Non-cooperative power and latency aware load balancing in distributed data centers

Rakesh Tripathi, S. Vignesh and Venkatesh Tamarapalli
Department of CSE, IIT Guwahati, Assam, India.
{t.rakesh, s.vignesh, t.venkat}@iitg.ernet.in
Anthony T. Chronopoulos
Department of Computer Science,University of Texas at San Antonio,USA.
Anthony.Chronopoulos@utsa.edu
Hajar Siar
Faculty of Electrical and Computer Engineering, Semnan University, Semnan, Iran.
h_siar@semnan.ac.ir

***Abstract*—In this paper we propose an algorithm for load balancing in distributed data centers based on game theory. We model the load balancing problem as a non-cooperative game among the front-end proxy servers. We model the operating cost associated with a data center as a weighted linear combination of the energy cost and the latency cost. We propose a non-cooperative load balancing game with the objective of minimizing the operating cost and obtain the structure of Nash equilibrium. Based on this structure, a distributed load balancing algorithm is designed. We compare the performance of the proposed algorithm with the existing approaches. Numerical results demonstrate that the solution achieved by the proposed algorithm approximates the global optimal solution in terms of the cost and it is also ensures fairness among the users.**

***Index Terms*—Distributed data centers, game theory, front-end proxy servers, optimization of combined energy and latency cost**

I. Introduction

In the recent past, there has been a tremendous growth in the demand for Internet-scale services like Web search, video streaming, and online gaming. Almost all of them are implemented on geographically distributed data centers. A geo-distributed data center is a collection of small geographically distributed data centers which provides high reliability and performance. In an operational data center there are several front-end proxy servers that map the client requests to the appropriate data centers. These proxy servers use different objectives like maximizing the utilization, minimizing the latency or the operating cost to map the requests. Since the client requests are handled in a distributed manner by the front-end nodes, there is a need for an efficient, distributed algorithm to determine the mapping strategy, also termed load balancing strategy.

In general, a load balancing strategy can be classified as static, semi-static or dynamic. In the static approach [1], all the information necessary for the decision making is available before the execution of the algorithm and it remains constant during the execution. In the semi-static approach [2], the required information is available at the beginning of each time step or a well defined point. For example, the load on the system and the cost of serving the same is available with reasonable accuracy just before the time slot. In the dynamic approach [3] the information is not known till the point of execution and it might change during the course of execution.

Load balancing algorithms can also be classified as centralized or decentralized. In a centralized approach, one node in the system collects the information necessary decide the strategy for load balancing. In the decentralized approach, multiple nodes participate to decide the load balancing strategy, either cooperatively or independently. Decentralized approach (cooperative or otherwise) is resilient and scalable compared to the centralized one, particularly for large-scale distributed data centers. In the cooperative approach, all the nodes form a coalition to decide the optimal solution, which improves the performance [4]. In the non-cooperative approach, each node maximizes/minimizes the utility independently, but eventually reach an equilibrium [5]. Non-cooperative game solutions to the load balancing problem are computationally efficient and can be implemented in a distributed manner across all the participating players (in our case, the front-end proxies) [6], [7].

The operational cost of a data center is influenced by factors such as electricity prices, server/data center failure, green energy availability, and client demand. Therefore, load balancing is challenging as the designed strategy must consider the spatio-temporal variation in these factors to minimize the operating cost. Most of the literature on load balancing in distributed data centers [8], [9] considered minimizing the operating cost, which is profitable for the operators, but has ignored the users' perspective. Users consuming the same resources may pay the same price, but experience variable delays. For business continuity, ensuring fairness in service latency across the requests from different clients is also important. The load balancing algorithms in the literature designed to provide fairness, did not consider the energy cost. Therefore, we consider the linear combination of operating cost (or energy cost) and revenue loss due to latency (including the network and queuing delays) as the objective function.

In a distributed data center, the user requests are served by the front-end proxy servers independent of each other.

Each proxy server prefers to get its requests served by the data center first to minimize the service delay. In order to model this selfish nature in distributed load balancing, we use the non-cooperative game theory approach. We propose a game-theoretic distributed load balancing algorithm, that is executed across a finite number of front-end proxy servers. The objective of the game is to minimize the sum of the energy cost and the revenue loss due to delayed service. The proposed approach reduces the cost compared to an approach that only minimizes the operating cost as in [10].

In summary, the main contributions of this work are as follows.

- For the first time, we model the load balancing in distributed data centers as a non-cooperative game among the front-end proxies. We consider the spatio-temporal variation in the electricity price, the offered load, and the availability in the model. We prove that the Nash equilibrium is the solution of this game, which is guaranteed to exist since the proposed objective function is continuous, convex and increasing [5], [11]. We characterize the Nash equilibrium and propose a distributed algorithm for computing the same.
- We evaluate the performance of the non-cooperative game theoretic algorithm (abbreviated as NCG) along with the existing ones, such as the proportional scheme and the global optimal scheme, using real-world data. The proposed NCG algorithm shows better fairness (in service latency) at a comparable cost.

The rest of this paper is organized as follows. Section II presents the literature on game theoretic approaches related to our work. In Section III we present the architecture and the cost model used in the formulation. We present the non-cooperative game model and derive the structure of Nash equilibrium in Section IV. A distributed algorithm to solve the game and its analysis is given in Section V. In Section VI, we present numerical results that demonstrate the performance of our algorithm against the optimal approach. Section VII concludes the paper and proofs of various theorems are presented in Appendix.

II. Related Work

In this section we review the literature on cost-aware load balancing and other game theory-based approaches in distributed systems, which are relevant to our work.

a) Electricity price-aware load balancing: The problem of load balancing requests across different data centers leveraging electricity price variability has been addressed in [8], [9], where the requests are routed to data centers operating with cheaper electricity. The authors of [10], considered availability of green energy sources, along with the electricity prices while choosing the data center. All these works used an optimization framework that is solved centrally. The work in [12] proposed a decentralized algorithm for load balancing considering the server availability, but did not optimize the latency or the operating cost at the data center. The work in [13] proposed three distributed algorithms for load balancing based on Gauss-Seidel, gradient projection and gradient descent methods. These methods are all known to be computationally expensive and may not be effective for dynamic load balancing.

b) Cooperative game theoretic approaches: Numerous efforts have been made for load balancing and resource management using cooperative game theory for example in communication network [14], distributed systems(DS) [15], [16], and grid computing [17]. In [14], the authors presented an excellent summary on coalition games and their use in wireless communication networks. They listed the applications of game theory for distributed resource allocation, congestion control, power control, and spectrum sharing in cognitive radio networks. The work in [15] modeled the static load balancing in a single class DS with heterogeneous computers, as a cooperative game among the nodes (being assigned user jobs). It is shown that the Nash bargaining solution (NBS) gives an optimal solution and it is also fair. A similar approach was used in [16], where the communication cost for job transfer between the nodes was also considered. The authors of [17] addressed the problem of job allocation in a grid environment. They presented a distributed algorithm based on the structure of the NBS to minimize the average job completion time. In [18], the authors proposed a data center selection framework to ensure latency fairness across clients using the NBS, and presented algorithms based on dual decomposition and sub gradient methods. All these works have studied the structure of NBS for load balancing in architectures not similar to the ones used in geo-distributed data centers and they also used different objectives. In this paper, we use non-cooperative game theory to account for the selfish nature in decentralized load balancing and due to its computational efficiency.

c) Non-cooperative game theoretic approaches: The authors of [19] proposed a price bidding strategy for multiple users competing for servers in a cloud environment, where non-cooperative game is used with the objective of maximizing the net profit while being time efficient. The authors of [20] proposed an auction-based online mechanism for VM provisioning in cloud environment. The main drawback of using these approaches is the slow response time, as the bidder has to wait for auction clearing. In this paper, we address the problem of online load balancing in Internet data centers, where the requests are small in size to be served with minimum latency.

The authors of [7] addressed the problem of load balancing in a DS consisting of n computers (or nodes) shared by m client regions (classes). They proposed a distributed algorithm for load balancing in distributed system using non-cooperative game theory. The work in [21], also considered the fact that the jobs submitted to a heavily-loaded computer are transferred to other lightly-loaded computers. Their objective function included communication delay along with the computational delay. In their model, a player has a selfish interest of optimizing his/her own expected response time. The problem is formulated as a non-cooperative game among the users, who try to minimize the expected response time of their own jobs. Nash equilibrium is introduced as the solution structure and a distributed algorithm is proposed for computing the

same. Similarly, the authors of [6] and [22] have applied non-cooperative game theory in a grid environment and future Internet architectures, respectively.

Fundamentally, the geo-distributed data centers differ from other communication systems, computational grids, and cloud environments in terms of the basic objectives, their characteristics and the way the resources are assigned to jobs (requests). Therefore, it is not possible to directly apply earlier literature for the architecture considered in this paper. Though using non-cooperative game theory for load balancing in distributed data centers is similar to the work in [7], our work is novel in the following aspects. Our architectural setup is different (three-tier architecture) and our algorithm is driven by the satisfy the goals of both front-end proxy servers (latency minimization) and the data center operators (energy cost minimization). We take into account minimizing the energy cost in meeting the demand, besides ensuring fairness across the user-perceived latency. Earlier load balancing algorithms that tried to provide fairness did not consider minimizing the cost of energy consumed. To the best of our knowledge, we are the first to propose a load balancing strategy for distributed data centers using non-cooperative game theory, whose objective is not only to minimize the latency but also to minimize the operating cost (or energy cost).

III. System Model

We assume that a distributed data center, as shown in Fig.1, consists of set of n data centers denoted by S and a set of m front-end proxy servers denoted by U. For simplicity, we consider the client regions to be co-located with the front-end proxy servers (we use the client region and front-end proxy interchangeably). Each data center houses m_s number of servers and is modeled as an M/M/1 queueing system. It is characterized by an expected (or average) processing rate $\mu_s = m_s\mu$, where μ is the processing rate of each server, $s = 1, ...n$. The arrival rate of demand from each client region is represented by L_u, $u = 1, ...m$. A front-end proxy server maps the client requests to multiple data centers, where λ_{su} is the portion of the demand mapped from a client region u to a data center s.

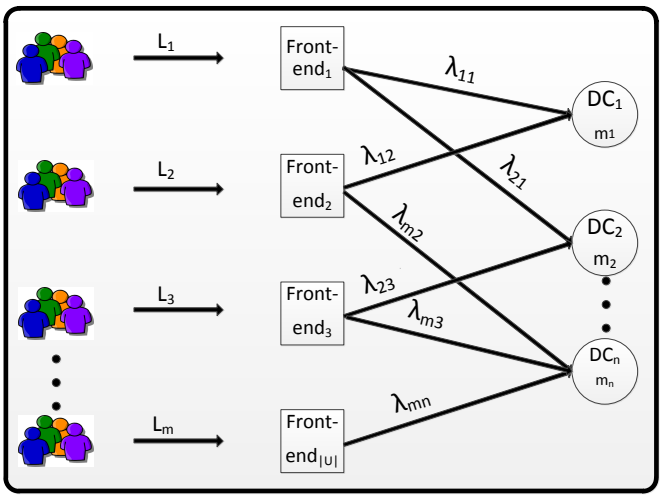

Fig. 1: Architecture of a distributed data center

To ensure that all the requests are served, the following conditions must be satisfied.

$$\sum_s \lambda_{su} = L_u \qquad \forall u \tag{1}$$

Since the number of requests served cannot be negative,

$$\lambda_{su} \geq 0 \qquad \forall s, u \tag{2}$$

To ensure the stability of the system, the following constraints need to be satisfied.

$$\sum_u \lambda_{su} < \mu_s \quad \forall s \tag{3}$$

Power Consumption Cost: As reported in [23], the power consumption of a server varies linearly with the load. Let P_{idle} be the average power drawn by the server in idle condition and P_{peak} be the power consumed at the peak utilization.

E_s is the power usage effectiveness(PUE) of a data center s, defined as the ratio of the total power entering the data center to the power used by the computing equipment.

The utilization of a data center, denoted by η, is given by

$$\eta = \sum_u \lambda_{su}/\mu_s \; m_s \tag{4}$$

The data center power consumption includes three components: the power consumed by idle servers, denoted by $m_s(P_{idle})$, the power consumed by the servers operating at a utilization η, denoted by $m_s(P_{peak} - P_{idle})\eta$, and the power consumed by the cooling and auxiliary equipment, denoted by $(m_s(E_s - 1)P_{peak})$. Therefore, the total power consumed at a data center location $s \in S$ can be written as [24],

$$P_s = m_s(P_{idle} + (E_s - 1)P_{peak}) + \sum_u \lambda_{su} m_s(P_{peak} - P_{idle})/\mu_s \; m_s \tag{5}$$

Eq. (5) can also be expressed as an affine function of the total workload at a data center as

$$P_s = \frac{(P_{peak} - P_{idle}) \sum_u \lambda_{su}}{\mu_s} + \epsilon' \tag{6}$$

where $\epsilon' = m_s(P_{idle} + (E_s - 1)P_{peak})$.

Due to the recent advances in hardware, DVFS scaling, and processor scheduling, it is possible to keep the power consumption proportional to the utilization [25]. Due to efficient cooling systems, PUE has also been significantly lowered [26]. The state-of-the-art average PUE could be as low as 1.02 [27]. When PUE is close to unity and P_{idle} is very low compared to P_{peak} [28], ϵ' is very low compared to the actual power consumed. Further, ϵ' is independent of the utilization and gets canceled in load calculation (see Appendix for details). Therefore, we assume $\epsilon' = 0$ as also done in [29], [30].

This assumption makes Eq.(6) to be linear in terms of the load. Thus, we can easily determine the power consumed to serve the requests λ_{su} from a client region u at a data center s as

$$P_{su} = \frac{(P_{peak} - P_{idle})\lambda_{su}}{\mu_s} \tag{7}$$

Given the load λ_{su} and a unit electricity price ρ_s, the cost incurred due to the power consumed at a data center (also termed the operating cost in this paper), is given by

$$\Theta_{su} = \rho_s \frac{(P_{peak} - P_{idle})}{\mu_s}\lambda_{su} \tag{8}$$

$$\Theta_{su} = \theta_s \lambda_{su} \tag{9}$$

where $\theta_s = \rho\frac{(P_{peak}-P_{idle})}{\mu_s}$ is constant.

Delay Cost: Let d_{su} be the propagation delay between the data center s and a client region u. We assume an M/M/1 model for the queue at the proxy, so that the expected average queuing delay at a data center s for a request from a client region u is given by

$$D_{su} = \frac{1}{\mu_s - \sum_u \lambda_{su}} \tag{10}$$

Therefore, the total delay incurred by a request, δ_{su} is given by

$$\delta_{su} = d_{su} + D_{su} \tag{11}$$

Since we assumed Web workload, the transmission delay is neglected. We use a linear model for the loss in revenue due to the delay incurred [13]. The cost incurred due to the delay experienced by a request λ_{su} of client region u at a data center s, denoted by Δ_{su} is given by

$$\Delta_{su} = \beta\lambda_{su}\delta_{su} \tag{12}$$

$$= \beta\lambda_{su}(D_{su} + d_{su}) \tag{13}$$

$$= \beta(\frac{\lambda_{su}}{\mu_s - \sum_u \lambda_{su}} + d_{su}\lambda_{su}) \tag{14}$$

where β is a constant.

IV. Load Balancing as a non-cooperative game

In this section, we model the load balancing problem as a non-cooperative game. The proposed approach combines the inherent efficiency of the centralized approach and the fault-tolerant nature of the decentralized approach. In a non-cooperative game there could be finite (or infinite) number of players who try to minimize/maximize their objective independently, but eventually reach an equilibrium. For finite number of players, this equilibrium is called Nash equilibrium whereas for infinite number of players this equilibrium is called a Wardrop equilibrium [5].

In distributed load balancing, the problem at hand for each front-end proxy server is to determine λ_{su}, which we model as a non-cooperative game among the front-end proxies. We define the vector $\boldsymbol{\lambda_u} = (\lambda_{1u}, \lambda_{2u}, ..., \lambda_{nu})$ as the load balancing strategy of a user $u, u = 1, 2, ..., m$ and the vector $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_m)$ as the load balancing strategy profile of the entire system.

Objective function: The objective function we use has two components, the power consumption cost defined in Eq. 9 and the delay cost defined in Eq. 14. Let Ψ_u denote the expected Cost Incurred for a Client region (CIC). We define the objective function for a front-end proxy serving a client region u as

$$\Psi_u(\boldsymbol{\lambda}) = \sum_{s=1}^{n}(\Theta_{su} + \Delta_{su}) \tag{15}$$

$$\Psi_u(\boldsymbol{\lambda}) = \sum_{s=1}^{n}\Big(\theta_s\lambda_{su} + \beta\lambda_{su}(\frac{1}{\mu_s - \sum_i \lambda_{si}} + d_{su})\Big) \tag{16}$$

Therefore, the goal of a front-end proxy at u is to find a feasible load balancing strategy $\boldsymbol{\lambda_u}$ such that $\Psi_u(\boldsymbol{\lambda})$ is minimized. The strategy of u depends on the strategies of other front-end proxies as Ψ_u is a function of $\boldsymbol{\lambda}$.

Definition IV.1. *Feasible strategy profile* is a strategy $\boldsymbol{\lambda}$ that satisfies the following

1) Positivity: $\lambda_{su} \geq 0, \quad \forall s, u$;
2) Conservation: $\sum_s \lambda_{su} = L_u \quad \forall u$;
3) Stability: $\sum_u \lambda_{su} < \mu_s \quad \forall s$;

Definition IV.2. *The non-cooperative load balancing game* is a game played among a set of players. Each player has a set of strategies and an associated cost with each strategy. The game in our scenario is a normal form game with continuous objective function and can be described as:

- *Players:* A finite set of players m denoted as U, $U = \{1, 2, \ldots, m\}$
- *Strategy sets:* Strategy sets: $\lambda_u = \lambda_{1u}, \ldots, \lambda_{su}, \ldots, \lambda_{nu}$ where,
  $\lambda_{su} \in [0, L_u]\ \forall u \in \{1, 2, \ldots, m\}, \forall s \in \{1, 2, \ldots, n\}$,
  s.t. $\sum_s \lambda_{su} = L_u \quad \forall u$
- *Cost:* The cost of a player u is represented by Ψ_u. Each player wants to minimize the cost.

Claim. We can get an upper bound (denoted by $U_{Bnd}(\Psi_u)$ on the objective function in Eq. 16.

Proof. In the Appendix.

Remark: It can be shown that our proposed game is equivalent to a game where the objective function is maximized.

Proof. We note that minimizing Ψ_u is equivalent to maximizing $U_{Bnd}(\Psi_u) - \Psi_u$. Thus, our game definition with this objective is in normal form as when the players maximize the objective function: $\Phi = U_{Bnd}(\Psi_u) - \Psi_u$ [31].

In order to obtain the load balancing strategy for the distributed data center, the above game has to be solved. The Nash equilibrium is the most commonly used solution for such games.

Definition IV.3. *Nash equilibrium* of the above mentioned load balancing game is a load balancing strategy $\boldsymbol{\lambda}$ such that for every front-end proxy u

$$\boldsymbol{\lambda_u} \in \arg\min_{\boldsymbol{\lambda_u}} \Psi_u(\boldsymbol{\lambda_1}, .., \boldsymbol{\lambda_u}, .., \boldsymbol{\lambda_m}) \tag{17}$$

A strategy $\boldsymbol{\lambda}$ is a Nash equilibrium if no player can gain by changing its current strategy to another feasible one. In our load balancing game, the Nash equilibrium has the property that no player can decrease the total cost incurred by choosing a different load balancing strategy $\boldsymbol{\lambda_u}$ given the other players' load balancing strategies. The Nash equilibrium exists for

our game because Ψ_u is continuous, convex and increasing. At the Nash equilibrium, the strategy profile is such that every player's load balancing strategy is a best reply given the other players' strategies. This best reply for a player provides a minimum cost for that player's demand given the other players' strategies. Thus, we first determine the best reply strategy $\boldsymbol{\lambda_u}$ for every player u and then we determine $\boldsymbol{\lambda} = (\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, .., \boldsymbol{\lambda_m})$. Let $\mu_s^u = \mu_s - \sum_{k=1,k\neq u}^{m} \lambda_{sk}$ be the available processing rate at a data center s as perceived by front-end proxy u. Hence, the problem of determining the best reply strategy reduces to finding the optimal job distribution for a system with one front-end proxy u ($\forall u \in U$), n distributed data centers with rates μ_s^u,($\forall s \in S$). We can now express the above problem in the following optimization framework, *Best-reply*$_u$, to seek the solution *Best-reply*.

$$\min_{\boldsymbol{\lambda_u}} \Psi_u(\boldsymbol{\lambda}) \tag{18}$$

subject to the following constraints.

$$\lambda_{su} \geq 0, \qquad \forall s \in S \tag{19}$$

$$\sum_{s=1}^{n} \lambda_{su} = L_u \tag{20}$$

$$\sum_{u=1}^{m} \lambda_{su} < \mu_s, \qquad \forall s \in S \tag{21}$$

The decision variable involved in this optimization is $\boldsymbol{\lambda_u} = (\lambda_{1u}, \lambda_{2u}, .., \lambda_{nu})$, as the strategies of other players are assumed to be constants. As our optimization framework has the objective of minimizing the CIC, we sort the data centers in ascending order of the cost factors, specifically in the ascending order of C_s, where C_s is defined as

$$C_s = \theta_s + \beta d_{su} \tag{22}$$

The following theorem gives the best reply strategy of player u i.e. the solution to the *Best-reply*$_u$.

Theorem 1. Assuming that data centers are sorted based on C_s, the solution $\boldsymbol{\lambda_u}$ for *Best-reply*$_u$ is given by

$$\lambda_{su} = \begin{cases} \mu_s^u - \sqrt{\frac{\beta \mu_s^u}{\alpha - C_s}} & \text{if } 1 \leq s < q_u \\ 0 \text{ if} & q_u \leq s \leq n \end{cases} \tag{23}$$

where q_u is the smallest integer satisfying

$$\sum_{i=1}^{q_u} \sqrt{\frac{\beta \mu_s^u}{\alpha - C_i}} \leq \sum_{i=1}^{q_u} \mu_i^u - L_u \tag{24}$$

The above constraint ensures that the demand is served from a cheaper data center first and the expensive one is used only when all the cheaper data centers are full. α is a Lagrangian multiplier whose value is given by

$$\alpha = \theta_{q_u} + \beta(\frac{1}{\mu_{q_u}^u} + d_{su}) \tag{25}$$

Proof. In the Appendix.

Based on the above theorem, we formulate the following algorithm for determining the best reply for a proxy u.

Algorithm 1 provides the steps to find *Best-reply*$_u$. In step 1, we sort the data centers based on the cost factor Eq.(22). Our aim is to assign greater loads to cheaper data centers. In steps 2 and 3, we initialize the value of p and t according to Eq. (24). The sum has been taken over all the data centers. In order to determine the smallest index data center that satisfies Eq. (24), while loop in step 4 is used. In particular, we assign load to n^{th} data center as 0, if the $(n-1)^{th}$data center processing rate is capable of satisfying Eq. (24). This loop continues till control condition ($p > t$) is met and the corresponding λ_{nu} is set to 0, and n and p are updated accordingly. Finally in step 5, we assign load to data centers according to Eq.(23).

Algorithm 1: *Best-reply*

Input: Total job arrival rate: L_u
Available processing rate at data centers: $\mu_1^u, \mu_2^u, .., \mu_n^u$
Cost factors of data center: $C_1, C_2, ..C_n$
Output: Load balancing strategy:$\boldsymbol{\lambda_u} = (\lambda_{1u}, \lambda_{2u}, .., \lambda_{nu})$

1 Sort the data centers in the ascending order of cost factors i.e. ($C_1 \leq C_2 \leq .. \leq C_n$)
2 $p \leftarrow \sum_{i=1}^{n} \mu_i^u - L_u$
3 $t \leftarrow \sum_{i=1}^{n} \sqrt{\frac{\beta \mu_i^u}{\alpha - C_i}}$
4 **while** $p > t$ **do**
  $\lambda_{nu} \leftarrow 0$
  $p \leftarrow p - \mu_n^u$
  $n \leftarrow n - 1$
  $t \leftarrow \sum_{i=1}^{n} \sqrt{\frac{\beta \mu_i^u}{\alpha - C_i}}$
5 **for** $i = 1, 2, .., n$ **do**
  $\lambda_{iu} \leftarrow \mu_s^u - \sqrt{\frac{\beta \mu_s^u}{\alpha - C_s}}$

Theorem 2. The load balancing strategy $\boldsymbol{\lambda} = (\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, ..., \boldsymbol{\lambda_m})$, given by the *Best-reply* algorithm is the best strategy for front-end proxy u and it solves the *Best-reply*$_u$ problem.

Proof. In the Appendix.

Remarks: (i) The execution time of this algorithm is $\mathcal{O}(nlgn)$. This is due to the sorting procedure in step 1. (ii) In order to execute this algorithm each front-end proxy needs to know all the necessary information such as electricity price, expected delay and available processing capacity at each data center location and estimated demand [32]. In many practical cases, all the proxies and data centers are owned by the same operator. It is possible that all data centers and proxy servers exchange the estimated load, capacity and electricity price in real-time; i.e. whenever electricity price changes or demand changes (due to diurnal pattern).

V. A Distributed load balancing algorithm

Based on the *Best-reply* algorithm discussed in the previous section, we design a greedy algorithm for computing the Nash equilibrium of the non-cooperative game. In order to compute the Nash equilibrium, we propose an algorithm given in Algorithm 2. In addition to the notation mentioned in Sections III and IV, we define the following

l — the iteration index
u — front-end proxy index
$\boldsymbol{\lambda_u^l}$ — the strategy of front-end proxy u at iteration l
Ψ_u^l — CIC of client region u at iteration l
ε — properly chosen acceptance tolerance
$norm$ — $\sum_{u=1}^{m} |\Psi_u^{(l-1)} - \Psi_u^l|$
Send$(u, (p, q, r))$ — send message (p, q, r) to front-end proxy u, where p is the current sum of $|\Psi_u^{(l-1)} - \Psi_u^l|$, q is the iteration number and r is $CONTINUE/STOP$ tag
Recv$(u, (p, q, r))$ — receive message (p, q, r) from front-end proxy u and p, q, r are same as in Send function

Algorithm 2: NASH distributed load balancing algorithm

Input: Total Arrival rate: L_u
Available processing rate at data centers: $\mu_1^u, \mu_2^u, .., \mu_n^u$
Cost factors of data center: $C_1, C_2, ..C_n$
Output: Load balancing strategy at equilibrium:$\boldsymbol{\lambda} = (\lambda_u, \forall u)$
Front-end proxy u, $u = 1, 2, .., m$ executes:

1 Initialization:
  $\boldsymbol{\lambda_u^l} \leftarrow \mathbf{0}$
  $\Psi_u^0 \leftarrow 0$
  $l \leftarrow 0$
  $norm \leftarrow 1$
  $sum \leftarrow 0$
  $tag \leftarrow CONTINUE$
  $left = [(u-2) \bmod m] + 1$
  $right = [u \bmod m] + 1$
2 **while** (1) **do**
    if $(u = 1)$ **then**
      if $l \neq 0$ **then**
        Recv$(left, (norm, l, tag))$
        if $norm < \varepsilon$ **then**
          Send$(right, (norm, l, STOP))$
          exit
      $sum \leftarrow 0$
      $l \leftarrow l + 1$
    else
      Recv$(left, (sum, l, tag))$
      if $tag = STOP$ **then**
        if $u \neq m$ **then**
          Send$(right, (sum, l, STOP))$
        exit
      for $s = 1, 2, .., n$ **do**
        $\mu_s^u \leftarrow \mu_s - \sum_{k=1, k\neq u}^{m} \lambda_{sk}$
        $\boldsymbol{\lambda_u} \leftarrow$ *Best-reply* $(\mu_1^u, .., \mu_n^u, C_1, .., C_n, L_u)$
        Compute Ψ_u
        $sum \leftarrow sum + |\Psi_u^{(l-1)} - \Psi_u^l|$
        Send$(right, (sum, l, CONTINUE))$
      end
end

In this algorithm, each front-end proxy computes its *Best-reply* strategy for every time slot using the current load balancing strategies of other front-end proxies and updates its strategy. In an iteration l of while loop, each front-end proxy server computes its *Best-reply* strategy. It then adds to the sum, the difference in its achieved operating cost compared to previous $(l-1)$ iteration. Then, it sends the sum to its neighbour in a round robin fashion. This continues for several iterations and finally stops when the difference in the total operating cost across all the front-end proxy servers, in successive iterations, is less than the *norm* stopping criterion. We assume that the front-end proxies synchronously update their *Best-reply* strategies in a round robin manner.

The execution of this algorithm is restarted periodically when the data center system parameters (e.g. electricity price, demand) change or a failure occurs. Once the equilibrium is reached, the front-end proxies continue to use the same strategy and the system remains in equilibrium until a new execution is initiated.

Discussion: We examine the message complexity of Algorithm 2. We assume that all the front-end proxy servers (collocated client regions) are logically connected in a ring topology. In every iteration, each front-end proxy server gathers the available capacity at each data center, which requires $\mathcal{O}(n)$ messages. Each front-end proxy server also needs to share its updated sum with the adjacent node, which requires $\mathcal{O}(1)$ messages. Thus, in each iteration all the front-end proxy servers require $\mathcal{O}(mn + m)$ messages, where n and m are the number of data centers and front-end proxy servers, respectively. Since the number of iterations is constant as depicted in Fig. 8, we conclude that the asymptotic message complexity is $\mathcal{O}(mn)$.

VI. Numerical Results

In this section, we compare and analyze the performance of the proposed algorithm with different existing strategies.The proposed algorithm is labeled as NCG. In addition, we also implement for comparison two other models in literature: Proportional scheme (PS) and Global optimal scheme (GOS).

- Proportional scheme(PS):It is distributed and decentralized approach, where each front-end proxy allocates loads in proportion to available processing rates at each data center. It can be noted that proportional scheme does not take into account either the operating cost or the communication delay between front-end proxy and data center.
- Global optimal Scheme(GOS):This scheme minimizes Cost Incurred (CIC) across all Client regions presented in Section IV. The load fractions (λ) are obtained by solving the following non-linear optimization problem:

$$\min_{\boldsymbol{\lambda}} \sum_{u} \Psi_u(\boldsymbol{\lambda}) \tag{26}$$

subject to the following constraints.

$$\lambda_{su} \geq 0, \qquad \forall s \in S, \forall u \in U \tag{27}$$

$$\sum_{s=1}^{n} \lambda_{su} = L_u \qquad \forall u \in U \tag{28}$$

$$\sum_{u=1}^{m} \lambda_{su} < \mu_s, \qquad \forall s \in S \tag{29}$$

GOS is evaluated (at a centralized location) using the optimization tool *fmincon* of Matlab. The parameters used with fmnicon are presented in Table III. For other parameters such as choice of optimization algorithm, optimality tolerance, step tolerance, checkgradient etc. we consider the default value as in [33]. GOS provides the system optimal solution. However, it does not provide fairness to users.

The main performance metrics used in our numerical results are normalized cost, expected response time and the fairness index. The fairness index $F(\boldsymbol{x})$ is calculated as

$$F(\boldsymbol{x}) = \frac{[\sum_{u=1}^{m} x_u]^2}{m \sum_{u=1}^{m} x_u^2} \tag{30}$$

where x_u is the expected latency at client region u. The sum of expected cost across for all client regions C is calculated as

$$C(\boldsymbol{\lambda}) = \sum_{u=1}^{m} \Psi_u(\boldsymbol{\lambda}) \tag{31}$$

where λ is the strategy at the equilibrium. Moreover, the Normalized cost is obtained by normalization with respect to the maximum cost across all approaches.

A. Experimental Setup

Data center locations: We considered data center locations in the USA based on power availability as mentioned in [34]. The locations are: Arizona, Illinois, Iowa, Mississippi, New Hampshire, Oklahoma, Oregon, Pennsylvania, South Carolina, Utah. The service rate of server is 60 requests per second. Relative processing rate available across all data center is presented in Table I, where relative processing the rate of a data center is calculated with respect to the fastest data center processing rate. The average electricity price across all the data center locations is shown in Table I [35].

Client locations: We considered 12 states of the USA, where a large number of Internet users are located [36], i.e. California, Florida, Georgia, Illinois, Michigan, New Jersey, New York, North Carolina, Ohio, Pennsylvania, Texas and Virginia.

Demand from different client locations is proportional to the number of Internet users from that region [36].

Demand: We used trace of traffic from Wiki dump [37] to build the workload profile. Considering hourly wikipedia workload as an aggregate demand for every hour, we distribute demand across different client locations proportional to the number of Internet users at each location. The peak demand from the trace is of the order of thousand requests per second, whereas literature suggests that data centers serve requests to the tune of million requests per second [8]. Therefore we have upscaled the demand by a factor of 3000. The relative job arrival rate for each client location is shown in Table II. The propagation delay between data center and client location is considered to be linearly proportional to the distance between them, and it increases by 10 ms for every 1000 km [38]. The energy consumption of an idle server at any given time could be 30%of the peak power [28], [39], P_{idle} and P_{peak} are taken to be 300W and 100W, respectively [40].

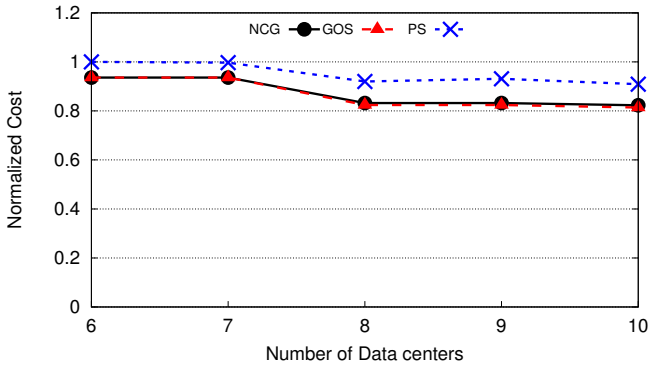

Fig. 2: Impact of the number of data centers on the cost

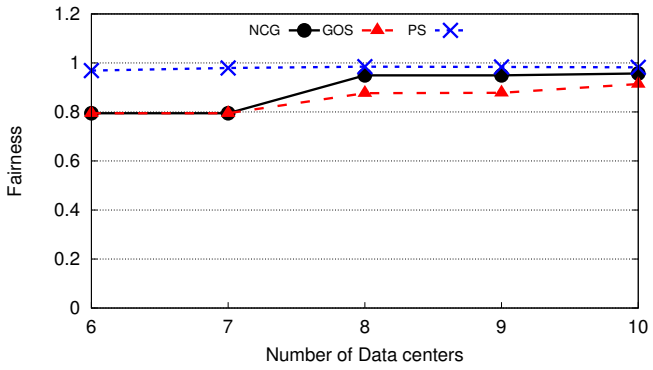

Fig. 3: Impact of the number of data centers on the latency fairness

B. Results

We present and discuss the result obtained from NCG, PS and GOS approaches by varying the number of data centers, system workload and factor term β (in Eq. 12). In each graph, we plotted the normalized value of cost wherein, the normalization is done with respect to maximum cost across all approaches. We run NCG, PS and GOS for a period of one day with the system state changing at every hour. The results presented are average values obtained over this time horizon.

1) Effect of System size:

In this part of numerical evaluation, we vary the number of data centers in the system from 6 to 10 and investigate its effect on cost and fairness. The total demand is set as mentioned in Section VI-A. We set β to 0.1. Fig. 2 shows the normalized cost with respect to the NCG models. It can be observed that NCG model closely approximates the value obtained by GOS. The cost obtained using NCG and GOS is almost the same because these models are aware of electricity price, propagation delay and data center processing rates and therefore perform load balancing in a cost effective manner whereas PS does not obtain optimum cost as it uniformly distributes the load without taking into consideration the electricity prices, propagation delays and data center processing rates. It can be also observed from Fig. 3 that NCG model achieves better fairness in average latency across all client regions when compared to GOS whereas the fairness of PS is approximately 1 across different number of data centers. The NCG approach has an additional advantage of being a

DC Locations	Arizona	Illinois	Iowa	Mississippi	New Hampshire	Oklahoma	Oregon	Pennsylvania	South Carolina	Utah
Electricity Price (in cents/kWh)	5.54	6.34	5.20	6.03	12.33	4.62	6.38	6.78	5.55	5.58
Relative Processing rate	0.4	1	0.2	0.6	0.2	0.6	0.4	1	0.4	1

TABLE I: Average electricity price across data center location

Client Locations	California	Florida	Georgia	Illinois	Michigan	New Jersey	New York	North Carolina	Ohio	Pennsylvania	Texas	Virginia
Relative job arrival rate	0.17	0.11	0.03	0.11	0.04	0.04	0.12	0.03	0.06	0.1	0.13	0.02

TABLE II: Relative job arrival rate of each client location

Parameter	Value
MaxFunctionEvaluations	10000
MaxIterations	5000
FunctionTolerance	10^{-6}
ConstraintTolerance	10^{-6}
OptimalityTolerance	10^{-6}
StepTolerance	10^{-10}
Other parameters	Default values

TABLE III: fmincon parameters

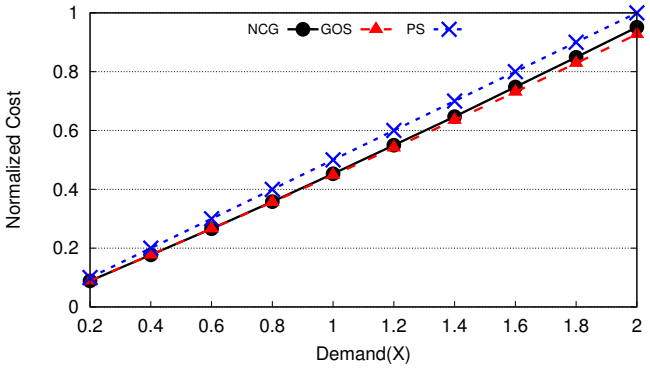

Fig. 4: Impact of demand on the cost

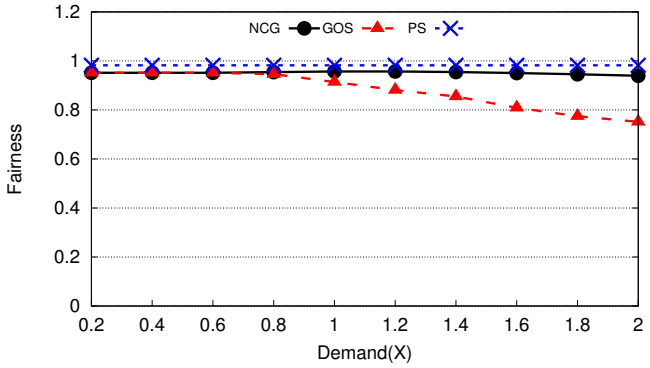

Fig. 5: Impact of demand on the latency fairness

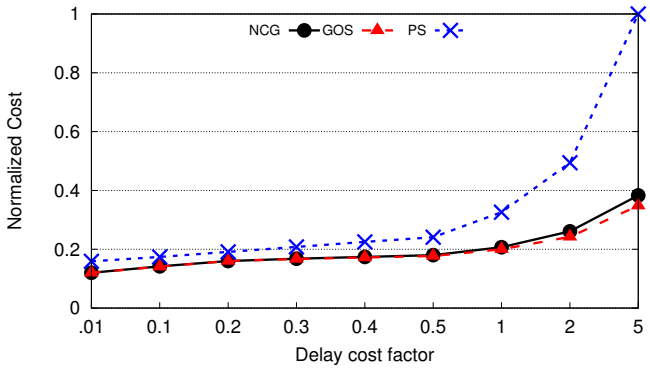

Fig. 6: Impact of cost incurred due to delay on the overall cost

decentralized load balancing scheme.

C. Impact of Demand

In order to understand the impact of demand on cost, we run the proposed algorithm and existing strategies on 10 data centers, 12 client regions. We vary the client demand from 0.2 to 2 times the original client demand obtained from [37]. It may be noted that demand is chosen such that it covers a broad range of data center utilization, where utilization is defined as the ratio of total arrival rate to aggregate processing rate of the system. Demand of 1 corresponds to a utilization of 44%. We set β to 0.1. We depict the results obtained in Fig.4. It can be seen that with increase in demand NCG yields almost the same cost as the GOS approach, which means that NCG approach is as effective GOS. From Fig. 5 it can be seen that NCG and PS maintain a fairness close to 1. Therefore the NCG scheme, apart from being decentralized, it has the additional advantage of user optimality with respect to fairness in latency.

D. Impact of β

The optimization objective that we have chosen to model is the cost of distributed data center in terms of energy cost and weighted latency. In order to understand the impact of β (delay cost factor) on cost and fairness index, we evaluate the proposed and existing schemes with 10 data centers and 12 client regions and client demand as in Section VI-A and varying $\beta \in \{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 1, 2, 5\}$. It can be observed from Fig. 6 that across all β values our model approximates the cost of GOS. With increase in β the cost also increases for all models which is quite intuitive as increase in weight of latency yields a higher cost.

E. Client latency comparison

In Fig. 7, we present the expected latency experienced by each client region. The PS scheme even though it guarantees fairness in latency across all client regions but has the disadvantage of needing higher expected latency. It can also be observed in case of GOS scheme there is a huge variation in expected latency across all client regions whereas NCG provides the minimum possible expected latency for each client region (according to the property of Nash equilibrium). It is not the case that NCG is always better than GOS (as it can be seen from the graph and a similar trend is also in [7]. In many cases, it can be observed that GOS is better. We know that GOS gives an overall system optimal solution that minimizes the linear combination of operating and delay.

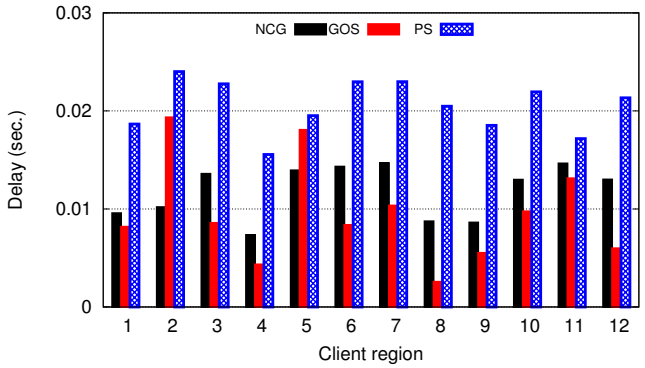

Fig. 7: Average latency perceived across various regions

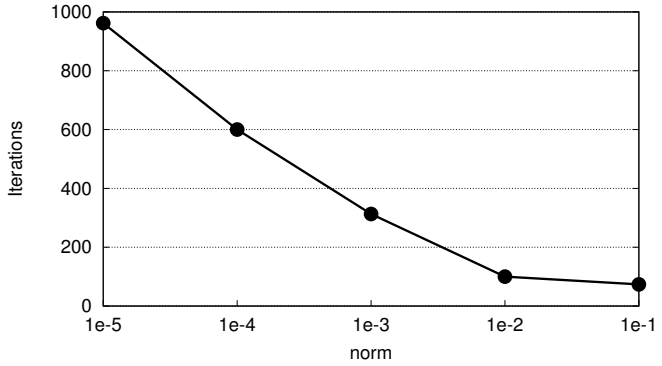

Fig. 8: Impact of norm on convergence

F. Convergence of NCG algorithm

In order to determine the number of iterations required for NCG to converge, we consider two scenarios. First we determine the number of iterations required with varying norm (norm $\in \{1e-5, 1e-4, 1e-3, 1e-2, 1e-1\}$), with 10 data centers and 12 clients regions. We set the demand as mentioned in Section VI-A and β as 0.1. Fig 8 shows the number of iterations required to converge.

VII. Conclusion

In this paper, we have formulated the load balancing problem in distributed data center as a non-cooperative game among front-end proxy servers (which is collocated with client regions). For the proposed game we characterize the structure of Nash equilibrium. Based on obtained Nash equilibrium structure we derive a distributed load balancing algorithm for computing the Nash equilibrium. We compared the performance of our non-cooperative game (NCG) with the existing proportional sharing and global optimal approach. The main advantage of NCG algorithm is, it is decentralized, low complexity (close to the optimal GOS) and it offers fairness and good average latency across all client regions.

Appendix

In this section, we prove the claim and theorems used in the paper.

A. Proof of Claim

Claim: We can get an upper bound (denoted by $U_{Bnd}(\Psi_u)$) on the objective function defined in Eq. 16.

Proof: From Eq. 16 we have

$$\Psi_u(\boldsymbol{\lambda}) = \sum_{s=1}^{n} \Big(\theta_s \lambda_{su} + \beta \lambda_{su} (\frac{1}{\mu_s - \sum_i \lambda_{si}} + d_{su})\Big) \quad (32)$$

Which we can expand to

$$\Psi_u(\boldsymbol{\lambda}) = \sum_{s=1}^{n} \theta_s \lambda_{su} + \beta \sum_{s=1}^{n} \lambda_{su} (\frac{1}{\mu_s - \sum_i \lambda_{si}}) + \beta \sum_{s=1}^{n} \lambda_{su} d_{su} \quad (33)$$

For the bound of these terms we have:

According to Eq. 1 we have $\sum_s \lambda_{su} = L_u$.
Therefore, in all three terms of Eq. 33, we have

$$\sum_{s=1}^{n} \theta_s \lambda_{su} = L_u \sum_{s=1}^{n} \theta_s \quad (34)$$

$$\beta \sum_{s=1}^{n} \lambda_{su} (\frac{1}{\mu_s - \sum_i \lambda_{si}}) = \beta L_u \sum_{s=1}^{n} (\frac{1}{\mu_s - \sum_i \lambda_{si}}) \quad (35)$$

$$\beta \sum_{s=1}^{n} \lambda_{su} d_{su} = \beta L_u \sum_{s=1}^{n} d_{su} \quad (36)$$

A lower bound of the objective function is zero because all terms are non-negative in the Eq. 16. We can get an upper bound of the objective function by noting the sum of the three terms above is less than $L_u[\sum_{s=1}^{n} \theta_s + \beta + \beta \sum_{s=1}^{n} d_{su}]$. The term L_u,θ_s,d_{su} have an upper bound. Thus we can use them and get an upper bound $U_{Bnd}(\Psi_u)$ for the objective function. Then we have, minimizing Ψ_u is equivalent to maximizing $U_{Bnd}(\Psi_u) - \Psi_u$. Thus, our game definition with this objective function is in normal form with the goal of maximizing the objective function: $\Phi = U_{Bnd}(\Psi_u) - \Psi_u$.

B. Proof of Theorem 1

Feasibility: Of the three constraints of the optimization framework, we observe that stability (Eq.29) is always satisfied by the Nash equilibrium solution because of Eq.(17) and the fact that total compute capacity of data center is greater than the cumulative client demand. Hence, we need to consider, Eq.27 and Eq.28 as the constraints for our optimization framework.

We first prove that $\Psi_u(\boldsymbol{\lambda})$ is a convex function in $\boldsymbol{\lambda_u}$ and that the feasible solution set formed by Eq.28 and Eq.27 is convex.

It can be easily shown from Eq.16 that $\frac{\partial \Psi_u(\boldsymbol{\lambda})}{\partial \lambda_{su}} \geqslant 0$ and $\frac{\partial^2 \Psi_u(\boldsymbol{\lambda})}{\partial (\lambda_{su})^2} \geqslant 0$ for $s = 1, 2, .., n$. Hence the Hessian of $\Psi_u(\boldsymbol{\lambda})$ is positive implying that $\Psi_u^{ft}(\boldsymbol{\lambda})$ is a convex function of $\boldsymbol{\lambda_u}$. All the constraints are linear and hence they define a convex polyhedron.

Hence $Best-reply_u$ is an optimization problem with a goal of minimizing a convex function over a convex feasible region. The first order KKT conditions are necessary and sufficient conditions for optimality.

Let $\alpha \geqslant 0, \kappa_s \geqslant 0, s = 1, 2, .., n$ denote the Lagrangian multipliers. The Lagrangian is given by

$$L(\lambda_{1u}, \lambda_{2u}, .., \lambda_{nu}, \alpha, \kappa_1, \kappa_2, .., \kappa_n) = \sum_{s=1}^{n} \Big(\theta_s \lambda_{su} + \beta \lambda_{su}(\frac{1}{\mu_s^u - \lambda_{su}} + d_{su})\Big) - \alpha\Big(\sum_{s=1}^{n} \lambda_{su} - L_u\Big) - \sum_{s=1}^{n} \kappa_s \lambda_{su} \quad (37)$$

The KKT conditions imply that $\lambda_{su}, s = 1, 2, ..n$ is the optimal solution to *Best-reply*$_u$ if and only if there exists $\alpha \geqslant 0, \kappa_s \geqslant 0, s = 1, 2, ..n$ such that

$$\frac{\partial L}{\partial \lambda_{su}} = 0, \quad (38)$$

$$\frac{\partial L}{\partial \alpha} = 0, \quad (39)$$

$$\kappa_s \lambda_{su} = 0, \ \kappa_s \geqslant 0, \ \lambda_{su} \geqslant 0, \ s = 1, 2, .., n \quad (40)$$

Solving Eq.(38),Eq.(39) and Eq.(40), we get

$$\theta_s + \beta\Big(\frac{\mu_s^u}{(\mu_s^u - \lambda_{su})^2} + d_{su}\Big) - \alpha - \kappa_s = 0 \quad (41)$$

$$\sum_{s=1}^{n} \lambda_{su} = L_u \quad (42)$$

$$\kappa_s \lambda_{su} = 0, \ \kappa_s \geqslant 0, \ \lambda_{su} \geqslant 0, \ s = 1, 2, .., n \quad (43)$$

These are equivalent to

$$\alpha = \theta_s + \beta\Big(\frac{\mu_s^u}{(\mu_s^u - \lambda_{su})^2} + d_{su}\Big), \text{ if } \lambda_{su} > 0, \quad 1 \leqslant s \leqslant n \quad (44)$$

$$\alpha \leqslant \theta_s + \beta\Big(\frac{\mu_s^u}{(\mu_s^u - \lambda_{su})^2} + d_{su}\Big), \text{ if } \lambda_{su} = 0, \quad 1 \leqslant s \leqslant n \quad (45)$$

$$\sum_{s=1}^{n} \lambda_{su} = L_u, \quad \lambda_{su} \geqslant 0, \quad 1 \leqslant s \leqslant n \quad (46)$$

From Eq.(44), we get the value of λ_{su} as

$$\lambda_{su} = \mu_s^u - \sqrt{\frac{\beta \mu_s^u}{\alpha - \theta_s - \beta d_{su}}} \text{ if } \lambda_{su} > 0, \quad 1 \leqslant s \leqslant n \quad (47)$$

Claim.*Since our objective is to minimize the CIC, we sort the data centers based on the cost factor ($C_s = \theta_s + \beta d_{su}$), i.e. $C_1 \leqslant C_2 \leqslant .. \leqslant C_n$. Under the given assumption on ordering of data center, we have the following order on load fraction:$\lambda_{1u} \leqslant \lambda_{2u} \leqslant .., \leqslant \lambda_{nu}$. This implies that there may be a case in which no load has been assigned to costliest data center. This mean that there exist an index q_u, $1 \leqslant q_u \leqslant n$ such that*

$$\lambda_{su} > 0, \ s = 1, .., q_u - 1 \quad (48)$$

$$\lambda_{su} = 0, \ s = q_u, .., n \quad (49)$$

The Lagrangian multiplier α has to be chosen appropriately in order to satisfy the conservation constraint Eq(28). Using Eq.(47), Eq.(28), we get

$$\sum_{i=1}^{q_u - 1} \sqrt{\frac{\beta \mu_s^u}{\alpha - \theta_s - \beta d_{su}}} = \sum_{i=1}^{q_u - 1} \mu_i^u - L_u \quad (50)$$

Using Eq.(22) above equation becomes

$$\sum_{i=1}^{q_u - 1} \sqrt{\frac{\beta \mu_s^u}{\alpha - C_i}} = \sum_{i=1}^{q_u - 1} \mu_i^u - L_u \quad (51)$$

Thus the q_u is the minimum index which satisfies

$$\sum_{i=1}^{q_u} \sqrt{\frac{\beta \mu_s^u}{\alpha - C_i}} \leqslant \sum_{i=1}^{q_u} \mu_i^u - L_u \quad (52)$$

From the above discussions, we know that one or more λ_{su} is positive, due to Eq.(27). Hence at the crossroad q_u, we have $\lambda_{q_u u} = 0$. And setting $\lambda_{q_u u} = 0$ in Eq.(47) we get

$$\alpha = \theta_{q_u} + \beta(\frac{1}{\mu_{q_u}^u} + d_{su}) \quad (53)$$

C. Proof of Theorem 2

The while loop in step 4 finds the minimum index q_u for which

$$\sum_{i=1}^{q_u} \sqrt{\frac{\beta \mu_s^u}{\alpha - C_i}} \leqslant \sum_{i=1}^{q_u} \mu_i^u - L_u \quad (54)$$

In the same loop, λ_{iu} are set to zero for $i = q_u, ..., n$. In step 5, λ_{iu} is set equal to $\mu_s^u - \sqrt{\frac{\beta \mu_s^u}{\alpha - C_s}}$ for $i = 1, ..., q_u - 1$. These are in accordance to Theorem 1. Thus, the allocation $\boldsymbol{\lambda_u} = (\lambda_{1u}, \lambda_{2u}, .., \lambda_{nu})$ computed by *Best-reply* algorithm is the optimal solution of *Best-reply*$_u$.

REFERENCES

[1] C. Kim and H. Kameda, "An algorithm for optimal static load balancing in distributed computer systems," *IEEE Transactions on Computers*, vol. 41, no. 3, pp. 381–384, 1992.

[2] M. Guo and L. Yang, *New Horizons of Parallel and Distributed Computing*. Springer US, 2006.

[3] K. Lu, R. Subrata, and A. Y. Zomaya, "Towards decentralized load balancing in a computational grid environment," in *Proc. of International Conference on Grid and Pervasive Computing*, 2006, pp. 466–477.

[4] B. M. Roger, "Game theory: analysis of conflict," 1991.

[5] M. J. Osborne and A. Rubinstein, *A course in game theory*. MIT press, 1994.

[6] R. Subrata, A. Y. Zomaya, and B. Landfeldt, "Game-theoretic approach for load balancing in computational grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 1, pp. 66–76, 2008.

[7] D. Grosu and A. T. Chronopoulos, "Noncooperative load balancing in distributed systems," *Journal of parallel and distributed computing*, vol. 65, no. 9, pp. 1022–1034, 2005.

[8] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *ACM SIGCOMM computer communication review*, vol. 39, no. 4, 2009, pp. 123–134.

[9] 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.of IEEE INFOCOM*, 2010, pp. 1–9.

[10] Y. Zhang, Y. Wang, and X. Wang, "Greenware: Greening cloud-scale data centers to maximize the use of renewable energy," in *Middleware*. Springer, 2011, pp. 143–164.

[11] Y. Shoham and K. Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[12] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "Donar: decentralized server selection for cloud services," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 231–242, 2010.

[13] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. Andrew, "Greening geographical load balancing," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 2, pp. 657–671, 2015.

[14] W. Saad, Z. Han, M. Debbah, A. Hjorungnes, and T. Basar, "Coalitional game theory for communication networks," *IEEE Signal Processing Magazine*, vol. 26, no. 5, pp. 77–97, 2009.

[15] D. Grosu, A. T. Chronopoulos, and M.-Y. Leung, "Load balancing in distributed systems: An approach using cooperative games," in *Proc. of IEEE IPDPS*, 2001, pp. 10–pp.

[16] S. Penmatsa and A. T. Chronopoulos, "Cooperative load balancing for a network of heterogeneous computers," in *Proc. of 20th IEEE IPDPS*, 2006, pp. 8–pp.

[17] R. Subrata, A. Y. Zomaya, and B. Landfeldt, "A cooperative game framework for QoS guided job allocation schemes in grids," *IEEE Transactions on Computers*, vol. 57, no. 10, pp. 1413–1422, 2008.

[18] H. Xu and B. Li, "A general and practical datacenter selection framework for cloud services," in *Proc. of IEEE CLOUD*. IEEE, 2012, pp. 9–16.

[19] K. Li, C. Liu, K. Li, and A. Zomaya, "A framework of price bidding configurations for resource usage in cloud computing," *IEEE Transactions on Parallel and Distributed Systems, forthcoming(2015)*.

[20] L. Mashayekhy, M. M. Nejad, D. Grosu, and A. V. Vasilakos, "An online mechanism for resource allocation and pricing in clouds," *IEEE Transactions on Computers*, vol. 65, no. 4, pp. 1172–1184, 2016.

[21] S. Penmatsa and A. T. Chronopoulos, "Game-theoretic static load balancing for distributed systems," *Journal of Parallel and Distributed Computing*, vol. 71, no. 4, pp. 537–555, 2011.

[22] S. Song, T. Lv, and X. Chen, "Load balancing for future internet: an approach based on game theory," *Journal of Applied Mathematics*, vol. 2014, 2014.

[23] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, 2007, pp. 13–23.

[24] A.-h. Mohsenian-rad and A. Leon-garcia, "Energy-information transmission tradeoff in green cloud computing," in *In Proc. of IEEE Conference on Global Communications (GLOBECOM)*. Citeseer, 2010.

[25] D. Grunwald, C. B. Morrey III, P. Levis, M. Neufeld, and K. I. Farkas, "Policies for dynamic clock scheduling," in *Proc. of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4*. USENIX Association, 2000, pp. 6–6.

[26] E. Samadiani, Y. Joshi, and F. Mistree, "The thermal design of a next generation data center: a conceptual exposition," *Journal of Electronic Packaging*, vol. 130, no. 4, p. 041104, 2008.

[27] "A revolution in data center efficiency," http://multimedia.3m.com/mws/media/1127920O/2-phase-immersion-coolinga-revolution-in-data-center-efficiency.pdf.

[28] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *ACM Sigplan Notices*, vol. 44, no. 3, 2009, pp. 205–216.

[29] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav, "It's not easy being green," in *Proc. of the ACM SIGCOMM*, 2012, pp. 211–222.

[30] K. Le, R. Bianchini, T. Nguyen, O. Bilgir, and M. Martonosi, "Capping the brown energy consumption of internet services at low cost," in *Proc. of International green Computing Conference*, Aug 2010, pp. 3–14.

[31] S. Tadelis, *Game theory: an introduction*. Princeton University Press, 2013.

[32] A. Gandhi, Y. Chen, D. Gmach, M. Arlitt, and M. Marwah, "Minimizing data center SLA violations and power consumption via hybrid resource provisioning," in *Proc. of IEEE IGCC*, 2011, pp. 1–8.

[33] "fmincon," http://in.mathworks.com/help/optim/ug/choosing-the-algorithm.html.

[34] M. Wardat, M. Al-Ayyoub, Y. Jararweh, and A. Khreishah, "To build or not to build? addressing the expansion strategies of cloud providers," in *Proc. of FiCloud*, Aug 2014, pp. 477–482.

[35] "Electricity price in usa," http://www.eia.gov/.

[36] "Internet world stats," http://www.internetworldstats.com/unitedstates.htm.

[37] "Page view statistics for wikimedia projects," http://dumps.wikimedia.org/other/pagecounts-raw/.

[38] A.-H. Mohsenian-Rad and A. Leon-Garcia, "Energy information transmission tradeoff in green cloud computing," *Carbon*, vol. 100, 2010.

[39] "Server efficiency: Aligning energy use with workloads," http://www.datacenterknowledge.com/archives/2012/06/12/server-efficiency-aligning-energy-use-with-workloads/.

[40] B. Subramaniam and W.-c. Feng, "Enabling efficient power provisioning for enterprise applications," in *Proc. of 14th IEEE/ACM CCGrid*, 2014, pp. 71–80.