness, a proof is shown in the following theorem.

Theorem 1 (Existence and Uniqueness): There exists a real $\tilde{\mu} > 0$ such that if $\mu \geq \tilde{\mu}$, then there exists an unique solution to Equation (3).

Proof: Please refer to the Appendix A. □

4.3 Algorithm Design

We design our algorithm based on the guidelines above. The actual problem we are solving is a continuous problem, which enables us to make use of the Lagrange multiplier information. Primal-dual methods [31] have been used to solve programming problems with much success, e.g., minimum cost flow problem. Here, we use a similar approach to deal with our problem in Equation (3), of which the first-order optimality conditions are as follows:

$$\nabla_x \mathcal{G}(x,y) + A_1^T\alpha + A_2^T\beta = 0, \tag{4}$$
$$\nabla_y \mathcal{G}(x,y) + \alpha = 0, \tag{5}$$
$$A_1 x + y = c, \tag{6}$$
$$A_2 x = e, \tag{7}$$

where α and β represent the Lagrange multiplier for the first and second constraint in Equation (3), respectively. Applying Newton's method directly, we get the following equation:

$$\begin{bmatrix} g_1 & g_2 & A_1^T & A_2^T \\ g_3 & g_4 & I_m & 0 \\ A_1 & I_m & 0 & 0 \\ A_2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\alpha \\ \Delta\beta \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}, \tag{8}$$

where

$$g_1 = \nabla_{xx}\mathcal{G}(x,y),$$
$$g_2 = \nabla_{xy}\mathcal{G}(x,y),$$
$$g_3 = \nabla_{yx}\mathcal{G}(x,y),$$
$$g_4 = \nabla_{yy}\mathcal{G}(x,y),$$

and

$$h_1 = -\left(\nabla_x\mathcal{G}(x,y) + A_1^T\alpha + A_2^T\beta\right),$$
$$h_2 = -\left(\nabla_y\mathcal{G}(x,y) + \alpha\right),$$
$$h_3 = -\left(A_1x + y - c\right),$$
$$h_4 = -\left(A_2x - e\right).$$

We observe that Equation (8) can be reduced as follow:

$$\begin{bmatrix} H & A_2^T \\ A_2 & 0 \end{bmatrix}\begin{bmatrix} \Delta x \\ \Delta\beta \end{bmatrix} = \begin{bmatrix} w \\ h_4 \end{bmatrix}, \tag{9}$$

where

$$H = g_1 - g_2A_1 - A_1^Tg_3 + A_1^Tg_4A_1,$$

and

$$w = h_1 - g_2h_3 - A_1^Th_2 + A_1^Tg_4h_3.$$

The rest of the Newton directions can then be obtained by

$$\Delta y = h_3 - A_1\Delta x,$$
$$\Delta\alpha = h_2 - g_3\Delta x - g_4\Delta y.$$

Letting Z be the null-space matrix of A_2, then we have $A_2Z = 0$. Let x_0 be any feasible point such that $A_2x_0 = e$, thus $\Delta x = Zx'$ for some x'. Substituting this into the top part of Equation (9) and multiplying both sides by Z^T, we get

$$Z^THZx' = Z^Tw, \tag{10}$$

where x' could be obtained by using a conjugate gradient method [32]. Next, update $x = x + \rho\Delta x$, where ρ is obtained by performing a linear search method [33]. Further, sequently update y, α and β. Our algorithm is summarized in Algorithm 1, where μ starts with a large value and ends up with one that may be close to zero while γ runs the opposite way. θ_μ and θ_γ are the rate of decrease of μ and the rate of increase of γ, respectively.

The convergence of our algorithm is shown in the following theorem.

Theorem 2 (Convergence) : (1). Let $x(\gamma,\mu)$ be any local minimizer of Equation (3)*. Then, $x(\gamma,\mu) \to \{0,1\}^{msn}$ as $\gamma \to \infty$ and $\mu \to 0$.*

(2). Let $\{\mu_t\}_{t=1}^{\infty}$ be a recursive sequence of positive numbers such that $\lim_{t\to\infty}\mu_t = 0$. Also, suppose that there exists $(x^*, y^*, \alpha^*, \beta^*)$ that satisfies Equation (4) *to Equation* (7)*, then*

$$\lim_{t\to\infty} x(\mu_t) = x^*.$$

Proof: Please refer to the Appendix B. □

Practicality. The dominating operation during one iteration of our algorithm is the conjugate gradient method. The complexity of the conjugate gradient method is $O(\eta \times \sqrt{\kappa})$, where η is the number of non-zero entries of the matrix Z^THZ, and κ is its condition number [34]. Hence, the complexity of our algorithm is $O(\hbar \times \eta \times \sqrt{\kappa})$, where $\hbar$ is the maximum iteration number of our algorithm. Normally, the smoothing parameter decreases from 1000 to 10^{-4}, while the penalty parameter increases from 1 to 10^5 [35]. If we set θ_μ and θ_γ to 0.2 and 5, respectively. Then, the maximum iteration number $\hbar$ is less than 10. In addition to the low value of $\hbar$, the value of η is also low since the matrix Z^THZ can be a spare matrix. Moreover, the condition number κ is usually set to no more than 100, and thus $\sqrt{\kappa}$ is less than 10 [34].

Algorithm 1 Algorithm for Request Allocation

Input: Request, $r(k,i), \forall k, \forall i$;
Bandwidth capacity, $c(j), \forall j$;
Amount of bandwidth for one request, $b(i), \forall i$;
The initial value of smoothing parameter, μ_0;
The initial value of penalty parameter, γ_0;
Smoothing parameter reduction, θ_μ;
Penalty parameter increment, θ_γ;
Output: $x(j,k,i)$;

1: Initialize $\gamma = \gamma_0, \mu = \mu_0$. Letting x_0 be any feasible point such that $A_2x_0 = e$, and then initialize y, α, β;
2: Update x by applying conjugate gradient method [32] for Equation (10) and performing a linesearch method [33];
3: Given Δx, sequently update y, α and β;
4: Update $\mu = \theta_\mu\mu, \gamma = \theta_\gamma\gamma$;
5: Repeat step 2 until convergence or γ and μ to their tolerance values;
6: **return** $x(j,k,i), \forall j, \forall k, \forall i$;

To further investigate the practicality, our algorithm can be triggered in a *laissez-fair* manner. In other words, whenever the bandwidth resource provided by one datacenter cannot sustain requests for applications placed on that datacenter, the request allocation algorithm is triggered. The idea is, if requests can be satisfied under current provision, we will maintain the same even if the request allocation is not the optimal. Once a datacenter reaches its peak workload, our algorithm can be triggered to make the optimal request allocation where high bandwidth utilization for provider and low delay for users are jointly considered.

5 Experiment Evaluation

In this section, we first show the detailed implementation of our simulation. Then, we sequently present the performance of provider and users.

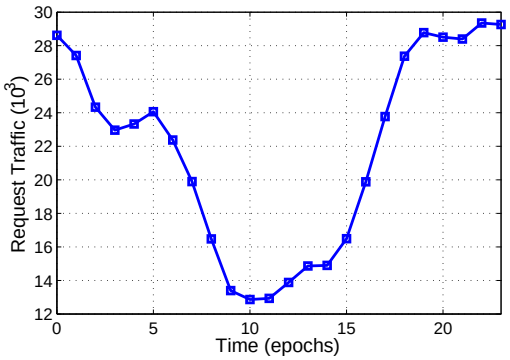


Fig. 4. A 24-epoch sampled request traffic of 100 URLs in the Wikipedia traces [36].

5.1 Simulation Setup

We simulate that the provider deploys 5 (m=5) datacenters and 100 (n=100) application instances in total. In our simulation, there are 500 (s=500) end-users, and we use WikiPedia request traces [36] to represent the request traffic. The dataset we use contains requests issued to Wikipedia from 04:10 AM, September 19, 2007 GMT to 04:11 AM, September 20, 2007 GMT with line record format $< number, timestamp, url >$. We extract requests of 100 URLs for a 24-hour period duration. In each hour, we randomly sample requests of one minute. Such that we have a 24-epoch sampled request traffic of 100 URLs in the Wikipedia traces [36], which is shown in Fig. 4. We believe that the dataset we use faithfully reflect the request distribution for a cloud service, and it is appropriate to use them for the purpose of verifying the performance of our request allocation algorithm.

Since the Wikipedia traces do not contain any end-user information, to emulate the geographical distribution of requests, we randomly split the total request traffic among 500 end-users. Without loss of generality, each datacenter is equipped with the same fixed bandwidth capacity (i.e., 6000 units), and each kind of application instance $i \in \mathcal{N}$ consumes the same amount of bandwidth (i.e. 1 unit) when handling one corresponding request. The transport delay mainly relies on the Internet while we mainly concern about the intra-datacenter network. However, if the transport delay is well measured, we can use it for optimization. For simplicity, we assume that the transport delay between datacenters and end-users is randomly generated from 0 to $50ms$ in our simulations.

Since we use the G/G/1 queue model, each application hosted by a datacenter is actually assumed to be served by a single server. The service time is related to the server speed. For simplicity, we set the mean service time in each queue to 50ms. Certainly, the corresponding squared coefficient of variation $\sigma^2_{\varphi(j,i)}$ equals to 0. In our simulation, we assume that all queues are empty at the first epoch, which means that the waiting time is 0. Hence, the response time equals to the mean service time at the first epoch. Then, at the end of each epoch, we compute the waiting time based on the arrival of requests, and further obtain the response time, which is then to be used for the request allocation in the next epoch. The initial value of μ is set as the maximum eigenvalue of $\nabla^2(-\mathcal{G}(x))$ and the initial value of γ is set as 1% of the absolute value of the minimum eigenvalue of $\nabla^2(-\mathcal{G}(x))$. We set $\theta_\mu = 0.5$, $\theta_\gamma = 2$. The tolerance values of μ and γ are set to be 10^{-4} and 10^4, respectively.

Our algorithm is compared with two algorithms. The first one is a greedy algorithm [37] which is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. In our implementation, the greedy algorithm greedily allocates each request to a datacenter that can provide the lowest delay. The second one is a locality algorithm that allocates each request to the closest datacenter [4]. These two algorithms mainly aim to reduce the delay from the perspectives of end-users, and do not consider benefit for the provider. To illustrate clearly, let RAAF denote our proposed algorithm, GA represent the greedy algorithm, and LA represent the locality algorithm.

5.2 The performance of provider

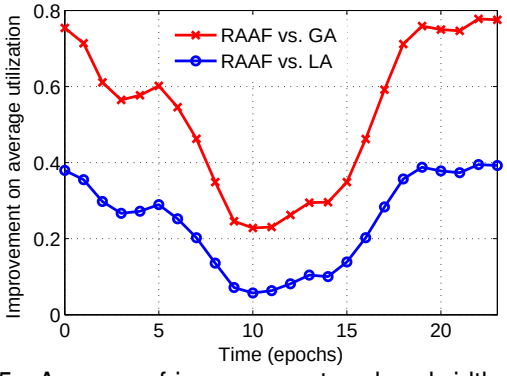


Fig. 5. Average of improvement on bandwidth utilization.

We first evaluate the performance of provider. Fig. 5 shows the improvements on average bandwidth utilization across all datacenters over a 24-epoch period of time. The improvement is computed as the average bandwidth utilization of our RAAF minus that of the compared algorithms (i.e., GA, or LA). As we can see, the average improvement on bandwidth utilization varies with the same trend as the total request pattern in Fig. 4. Specifically, maximum improvement is almost 0.8 and 0.4, compared with GA and LA, respectively. Moreover, when the number of requests increases, improvements on bandwidth utilization become more evident. The reason is that when applying

the greedy algorithm and locality algorithm, partial datacenters are overloaded, while some other datacenters are likely to experience extremely low bandwidth utilization. Moreover, such overloading can be more worse when the number of requests is increasing. Hence, these results verify that our request allocation algorithm can achieve high bandwidth utilization for the provider.

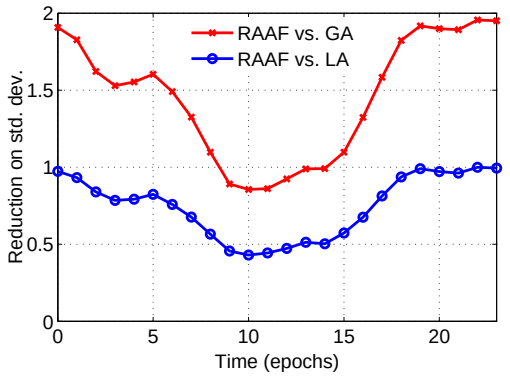


Fig. 6. Reduction in standard deviation of the bandwidth utilization.

In addition to the bandwidth utilization, another important performance for the provider is the standard deviation of bandwidth utilization. We show the reduction in standard deviation of the bandwidth utilization across all datacenters over the 24-epoch period of time in Fig. 6. Similarly, the reduction can be computed as the standard deviation of the compared algorithms (i.e, GA, or LA) minus that of our RAAF. As we know, a reduced standard deviation of bandwidth utilization reflects a more balanced bandwidth utilization. It is obvious that the standard deviation of bandwidth utilization is successfully reduced. More precisely, the maximum reduction of the standard deviation of the bandwidth utilization is almost 2 and 1, compared with the GA and LA, respectively. We can further observe that the reduction on the standard deviation of bandwidth utilization is closely following the total requests pattern in Fig. 4. This implies that the reduction of standard deviation of bandwidth utilization increases as the number of requests increases. These results show that our request allocation is able to mitigate bandwidth resource under-utilization due to imbalanced bandwidth usage across all datacenters.

5.3 The performance of users

For the performance of users, we consider an important metric, the number of requests that is successfully handled. As shown in Fig. 7, it is clear that the number of handled requests has increased by applying our RAAF algorithm, with improvements of more than 2×10^4 and 1×10^4, compared with the GA and LA algorithm, respectively. Moreover, the improvements on the number of handled requests

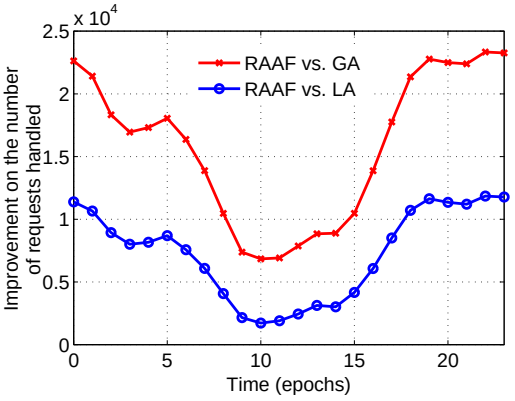


Fig. 7. Improvement on the number of requests successfully handled.

are closely following the improvement on bandwidth utilization. This result confirms what we observed in the improvement on bandwidth utilization, since the more efficient available bandwidth is being utilized, the more requests can be handled.

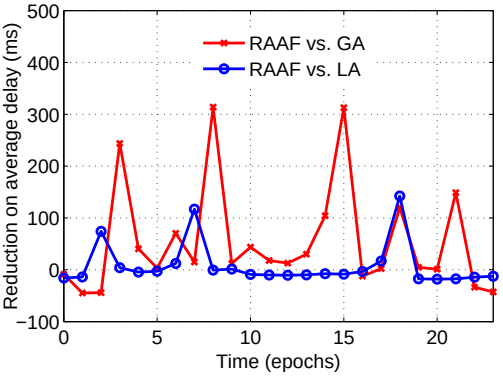


Fig. 8. Average of reduction on user delay.

In addition to reducing the loss of the number of requests, our algorithm can also reduce the user delay. To verify this, we plot the reduction on average user delay across all users over the time in Fig. 8. Note that only the successfully handled requests are used for the computation of user delay in our implementation. We have two main observations from Fig. 8. First, the user delay is successfully reduced at most of the time, compared with the GA algorithm. Moreover, maximum reduction on user delay is more than 300 ms. Second, the average user delay in our RAAF algorithm is the same with that of LA at most of the time, and is even lower than that of LA at some epochs. This implies that our RAAF can successfully reduce the average user delay at some epochs, compared with LA. Hence, we can conclude that apart from the loss user requests, our algorithm can still reduce user delay that is computed by the successfully handled requests.

To further investigate the performance of users, in Fig. 9, we show the reduction in the standard

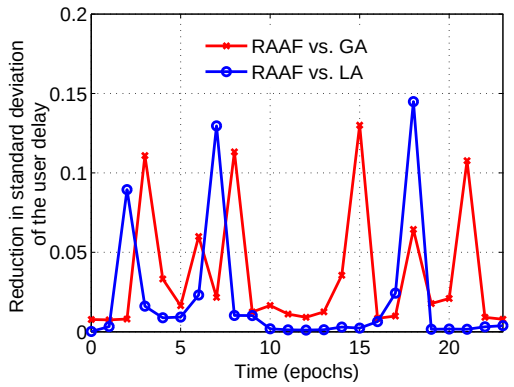


Fig. 9. Reduction in standard deviation of the user delay.

deviation of user delay. A lower standard deviation of user delay reflects a more fair delay among users. It is clear that the standard deviation of user delay is successfully reduced. More precisely, the maximum reduction is almost 0.15, compared with GA. Moreover, the maximum reduction is more than 0.1, compared with LA. This implies that our Nash bargaining game based model can ensure the fairness among users. For the convergence of our algorithm, we direct the interest readers to [30]. To sum up, we believe that our algorithm is practical for real-word problems.

6 Conclusion

Our focus in this paper is the request allocation in geographically distributed datacenters. To efficiently allocate requests, we apply the SDN controller to enable the central control of the network, and jointly consider high bandwidth utilization for provider and low delay for users. Specifically, the provider's requirement of high bandwidth utilization at all datacenters and users' low delay requirements are both modeled based on the Nash bargaining game. Then, we formulate the design of request allocation under those requirements as an optimization problem, which is an integer optimization as well as NP-hard. To efficiently solve such an optimization problem, we propose a request allocation algorithm by introducing auxiliary variables to eliminate inequality constraints, rather than directly applying the Logarithmic Smoothing technique. Theoretical analysis proves the existence and uniqueness of our optimal solution and the convergence of our algorithm. We empirically evaluate our algorithm based on real-world workload traces. The experimental results show that our algorithm can efficiently improve the bandwidth utilization for the provider and reduce the delay for users, compared with both greedy and locality algorithms. As future work, we plan to more thoroughly study the decentralized implementation of request allocation by deploying one controller in each datacenter.

Appendix A
Proof of Theorem 1

Let $x^*(\mu)$ be a solution to

$$\begin{aligned} \min \quad & \Theta(x) = -\mathcal{G}(x) + \mu\Phi(x,y) \\ \text{s.t.} \quad & A_1 x + y = c, \\ & A_2 x = e, \end{aligned} \tag{11}$$

We first show the existence of $x^*(\mu)$. Let

$$\Lambda = \{x : A_1 x + y = c, A_2 x = e\}$$

and j, k, i represent the indices j, k, i. Define

$$\Omega = [1/4, 3/4]^{msn} \bigcap \Lambda.$$

Observe that the Hessian of $\Phi(x,y)$ is a diagonal matrix with diagonal entry being

$$\frac{1}{x_{j,k,i}^2} + \frac{1}{(1-x_{j,k,i})^2}.$$

This implies that there exists a real $\tilde{\mu} > 0$, such that $\Theta(x)$ is a strictly convex function, which has been proved in [38].

Since Ω is a compact set and $\Theta(x)$ is a continuous function on Ω, there exists real number $\pounds_1, \pounds_2$ such that

$$\pounds_1 \leq \Theta(x) \leq \pounds_2$$

for all $x \in \Omega$. Since

$$\lim_{x \to 0^+, 1^-} \Theta(x) = \infty,$$

there exists $0 < \epsilon < \frac{1}{4}$ such that for all $x \in ((0,\epsilon] \bigcup [1-\epsilon, 1))^{msn} \bigcap \Lambda$,

$$\Theta(x) > \pounds_2.$$

Define $\Omega_1 = [\epsilon, 1-\epsilon]^{msn} \bigcap \Lambda$. Again by continuity of $\Theta(x)$ on Ω_1, there exists $z \in \Omega_1$, such that

$$\Theta(z) \leq \Theta(x)$$

for all $x \in \Omega_1$. Moreover, $\Theta(z) \leq \pounds_2$ as $\Omega \subset \Omega_1$. Then, we have

$$\Theta(z) < \Theta(x)$$

for all $x \in (0,1)^{msn} \backslash \Omega_1$. Thus, z is required $x^*(\mu)$. The uniqueness of $x^*(\mu)$ follows from the convexity of $\Theta(x)$.

Now, we show that Equation (2) and the following equation have the same minimizer.

$$\begin{aligned} \min \quad & -\mathcal{G}(x) + \gamma\Psi(x) \\ \text{s.t.} \quad & A_1 x + y = c, \\ & A_2 x = e. \end{aligned} \tag{12}$$

Let $\phi(x) = -\mathcal{G}(x)$. Let $t^{(p)}$ for $p = 1, 2, \cdots, 2^{msn}$ denote the elements of the set $\{0,1\}^{msn}$. $T_{(p)}$ denotes the set

$$\{\dot{x} \in [0,1]^{msn} : \|\dot{x} - t^{(p)}\| < 1/4\}.$$

Suppose $x \in T_{(q)}$ for some q. Then for $t^{(q)}_{j,k,i} = 0$, we have

$$\begin{aligned} x_{j,k,i} &= |x_{j,k,i} - t^{(q)}_{j,k,i}| \\ &\leq \|x - t^{(q)}\| \\ &\leq 1/4, \end{aligned}$$

so that

$$\begin{aligned} |x_{j,k,i} - t^{(q)}_{j,k,i}| &= x_{j,k,i} \\ &\leq 2x_{j,k,i}(1 - x_{j,k,i}). \end{aligned}$$

Similarly, for $t^{(q)}_{j,k,i} = 1$, we have

$$\begin{aligned} 1 - x_{j,k,i} &\leq |x_{j,k,i} - t^{(q)}_{j,k,i}| \\ &\leq |x - t^{(q)}| \\ &\leq |x_{j,k,i} - t^{(q)}_{j,k,i}|, \end{aligned}$$

so that

$$\begin{aligned} |x_{j,k,i} - t^{(q)}_{j,k,i}| &= 1 - x_{j,k,i} \\ &\leq 2x_{j,k,i}(1 - x_{j,k,i}). \end{aligned}$$

By Taylor's theorem, there exists $\tau \in [0,1]^{msn}$ such that

$$\phi(x) = \phi(t^{(q)}) + (\nabla\phi(\tau))^T (x - t^{(q)}).$$

Since $\nabla\phi(x)$ is continuous on the compact set $[0,1]^{msn}$, there exists $L_0 > 0$ such that

$$\begin{aligned} \phi(t^{(q)}) - \phi(x) &\leq |\phi(x) - \phi(t^{(q)})| \\ &= |(\nabla\phi(\tau))^T (x - t^{(q)})| \\ &\leq L_0 \|(x - t^{(q)})\| \\ &= L_0 \sqrt{\sum_{j,k,i} (x_{j,k,i} - t^{(q)}_{j,k,i})^2} \\ &\leq L_0 \sum_{j,k,i} |x_{j,k,i} - t^{(q)}_{j,k,i}| \\ &\leq 2L_0 \sum_{j,k,i} x_{j,k,i}(1 - x_{j,k,i}) \\ &= 2L_0 \Psi(x). \end{aligned}$$

So if $\gamma > 2L_0$,

$$\phi(t^{(q)}) \leq \phi(x) + \gamma\Psi(x)$$

for $x \in T_{(q)}$. Suppose $x \in C = [0,1]^{msn} \backslash \left(\bigcup_{p=1}^{2^{msn}} T_{(p)} \right)$. There exists L_1, L_2, such that $\phi(x) \geq L_1$ and $\Psi(x) \geq L_2$ for all $x \in C$. In particular, $L_2 > 0$ since $x \neq t^{(p)}$ for all p. This implies for all $x \in C$,

$$\begin{aligned} \phi(x) + \gamma\Psi(x) &\geq L_1 + \gamma L_2 \\ &\geq \phi(t^{(p)}) \end{aligned}$$

for all p if $\gamma \geq (L_3 - L_1)/L_2$ where $L_3 = \max_p \phi(t^{(p)})$. Thus, if $\gamma > \max\{2L_0, (L_3 - L_1)/L_2\}$, we have $L_3 \leq \phi(x) + \gamma\Psi(x)$. Letting

$$p' = \arg\min_p \phi(t^{(p)})$$

and

$$\gamma^* = \max\{2L_0, (L_3 - L_1)/L_2\},$$

we also have

$$\begin{aligned} \phi(t^{(p')}) + \gamma\Psi(t^{(p')}) &= \phi(t^{(p')}) \\ &\leq \phi(x) + \gamma\Psi(x) \end{aligned}$$

for all $x \in [0,1]^{msn} \bigcap \Lambda$ if $\gamma > \gamma^*$. $t^{(p')}$ is the same minimizer of Equation (2) and Equation (12). Note that $\mathcal{G}(x,y) = \Theta(x) + \gamma\Psi(x)$. Thus, the theorem follows from the observation that Equation (3) and Equation (11) have the same minimizer.

APPENDIX B
PROOF OF THEOREM 2

(1): Let $x(\gamma)$ be a solution to Equation (12). Obviously,

$$\lim_{\mu \to 0} x(\gamma, \mu) = x(\gamma).$$

Observe that Equation (5) is a sequence of penalty subproblems for Equation (2). By the theoretical analysis of theorem 1, we have $x(\gamma) \to \{0,1\}^{msn}$ for sufficiently large γ. Thus, $x(\gamma, \mu) \to \{0,1\}^{msn}$ as $\gamma \to \infty$ and $\mu \to 0$.

(2): We can simply get following equations:

$$\nabla_{x^*} \mathcal{G}(x^*, y^*) + A_1^T \alpha^* + A_2^T \beta^* = 0, \tag{13}$$
$$\nabla_{y^*} \mathcal{G}(x^*, y^*) + \alpha^* = 0, \tag{14}$$
$$A_1 x^* + y^* = c, \tag{15}$$
$$A_2 x^* = e. \tag{16}$$

And for each t,

$$\nabla_{x(\mu_t)} \mathcal{G}(x(\mu_t), y(\mu_t)) + A_1^T \alpha(\mu_t) + A_2^T \beta(\mu_t) = 0, \tag{17}$$
$$\nabla_{x(\mu_t)} \mathcal{G}(x(\mu_t), y(\mu_t)) + \alpha(\mu_t) = 0, \tag{18}$$
$$A_1 x(\mu_t) + y(\mu_t) = c, \tag{19}$$
$$A_2 x(\mu_t) = e. \tag{20}$$

From Equation (15), Equation (16), Equation (19), Equation (20) we can get

$$\begin{bmatrix} A_1 & I_m \\ A_2 & 0 \end{bmatrix} \begin{bmatrix} x(\mu_t) - x^* \\ y(\mu_t) - y^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{21}$$

Also from Equation (13), Equation (14), Equation (17), Equation (18) we have

$$\begin{bmatrix} H_1 \\ H_2 \end{bmatrix} + \begin{bmatrix} A_1 & I_m \\ A_2 & 0 \end{bmatrix}^T \begin{bmatrix} \alpha(\mu_t) - \alpha^* \\ \beta(\mu_t) - \beta^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{22}$$

where

$$H_1 = \nabla_{x(\mu_t)} \mathcal{G}(x(\mu_t), y(\mu_t)) - \nabla_{x^*} \mathcal{G}(x^*, y^*),$$

and

$$H_2 = \nabla_{y(\mu_t)} \mathcal{G}(x(\mu_t), y(\mu_t)) - \nabla_{y^*} \mathcal{G}(x^*, y^*).$$

Premultiplying both sides of Equation (22) by

$\begin{bmatrix} x(\mu_t) - x^* \\ y(\mu_t) - y^* \end{bmatrix}^T$ and using Equation (15), Equation (19) and Equation (21), we obtain $(x(\mu_t) - x^*)^T(H_1 + H_2) = 0$. This implies that $\lim_{t\to\infty} x(\mu_t) = x^*$.

ACKNOWLEDGMENTS

The authors would also like to sincerely thank the editors and anonymous reviewers for their thoughtful suggestions and constructive comments, which have greatly helped and inspired us to improve the quality of this paper. This work is supported by the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61225010), the National Basic Research Program of China (973 Program) (Grant nos 2012CB315901 and 2013CB329104), the State Key Program of National Natural Science of China (Grant No. 61432002), NSFC (Grant nos. of 61173161, 61173162, 61272417 and 61370199), and the Fundamental Research Funds for the Central Universities (Grant No. DUT13ZD101).

REFERENCES

[1] "Amazon web services," http://aws.amazon.com.
[2] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang, "Unreeling netflix: Understanding and improving multi-CDN movie delivery," in *Proceedings of IEEE INFOCOM*, 2012, pp. 1620–1628.
[3] A.Cockcroft, "Netflix in the cloud," http://velocityconf.com/velocity2011/public/schedule/detail/17785, 2011.
[4] Bernard Wong and Emin Gün Sirer, "Closestnode.com: an open access, scalable, shared geocast service for distributed systems," *Operating Systems Review*, vol. 40, no. 1, pp. 62–64, 2006.
[5] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron, "Towards predictable datacenter networks," in *Proceedings of ACM SIGCOMM*, Toronto, ON, Canada, 2011.
[6] Nikolaos Laoutaris, Michael Sirivianos, Xiaoyuan Yang, and Pablo Rodriguez, "Inter-datacenter bulk transfers with netstitcher," in *Proceedings of ACM SIGCOMM*, Toronto, Canada, 2011.
[7] Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.
[8] Aameek Singh, Madhukar Korupolu, and Dushmanta Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in *Proceedings of ACM/IEEE conference on Supercomputing*, 2008, p. 53.
[9] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Proceedings of Springer ICA3PP*, 2010, pp. 13–31.
[10] Asfandyar Qureshi, Rick Weber, Hari Balakrishnan, John V. Guttag, and Bruce M. Maggs, "Cutting the electric bill for internet-scale systems," in *Proceesings of ACM SIGCOMM*, 2009, pp. 123–134.
[11] Hong Xu and Baochun Li, "Cost efficient datacenter selection for cloud services," in *Proceedings of IEEE International Conference on Communications in China (ICCC)*, 2012, pp. 51–56.
[12] Patrick Wendell, Joe Wenjie Jiang, Michael J Freedman, and Jennifer Rexford, "Donar: decentralized server selection for cloud services," in *Proceedings of ACM SIGCOMM*, 2010, vol. 40, pp. 231–242.
[13] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al., "B4: Experience with a globally-deployed software defined wan," in *Proceedings of ACM SIGCOMM*, Hong Kong, 2013.
[14] Hong Xu and Baochun Li, "A general and practical datacenter selection framework for cloud services," in *Proceedings of IEEE CLOUD*, 2012, pp. 9–16.
[15] Zheng Zhang, Ming Zhang, Albert G. Greenberg, Y. Charlie Hu, Ratul Mahajan, and Blaine Christian, "Optimizing cost and performance in online service provider networks," in *Proceedings of USENIX NSDI*, 2010, pp. 33–48.
[16] Chase Qishi Wu, Xiangyu Lin, Dantong Yu, Wei Xu, and Li Li, "End-to-end delay minimization for scientific work?ows in clouds under budget constraint," *IEEE Transactions on Cloud Computing*, 2014.
[17] Vimal Mathew, Ramesh K. Sitaraman, and Prashant J. Shenoy, "Energy-aware load balancing in content delivery networks," in *Proceedings of IEEE INFOCOM*, 2012, pp. 954–962.
[18] Alberto Leon-Garcia Amir-Hamed Mohsenian-Rad, "Coordination of cloud computing and smart power grids," in *Proceedings of IEEE SmartGridComm*, 2010, pp. 368–372.
[19] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H Low, and Lachlan LH Andrew, "Greening geographical load balancing," in *Proceedings of ACM SIGMETRICS*, 2011, pp. 233–244.
[20] Keerthana Boloor, Rada Chirkova, Yannis Viniotis, and Tiia Salo, "Dynamic request allocation and scheduling for context aware applications subject to a percentile response time sla in a distributed cloud," in *Proceedings of IEEE CloudCom*, 2010, pp. 464–472.
[21] Kien Le, Ricardo Bianchini, Thu D. Nguyen, Ozlem Bilgir, and Margaret Martonosi, "Capping the brown energy consumption of internet services at low cost," in *Proceedings of IEEE Green Computing Conference*, 2010, pp. 3–14.
[22] Marco Polverini, Antonio Cianfrani, Shaolei Ren, and Athanasios V. Vasilakos, "Thermal-aware scheduling of batch jobs in geographically distributed data centers," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 71–84, 2014.
[23] Hong Xu and Baochun Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proceedings of IEEE INFOCOM*, 2013, pp. 854–862.
[24] Peter Xiang Gao, Andrew R Curtis, Bernard Wong, and Srinivasan Keshav, "It's not easy being green," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 211–222, 2012.
[25] "DynDNS," http://dyn.com/dns/.
[26] Abhinay Muthoo, *Bargaining theory with applications*, Cambridge University Press, 1999.
[27] Ron Kohavi and Roger Longbotham, "Online experiments: Lessons learned," *IEEE Computer*, vol. 40, no. 9, pp. 103–105, 2007.
[28] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor Shridharbhai Trivedi, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*, John Wiley & Sons, 2006.
[29] Eranda Cela, *The quadratic assignment problem: theory and algorithms*, vol. 281, Kluwer Academic Publishers Dordrecht, 1998.
[30] Walter Murray and Kien-Ming Ng, "An algorithm for nonlinear optimization problems with binary variables," *Computational Optimization and Applications*, vol. 47, no. 2, pp. 257–288, 2010.
[31] Layne T Watson, "Theory of globally convergent probability-one homotopies for nonlinear programming," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 761–780, 2001.
[32] Gene H Golub and Charles F Van Loan, *Matrix computations*, vol. 3, JHU Press, 2012.
[33] Anders Forsgren and Walter Murray, "Newton methods for large-scale linear equality-constrained minimization," *SIAM Journal on Matrix Analysis and Applications*, vol. 14, no. 2, pp. 560–587, 1993.
[34] Jonathan Richard Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," 1994.
[35] Kien-Ming Ng, *A continuation approach for solving nonlinear optimization problems with discrete variables*, Ph.D. thesis, stanford university, 2002.
[36] "WikiBench," http://www.wikibench.eu.
[37] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, et al., *Introduction to algorithms*, vol. 2, MIT press Cambridge, 2001.
[38] Dimitri P Bertsekas, "Nonlinear programming," 1999.

Wenxin Li received the B.E. degree from the School of Computer Science and Technology, Dalian University of Technology, China, in 2012. Currently, he is a Ph.D. candidate in the School of Computer Science and Technology, Dalian University of Technology, China. His research interests include datacenter networks and cloud computing.

Heng Qi was a Lecture at the School of Computer Science and Technology, Dalian University of Technology, China. He got bachelor's degree from Hunan University in 2004 and master's degree from Dalian University of Technology in 2006. He servered as a software engineer in GlobalLogic-3CIS from 2006 to 2008. Then he got his doctorate degree from Dalian University of Technology in 2012. His research interests include computer network, multimedia computing, and mobile cloud computing. He has published more than 20 technical papers in international journals and conferences, including ACM Transactions on Multimedia Computing, Communications and Applications (ACM TOMCCAP) and Pattern Recognition (PR).

Keqiu Li received the bachelors and masters degrees from the Department of Applied Mathematics at the Dalian University of Technology in 1994 and 1997, respectively. He received the Ph.D. degree from the Graduate School of Information Science, Japan Advanced Institute of Science and Technology in 2005. He also has two-year postdoctoral experience in the University of Tokyo, Japan. He is currently a professor in the School of Computer Science and Technology, Dalian University of Technology, China. He has published more than 100 technical papers, such as IEEE TPDS, ACM TOIT, and ACM TOMCCAP. He is an Associate Editor of IEEE TPDS and IEEE TC. He is a senior member of IEEE. His research interests include internet technology, data center networks, cloud computing and wireless networks.

Ivan Stojmenovic was editor-in-chief of IEEE Transactions on Parallel and Distributed Systems (2010-3), and is founder of three journals. He is editor of IEEE Transactions on Computers, IEEE Network, IEEE Transactions on Cloud Computing and ACM Wireless Networks and steering committee member of IEEE Transactions on Emergent Topics in Computing. Stojmenovic has top h-index in Canada for mathematics and statistics, and has > 16000 citations. He received five best paper awards. He is Fellow of IEEE, Canadian Academy of Engineering and Academia Europaea. He received Humboldt Research Award in Germany and Royal Society Research Merit Award in UK.

Julong Lan is currently the chief engineer, and professor in National Digital Switching System Engineering & Technological Research Center. He is also the committee of the China's Next Generation Broadcast Networks expert committee. He is the PI of the 973 major program "Research on Reconfigurable Information Communication Network Architecture". His main research interests include network architecture and protocol design.