Latency and Energy-Aware Load Balancing in Cloud Data Centers: A Bargaining Game Based Approach

Avadh Kishor, Rajdeep Niyogi, Anthony Theodore Chronopoulos, *Senior Member, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—With the rapid surge in cloud services, cloud load balancing has become a paramount research issue. The major part of a cloud computing system's operational costs is attributed to energy consumption. Therefore, to provide better QoS, considering the energy minimization factor in load balancing is essential. This paper addresses the latency and energy-aware load balancing problem in a cloud computing system. Specifically, two fundamental performance criteria–response time and energy–for the load balancing problem are considered. To solve this problem, first, the load balancing problem is formulated as an optimization problem. Then it is modeled as a cooperative game so that the solution of the game, called the Nash bargaining solution (NBS), can simultaneously optimize both criteria. The existence and computation of NBS are analyzed theoretically, and an efficient algorithm, called Latency and Energy aWare load balancIng Scheme (LEWIS), is proposed to compute the NBS. Further, to assess the efficacy of LEWIS, it is compared with three other approaches, i.e., Coop_RT, Coop_EN, and NCG, on problem instances of various settings. The experimental results show that LEWIS not only provides less response time while consuming less energy but also gauntness fairness to the end-users.

Index Terms—Cloud computing, load balancing, bargaining problem, cooperative game, nash bargaining solution, energy minimization

1 Introduction and Motivation

With the explosive growth in Internet services, cloud computing (CC) has become a reliable paradigm for service (cloud) providers to provide on-demand services to the end-users. Data centers (DCs) are a crucial infrastructure for cloud computing, because a DC provides a large number of computing devices to a cloud service model. To meet the ever-increasing demand for computing resources, more and more cloud providers, who own one or more DCs, are participating in the cloud market, resulting in an ever increasing number of DCs. Consequently, energy consumption is increasing tremendously [1]. In 2006, the energy consumed by DCs of U.S. was 61×10^9 kWh, 1.5% electricity consumed in the country, at the cost of \$4.5 $\times10^9$ [2]. In a research report [3], it is reported that DCs in U.S. consumed approximately 91×10^9 kWh in 2013 which is an increment of $\approx 49\%$ from 2006 to 2013. Despite taking several steps in reducing the energy requirement for computing needs, keeping in view the efficiency level of 2010, the predicted energy consumption of DCs is to reach $10{,}300\times10^9$ kWh [4]. Moreover, some researchers have studied the cost of managing DCs and concluded that about 40% of the budget is used for power-related categories [5]. Hence, in order to improve the profit of a cloud provider, it is imperative to reduce energy consumption.

Energy consumption and resource utilization in a cloud environment is directly related to each other [6]. Specifically, idle and under-utilized resources contribute a significant amount to wasted energy. In [7], it is reported that the average resource utilization in most of the DCs is 20-30%; and the energy consumption of idle resources can be as much as 60% of the peak power. These research inferences clearly indicate a viable strategy called load balancing for energy reduction in DCs. Load balancing can be defined as a way of distributing the workload among a set of resources to achieve optimal or near-optimal energy levels while maintaining the QoS level of users. Here, an optimal energy level means that DCs' performance per Watt is maximized. Thus, load balancing is and continues to be a paramount issue in CC systems to provide an optimal solution for both cloud providers and users. Load balancing becomes more challenging when, in addition to energy optimization (i.e., efficient resource utilization), some other factors such as fairness and response time are considered.

- *Avadh Kishor is with the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, Punjab 147004, India. E-mails: akishor@cs.iitr.ac.in, avadh.kishor@thapar.edu.*
- *Rajdeep Niyogi is with the Department of Computer Science and Engineering, Indian Institute of Technology, Roorkee, UK 247667, India. E-mail: rajdeep.niyogi@cs.iitr.ac.in.*
- *Anthony Theodore Chronopoulos is with the Department of Computer Science, The University of Texas at San Antonio, San Antonio, TX 78249 USA, and also with the Department Computer Engineering & Informatics, University of Patras, 26500 Rio, Greece. E-mail: antony.tc@gmail.com.*
- *Albert Y. Zomaya is with the School of Information Technologies, University of Sydney, Sydney, NSW 2006, Australia. E-mail: albert.zomaya@sydney.edu.au.*

Manuscript received 21 Oct. 2020; revised 29 July 2021; accepted 17 Oct. 2021. Date of publication 20 Oct. 2021; date of current version 8 Mar. 2023.
The second author was in part supported by a research Grant from Google.
(Corresponding author: Avadh Kishor.)
Recommended for acceptance by P. Spirakis.
Digital Object Identifier no. 10.1109/TCC.2021.3121481

1.1 Contribution

In this paper, a load balancing problem in a CC system where a set of independent users share a set of DCs (owned by a set of cloud providers). There is a set of front-end dispatchers (FEDs) to map users' requests in a distributed manner; each FED acts as a service broker between DCs and users. It is also assumed that DCs are heterogeneous in terms of their processing capacity. Solving this problem is developing a load balancing scheme to achieve cloud providers' and users' goals. In this paper, the following assumptions are made about the goals of cloud providers and users.

- *From cloud providers' perspective:* cloud providers, with their overall system perspective, expect to optimize energy consumption and cost-performance ratio with fairness consideration subject to better (high) resource utilization. Here, the fairness is considered as proportional fairness, which means each provider should get their proportionate share in the system.
- *From the users' perspective:* the users are not directly involved in any decision related to allocating jobs to DCs; instead, they are dependent on FEDs. Thus, the FEDs, in their perspective, expect to minimize their energy cost with fairness consideration subject to minimum response time. Here, fairness is considered in the sense that each FED should receive the same level of utility (in terms of response time and energy cost). Fairness is essential because if a FED provides the same level of service at a higher price (compared to others) vice versa, this risks discouraging its users for future demand.

It is also important to note that, in addition to reducing cost (by optimizing energy consumption), satisfying QoS values, mainly latency and fairness constraints, are significant challenges. These three criteria: energy, response time, and fairness, often present contradicting goals in data center workloads allocation. The reason behind this contradiction is explained as follows. When multiple users share a set of DCs (owned by different cloud providers), executing jobs of a few users first and deferring other users' jobs may reduce the energy cost and increase the response time for other users and adversely affect fairness among users. Therefore, cloud providers look for ways to reduce their operational costs (minimize energy consumption) and improve their profits by satisfying these criteria. In the past, some efforts (a detailed description can be found in the next section) have been made to solve the energy-aware load balancing problem. These works have either considered cloud providers' perspectives or users' perspectives.

This paper addresses a load balancing problem to provide a performance guarantee for users and cloud providers. To do this, first, the problem is formulated as an optimization problem. Then it is cast into a cooperative game among multiple cloud providers. To find the solution of the game, i.e., Nash bargaining solution (NBS), first, the existence of the NBS is established, and then an efficient algorithm, called Latency and Energy aWare load balancIng Scheme (LEWIS), is proposed. In summary, the main contributions of this paper are:

- The above-discussed problem is modeled as a cooperative game called cooperative load balancing game.
- The existence and computation of the NBS are analyzed theoretically.
- To compute the NBS, an efficient algorithm called LEWIS is proposed.
- The fairness of LEWIS, for cloud providers and users, is established theoretically as well as experimentally.
- To establish the efficacy of LEWIS rigorous experimental study (including comparison with three other approaches) is carried out.

1.2 Organization

The rest of the paper is organized as follows. In Section 2, a review of the existing works is presented. The problem description and modeling are given in Section 3. The game formulation, NBS characterization, and LEWIS is presented in Section 4. Section 5 is devoted to the experimental results and its discussions. Conclusions are drawn in Section 6.

2 Related Work

Load balancing is an important and relevant research problem in parallel and distributed computing. In the last decade, this problem has been studied extensively in cloud computing to minimize response time (latency), maintain proper utilization of resources, and reduce energy consumption. We refer readers to [8] for a survey of general load balancing in distributed systems, and [9], [10] for a detailed exposition of energy-aware load balancing in the cloud. Recently, several game-theoretic approaches have been proposed to solve the load balancing problem. Given that we adopt a cooperative game-theoretic approach to solve the problem, this section reviews related work and clarifies our contributions, dominantly in the context of game-theoretic settings. For ease of exposition, we classify previous works in two main categories: (i) game-theoretic approaches and (ii) classical approaches (which do not use game theory).

2.1 Game-Theoretic Approaches

In the cloud domain, game theory provides a framework to study and model the conflicts and interactions (cooperative or noncooperative) among decision-makers (cloud providers and end-users). There exists a considerable amount of literature that considers a game-theoretic solution for the load balancing problem. For ease of exposition, game-theoretic approaches are further divided into two following subcategories.

In the first sub-category, we discuss those load balancing approaches in which response time, utilization imbalance, operating cost, and fairness are considered the problem's objectives. In this direction, different (non)cooperative game theory-based approaches [11], [12], [13], [14], [15], [16], [17], [18] are suggested to deal with the load balancing problem. Grosu *et al.* [11], [12] considered the load balancing problem in a general distributed system where a set of users are sharing a set of resources. The payoff for the players is response time. The authors presented early results on the application of (noncooperative/cooperative) game theory to the load balancing problem for job allocation in distributed systems. The game solution was proved to be a Nash equilibrium solution for the noncooperative game and a Nash bargaining solution for the cooperative game. The authors of [13] and [14] modeled the problem of load balancing in a grid computing environment consisting of n brokers and m providers. They proposed a binary search-based algorithm to compute the NBS of the game. Although these schemes [13], [14] have shown better performance for cloud providers, they still gave an insufficient results on the users' performance. The following works have proposed game theoretic solution for the load balancing in cloud computing distributed system models. For example in [15], the authors have proposed a hierarchal algorithm to compute the NBS of the game. However, the algorithm proposed in [15] provides better performance for users but disregards the cloud providers' performance. Moreover, the approaches considered in [13], [14], [15] have not looked into the problem's energy aspect. In [16], [17], [18], the load balancing problem is modeled as a noncooperative game between users. In [16] only one objective, i.e., minimizing the response time of the users, is considered. An iterative proximal algorithm (IPA) is proposed to compute the optimal response time. In [17] and [18], load balancing problem is modeled as bi-objective optimization problem. In particular, two objectives: response time and execution cost minimization are considered in [17], whereas minimization of response time and utilization imbalance are considered in [18].

In the second sub-category of game-theoretic approaches, we discuss those works that considered energy reduction one of their objectives. In this line of research, several works [19], [20], [21], [22], [23], [24], [25], [26] have model load balancing as a (non)cooperative game with their primary objective of reducing energy consumption. Khan *et al.* [19] addressed the energy-aware task allocation problem in computational grids equipped with dynamic voltage scaling (DVS) feature. In this work, a set of tasks is mapped onto a set of machines. The mapping problem is modeled as a min-max optimization problem. The authors of [21], [22], [23], [24], [25] have formulated the load balancing problem in the cloud environment non-cooperative game. In [21], to efficiently utilize the resource, bidding strategies for the users are designed. To choose the optimal bids (that helps users to decide whether to use cloud service) for a user, an IPA is suggested. In [22], the energy and latency aware problem is considered, where a linear combination of energy cost and latency is considered as the utility of the users. To simultaneously optimize both objectives, an iterative algorithm is proposed. In [23], [24], [25], [26], a Stackelberg game model is adopted to solve the load balancing problem. In these works, the following factors are common. They all consider that a set of users are sharing the resources of a single cloud provider. Despite taking the different energy consumption models, they all employ a powering-off policy to minimize energy consumption. They all adopt a similar leader (i.e., provider of resources) follower (i.e., users) structure of the Stackelberg game. In addition to these similarities, this works [23], [24], [25], [26] have proposed different types of algorithms to solve the problem. Although approaches in [23], [24], [25], [26] exhibit significant energy consumption improvement, they disregard: a) cost-performance ratio of the overall system and b) fairness guarantee between users and providers.

Recently, in [27] and [28], two different game-theoretic approaches are suggested to solve the energy consumption problem in the cloud. Specifically, [27] focuses on the thermal imbalance between the DCs. Here, thermal imbalance means the heat produced by different DCs. To achieve the thermal uniformity between DCs, in [27], authors first modeled the thermal profile of each node, and then considering the thermal profile as the utility function, a cooperative game is designed. Finally, in [27], it is showed that the NBS of the proposed game, i.e., strategies of each node, helps in improving the thermal balance and avoid hotspots. In [28], resource management issue between three parties, i.e., service providers (IaaS), application providers (SaaS), network providers (NaaS) is considered. The main objective in [28] is to reduce infrastructure energy consumption and cost. To solve this problem, in [28], a noncooperative game-based technique is proposed where all three parties are considered as the players. It is shown experimentally in [28] that the solution of the game (i.e., the allocation strategy) reduces energy consumption and also cost-effective.

2.2 Classical Approaches

We categorize classical approaches further into the two sub-categories: (i) software-based approaches such as virtual machine (VM) migration and server consolidation, and (ii) hardware-based approaches, i.e., dynamic voltage frequency scaling (DVFS).

In the first sub-category, a wide range of approaches [6], [29], [30], [31] have been proposed to address the energy consumption issue in cloud environment. Lee and Zomaya [6] adopted the concept that energy consumption is related to server utilization, and to minimize the energy consumption, two heuristics for task consolidation is proposed. The main result of [6] is that by combining more than one task instead of assigning it individually, energy consumption can be minimized. Similarly, Mazumdar and Pranzo [29] considered the power management and efficient resource utilization problem in DCs. Using mathematical modeling, the authors proposed a multi-objective optimization problem to exploit efficient server consolidation. To solve the multi-objective problem, in [29] a heuristic is suggested. To reduce the power consumption in DCs without violating the service level agreement (SLA), a technique to dynamically predict the workload on nodes is suggested in [30]. Specifically, in [30], the Gray-Markov model is adopted to predict the current and future utilization of nodes. It is established in [30] that the proposed consolidation approach is able to reduce energy consumption and VM migration. Mishra *et al.* [31] proposed a VM consolidation-based technique to reduce energy, makespan, and task rejection rate. In [31], an adaptive approach is employed to map VMs to physical machines. Their mapping decision is made by considering energy conservation as a primary aim.

In the second sub-category of classical approaches, we discuss DVFS-based energy-efficient design techniques, which has been widely [32], [33], [34], [35], [36], [37] used in energy-related load balancing approaches. In [32], [33] a

TABLE 1
Comparison Between the Proposed LEWIS With Existing Approaches

Existing Approaches	Comparison parameters								
	1	2	3	4	5	6	7	8	9
Kishor *et al.* [17]	✗	✓	✓	✗	✓	✗	✗	✓	✗
Tripathi *et al.* [22]	✓	✓	✓	✗	✓	✓	✗	✓	✗
Yang *et al.* [23]	✗	✗	✓	✗	✓	✓	✗	✓	✗
Liu *et al.* [15]	✓	✓	✓	✓	✓	✗	✗	✗	✗
Khan *et al.* [19]	✗	✗	✓	✓	✓	✓	✓	✗	✗
Cao *et al.* [32]	✓	✓	✓	✓	✓	✓	✗	✗	✗
Proposed LEWIS	✓	✓	✓	✓	✓	✓	✓	✓	✓

Note- 1: Multiple heterogeneous multi-server system, 2: Geo-distributed DCs, 3: users' perspective, 4: Providers' perspective, 5: Response time, 6: Energy optimization, 7: Minimum performance guarantee to provider, 8: Fairness to users, 9: Proportionally fair allocation for provider; ✓: considered, ✗: Not considered.

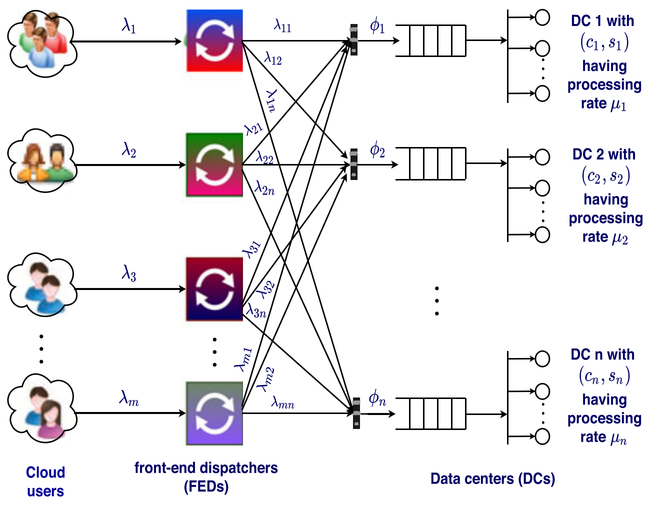


Fig. 1. Cloud computing system architecture.

queuing theory-based optimization method is proposed to manage the energy consumption under DVFS environment effectively. In [34], a DVFS based approach, is suggested to minimize downward and upward energy consumption in heterogeneous servers executing parallel applications. Mezmaz *et al.* [35] considered a multi-objective workflow scheduling problem that includes minimization of makespan and energy consumption. To solve the problem, a multi-objective algorithm by hybridizing energy-conscious scheduling heuristic with the genetic algorithm is proposed in [35]. In [35], the slack time of generated schedule is tuned using DVFS. In [36], a deadline constrained energy-aware algorithm for workflow scheduling is proposed. In approach [36], a VM reuse policy to reuse the idle VM is suggested. To save the energy of leased VMs, a DVFS based slacking algorithm is used. In [37], a cutting-edge cyber-physical system is considered. In particular, a solution to the vehicular fog-computing services is proposed in [37]. The proposed approach enables smart resource management to optimize the communication-plus-computing energy efficiency to achieve the best QoS requirement.

2.3 Qualitative Comparison of the Proposed Approach With Existing Approaches

Having reviewed the related work, we remark that most of the load balancing algorithms (based on the game-theoretic approaches) do not deal with all the three criteria: energy, response time, and fairness, simultaneously. Some of the approaches consider only response time and fairness (up to some extent) but do not look into the energy perspective. However, some approaches dealt with energy consumption, but they either do not consider response time or fairness, or user goal (i.e., only take the resources into account). The comparison of the proposed approach LEWIS with other existing approaches over various parameters is demonstrated in Table 1. Thus, we address a load balancing problem to provide a performance guarantee for users and cloud providers while maintaining the three criteria mentioned above. Though we use the concept of cooperative game theory, similar to the works in [19], [20], our workload model and objectives are different and novel in the following aspects. We adopt the different architectural setup (i.e., three-tier architecture shown in Fig. 1). Our game modeling is done in the wake of satisfying each provider's goal, i.e., minimum energy consumption, improved cost-performance ratio, and proportionally fair allocation of workloads. Our energy model is different. We rely on scheduling decisions to achieve the energy-saving goal. At the same time, in [19], [20] powering off server policy is adopted, which is not reliable in a large-scale cloud market of many cloud providers. We consider the utility (in terms of response and energy cost) of users while ensuring guaranteed fairness.

3 System Model and Problem Formulation

This section describes the CC system model, load allocation model, cloud service model, and energy model followed by the formulation of the problem addressed in this paper.

3.1 Cloud Computing System Model

The CC System architecture considered in this paper, depicted in Fig. 1 consists of three parts, i.e., cloud users (say, users), front-end dispatchers (say, service brokers), and service providers (owner of DCs). This three-tier architecture is prevalent in current CC environment [22], [23], [38]. In this architecture, there is a set of $\mathcal{N} = \{1, 2, \ldots, n\}$ of n DCs, and each DC is owned and operated by by a company called a cloud provider. We assume that each DC j has c_j homogeneous computing units, say servers, with processing rate s_j. Thus, the processing rate of DC j, μ_j, is calculated as $\mu_j = c_j s_j$. Notably, while the servers in each DC are homogeneous, different DCs can have different types of servers.

We denote by $\mathcal{M} = \{1, 2, \cdots, m\}$ a set of front-end dispatchers (FEDs) intended to allocate the users' jobs to the set of n servers. Following the previous works [22], [23], [25], we assume that users associated with a FED are co-located with that FED and a load allocation mechanism which transparent the users. In this mechanism: a) users will assign their jobs to the associated FED who will further allocate jobs to a set of DCs; b) DCs will send the processed result to the FED, and then FED will relay back the result to the users.

Remark 1. Assumptions about the cloud providers and users in the model are as follow:

1) cloud providers, with their overall system perspective, ex- pect to optimize energy consumption and cost-performance ratio with fairness consideration subject to better (high) resource utilization.
2) the users are not directly involved in any decision related to allocating jobs to DCs; instead, they are dependent on FEDs. Thus, the FEDs, in their perspective, expect to minimize their energy cost with fairness consideration subject to minimum response time.

3.2 Load Allocation Model

As mentioned above, users generate jobs to be executed by the DCs; a group of users sends jobs to an associated FED to be allocated for processing. Let each FED i ($i \in \mathcal{M}$) receive jobs of size λ_i (measured by jobs per unit of time) from the users associated to it. Each FED i decides what fraction of λ_i should be allocated to each DC j ($j \in \mathcal{N}$). Let λ_{ij} be the fraction of load (jobs) from FED i allocated to DC j. Thus, each FED i can decide any value of λ_{ij}, as long as $0 \le \lambda_{ij} \le \lambda_i$ and $\sum_{j=1}^{n} \lambda_{ij} = \lambda_i$. The strategy of FED i is the vector $\boldsymbol{\lambda_i} = (\lambda_{i1}, \dots, \lambda_{in})$. strategy of all the FEDs is jointly defined as the vector $\boldsymbol{\lambda} = (\boldsymbol{\lambda_i}, \dots, \boldsymbol{\lambda_m})$.

3.3 Cloud Service Model

The aggregate jobs arrival rate at DC j is $\phi_j = \sum_{i=1}^{m} \lambda_{ij}$. Each DC is modeled as an M/M/c-FCFS queuing system, which is elaborated as follows. At each DC j, the job arrival rate follows ϕ_j a Poisson process, i.e., inter-arrival times are independent and identically distributed (i.i.d.) exponential random variables with mean $1/\phi_j$. Since it may not be possible to process all the jobs immediately after their arrival, each DC maintains an infinite length queue to hold the jobs. To execute the jobs waiting in the queue, each DC adopts the first-come-first-served (FCFS) policy. The execution times of the jobs on each DC are also i.i.d. exponential.

Let $\pi_j(k)$ be the probability that M/M/c queuing system of DC j contains k jobs (waiting or being processed). The utilization of DC j is defined as: $\rho_j = \phi_j/\mu_j$. Now, given stability condition of queue as $\rho_j < 1$, with reference to [15], [32], we have

$$\pi_j(k) = \begin{cases} \pi_j(0)\frac{(c_j\rho_j)^k}{k!}, & \text{if } k < c_j \\ \pi_j(0)\frac{c_j^{c_j}\rho_j^k}{c_j!}, & \text{if } k \ge c_j \end{cases}, \tag{1}$$

where

$$\pi_j(0) = \left[\sum_{k=0}^{c_j-1} \frac{(c_j\rho_j)^k}{k!} + \frac{(c_j\rho_j)^{c_j}}{c_j!} \times \frac{1}{1-\rho_j}\right]^{-1}.$$

The average number of jobs in DC j is given as:

$$\overline{N}_j = \sum_{k=0}^{\infty} k\pi_j(k) = c_j\rho_j + \frac{\rho_j}{1-\rho_j} \times \Pi_j, \tag{2}$$

where Π_j is the probability that the incoming jobs arriving at DC j go to waiting in the queue. To facilitate the theoretical analysis of the queuing system, we assume without loss of generality that all the servers of a DC are likely to be busy, which means Π_j will be 1 [15]. Now, the average response time (of a job) at DC j is

$$\begin{aligned} \mathcal{D}_j &= \frac{\overline{N}_j}{\phi_j}; \qquad \text{[by Little's law]}, \\ &= \frac{1}{\phi_j}\left(c_j\rho_j + \frac{\rho_j}{1-\rho_j}\right); \text{[by (2) and with } \Pi_j = 1]. \end{aligned} \tag{3}$$

In other words, with $\rho_j = \phi_j/\mu_j$, $\mathcal{D}_j$ can be rewritten as

$$\mathcal{D}_j(\phi_j) = \frac{c_j}{\mu_j} + \frac{1}{\mu_j - \phi_j}. \tag{4}$$

3.4 Energy Model

The energy consumed in a DC of the cloud system is mostly due to the two components: 1) computing component, i.e., servers and 2) non-computing components, i.e., cooling and other auxiliary equipment [39]. Let P_j^{PUE} be the power usage effectiveness of DC j, i.e., the ratio between the total power consumed by DC j to the power used by the servers in DC j. We represent the energy per unit time consumed by k_{th} server of DC j by ξ_{jk}^{idle} when k_{th} server is idle and by $\xi_{jk}^{\max}$ when the utilization of k_{th} server is maximum. Given the total load allocated to DC, ϕ_j, the average energy consumption per unit of time of DC j is given as [39]

$$\begin{aligned} E_j(\phi_j) = &\underbrace{\sum_k \xi_{jk}^{\text{idle}}}_{\mathbf{I}} + \underbrace{\frac{\phi_j}{\mu_j}\sum_k \left(\xi_{jk}^{\max} - \xi_{jk}^{\text{idle}}\right)}_{\mathbf{II}} \\ &+ \underbrace{\sum_k (P_j^{\text{PUE}} - 1)\xi_{jk}^{\max}}_{\mathbf{III}}. \end{aligned} \tag{5}$$

In (5), part **I**, **II**, and **III** denote the energy consumption in DC j by idle servers, active servers, and non-computing equipment. Similar to [22], the basis of our energy model is the energy consumed by only computing component, implies that $P_j^{\text{PUE}} = 1$. Therefore, part **III** can be eliminated from (5) because it evaluates to zero. We have taken into consideration that each server of a DC is likely to be active, because if a server is idle, it can be placed in hibernation state. This consideration implies that part **I** can be eliminated from (5). Moreover, we have also assumed in Section 3.1 that all the servers of DC j are homogeneous. This assumption facilitates that, for simplicity, $\xi_{jk}^{\max}$ and ξ_{jk}^{idle} can be written as $\xi_j^{\max}$ and ξ_j^{idle}, respectively. Now, the energy consumed by DC j consists of c_j active servers to process the ϕ_j amount of load is given as

$$E_j(\phi_j) = \frac{\phi_j}{\mu_j}\left(\xi_j^{\max} - \xi_j^{\text{idle}}\right)c_j. \tag{6}$$

3.5 Problem Statement

Given the cloud service model and energy model, the goal of load balancing problem is to choose the load on each DC j

(i.e., ϕ_j) in order to minimize the average response time $\mathcal{D}_j(\phi_j)$. The minimization of $\mathcal{D}_j(\phi_j)$ is achieved by solving, for each cloud user $j \in \mathcal{N}$

$$\min_{\phi_j} \mathcal{D}_j(\phi_j), \tag{7}$$

$$\text{subject to} \qquad E_j \leq E_j^0, \tag{8}$$

$$\sum_{j=1}^{n} \phi_j = \lambda_{\sum}, \tag{9}$$

$$\phi_j \geq 0, \tag{10}$$

$$\phi_j < \mu_j, \tag{11}$$

where $\lambda_{\sum} = \sum_{i=1}^{m} \lambda_i$. Constraints (8) represents the satiability constraint that provides assurance that the energy consumption of a DC is at most E_j^0. Constraint (9) is the conservation constraint that assure the equality of the total arrival load and total processed load. Constraint (10) corresponds to the positivity constraint; it assures that a cloud provider allows a non-negative amount of load to its DC. Constraint (11) is the stability constraint; it provides that the rate of job arrival on a DC is less than its processing rate.

4 Bargaining Game Based Framework

This section presents the game-theoretic formulation of the load balancing problem, characterization of NBS, and an algorithm called LEWIS to compute the NBS.

4.1 Game Formulation

The above-discussed load balancing problem is a type of bargaining problem [40], which refers to a situation where: 1) individuals (cloud providers) have the possibility of collaborating for a mutually profitable agreement, 2) imposing an agreement on an individual without his approval is not possible, and 3) there is a conflict of interest about which agreement to conclude. From a game-theoretic point of view, such a situation can be modeled using the concept of a cooperative game [41]. The solution of the game, called Nash bargaining solution (NBS), is a point at which each individual agrees to conclude.

In the context of modeling discussed in Section 3, the cooperative load balancing game is defined as follows.

Definition 1 (Cooperative load balancing game). *As described in [15], a cooperative game, namely $\mathcal{G}$, is consists of three components: 1) a set of players, 2) a set of strategies, and 3) a set of payoff functions. In this paper, a cloud provider is regarded as a player, i.e., the set of players is a set $\mathcal{N} = \{1, 2, \ldots, n\}$ of cloud providers. The job allocation rate, ϕ_j, that a cloud provider j is allowing for his own DC j is termed as the strategy of player j. The payoff of each player is defined as $f_j(\phi_j) = -\mathcal{D}_j(\phi_j)$.*

Remark 2. We assume that there is a scheduler (single common point) in the proposed bargaining game-based load balancing which collects the information, then it solves the problem, and it does the job assignment.

To provide the formal description of the above-defined game and characterization of the solution to the game, we, with reference to existing theoretical frameworks [15], [19], [41], define some required terminologies as follows.

Let the vector $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)$ be the strategies of all the players, and the vector $\boldsymbol{\mathcal{F}} = (f_1, \ldots, f_n)$ be the payoff of all the players w.r.t. $\boldsymbol{\phi}$. Let E_j^0 be the maximum energy consumption that player j is allowing for DC j without interacting any other player. According to (6), initial job allocation strategy, ϕ_j^0, of player j, in terms of E_j^0 can be expressed as:

$$\phi_j^0 = E_j^0 \frac{\mu_j}{\left(\xi_j^{\max} - \xi_j^{\text{idle}}\right) c_j}. \tag{12}$$

Notably, the value of ϕ_j^0 will be less than μ_j, i.e., $\phi_j^0 = \mu_j - \epsilon$, where $\epsilon > 0$. Thus, by substituting (12) into (4), the initial payoff of the Cloud provider j, denoted as u_j^0, is given as:

$$u_j^0 = f_j^0(\phi_j^0) = \frac{1}{\mu_j}\left[1 + \frac{\left(\xi_j^{\max} - \xi_j^{\text{idle}}\right) c_j}{\left(\xi_j^{\max} - \xi_j^{\text{idle}}\right) c_j - E_j^0}\right]. \tag{13}$$

Definition 2 (Minimal payoff requirement). *Given initial strategy vector $\boldsymbol{\phi^0} = (\phi_1^0, \ldots, \phi_n^0)$, the minimal payoff requirement (or initial agreement point) of all the players, denoted as the vector $\boldsymbol{u^0} = (u_1^0, \ldots, u_n^0)$, represents a minimum payoff guarantee that each player (must) expect from the system.*

Definition 3 (The set of feasible strategies). *The set of feasible strategies, $\mathbb{Q} \subset \mathbb{R}_+^n$, is defined as the space of strategy vectors, i.e., the n-dimensional space bounded by the constraints (9), (10), and another inequality constraint: $\phi_j \leq \phi_j^0$ [this constraint is modified to satisfy the compactness requirement]. Formally,*

$$\mathbb{Q} = \left\{ \boldsymbol{\phi} \in \mathbb{R}_+^n : 0 \leq \phi_j \leq \phi_j^0, \forall j \in \mathcal{N}; \sum_{j=1}^{n} \phi_j = \lambda_{\sum} \right\} \tag{14}$$

In addition, let $\mathbb{F} \subset \mathbb{R}_+^n$ be the set of payoff vectors corresponding to the $\mathbb{Q}$. Formally,

$$\mathbb{F} = \{\boldsymbol{\mathcal{F}} \in \mathbb{R}_+^n : \boldsymbol{\mathcal{F}} \text{ is defined over } \boldsymbol{\phi} \in \mathbb{Q}\}$$

.

Definition 4 (The set of achievable payoff vectors). *Given $\mathbb{F} \subset \mathbb{R}_+^n$ be a non-empty convex and closed set the set of payoff vectors, the set of achievable payoff vectors w.r.t. $\boldsymbol{\phi^0}$ is a set Γ which ensures that each player gets at least their minimum required payoff. Formally, $\Gamma = \{(\mathbb{F}, \boldsymbol{u^0}) : \boldsymbol{u^0} \in \mathbb{R}_+^n \; suchthat \; \mathbb{F}^0 = \{\mathcal{F} \in \mathbb{F} : f_j \geq u_j^0, \forall j \in \mathcal{N}\}\}$.*

Remark 3. The above formulated game (Definition 1) can be formally defined by the tuple $\mathcal{G} = (\mathbb{Q}, \boldsymbol{\mathcal{F}})$. Further, $\mathcal{G}$ is an instance of an n-player bargaining problem defined by the pair $(\mathbb{F}, \boldsymbol{u^0})$. The aim behind solving the bargaining problem is to determine the strategy vector $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)$ so that each player achieves strictly better payoff compared to his initial (minimum required) payoff.

4.2 Characterization of Nash Bargaining Solution

Before addressing the characterization of NBS of the bargaining problem $(\mathbb{F}, \boldsymbol{u^0})$, we define the notion of Pareto optimality w.r.t the set of feasible strategies $\mathbb{Q}$.

Definition 5 (Pareto optimality). *A strategy vector $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n) \in \mathbb{Q}$ is said to be a Pareto optimal strategy vector to the game$\mathcal{G}$ if and only if there is no $\boldsymbol{\phi'} \in \mathbb{Q}$ that dominates $\boldsymbol{\phi}$. That is, there is no $\boldsymbol{\phi'}$ such that: $\forall k \in \mathcal{N}, f_k(\phi'_k) \geq f_k(\phi_k)$ and $\exists k \in \mathcal{N}, f_k(\phi'_k) > f_k(\phi_k)$.*

A payoff vector $\mathcal{F} \in \mathbb{F}$ is said to be the Pareto optimal payoff vector to the $\mathcal{G}$ if and only if it is defined over Pareto optimal strategy vector $\boldsymbol{\phi}$.

In other words, strategy vector $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)$, is called the Pareto optimal if there does not exist any strategy vector that yields strictly superior payoff to all the players simultaneously.

Definition 6 (Nash bargaining solution). *A mapping $\mathbb{S} : \Gamma \rightarrow \mathbb{R}^n_+$ is said to be a Nash bargaining solution if: a) $\mathbb{S}(\mathbb{F}, \boldsymbol{u^0}) \in \mathbb{u}^0$, b) $\mathbb{S}(\mathbb{F}, \boldsymbol{u^0})$ is Pareto optimal and satisfies three fairness axioms*[1].

Definition 7 (Bargaining point). *A payoff vector $\mathcal{F} = (f_1, \ldots, f_n) \in \mathbb{F}$ is said to be a bargaining point if and only if it is the output of function $\mathbb{S}(\mathbb{F}, \boldsymbol{u^0})$. A strategy vector that derives the bargaining point is called NBS.*

Definition 8 (Proportional fairness [42]). *A strategy vector $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n) \in \mathbb{Q}$ is said to be proportionally fair if and only if the aggregate of proportional changes for any other vector $\boldsymbol{\phi'} = (\phi'_1, \ldots, \phi'_n) \in \mathbb{Q}$ is negative:*

$$\forall j \in \mathcal{N}, \qquad \sum_{j=1}^{n} \frac{\phi'_j - \phi_j}{\phi_j} < 0. \tag{15}$$

According to the proportional fairness criterion, the load is assigned to a DC according to its processing capacity while maintaining over all performance, i.e., a DC with higher processing rate (with high bargaining power) is assigned a higher load compared to the DC with lower processing rate (with low bargaining power).

Theorem 1. *There exists a unique bargaining point for the game $\mathcal{G}$.*

Proof. In supplementary Section 2, available online. □

4.3 Computation of Nash Bargaining Solution

In the game $\mathcal{G}$, each player knows the pair $(\mathbb{F}, \boldsymbol{u^0})$ -> to act in cooperation; each player enters in the $\mathcal{G}$ with the aim to achieve win-win solution. So, achieving the win-win solution in coordination means to achieve better payoff for every player jointly. Moreover, for simplicity, we assume that each player in $\mathcal{G}$ can achieve payoff superior to his minimum required payoff.

In order to solve the $\mathcal{G}$, we adopt the concept of NBS. The following theorems give the detail of NBS computation.

1. The detailed description of fairness axioms is given in the supplementary Section 1, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCC.2021.3121481

Theorem 2. *The NBS of the game $\mathcal{G}$ is computed by solving the following optimization problem (labeled as BP):*

$$\begin{aligned}(\text{BP}) \quad \mathbf{max}\mathcal{H}(\boldsymbol{\phi}) &= \sum_{j=1}^{n} \ln\left(f_j - u_j^0\right) \\ &= \sum_{j=1}^{n} \ln\left(u_j^0 - \left(\frac{c_j}{\mu_j} + \frac{1}{\mu_j - \phi_j}\right)\right),\end{aligned} \tag{16}$$

$$\text{subject to} \qquad \sum_{j=1}^{n} \phi_j = \lambda_{\sum}, \tag{17}$$

$$\phi_j \geq 0; \forall j \in \mathcal{N}, \tag{18}$$

$$\phi_j < \mu_j, \forall j \in \mathcal{N}. \tag{19}$$

Before giving the proof of Theorem 2, we need to discuss following facts, which will be used in the proof of Theorem 2.

Fact 1 For simplicity, we have assumed that each player in $\mathcal{G}$ is able to achieve payoff superior to their minimal required payoff. In other words, there exist a strategy vector $\phi^* = (\phi^*_1, \ldots, \phi^*_n) \subset \mathbb{Q}$ such that $f_j(\phi^*_j) > u_j^0$ for all $j = 1, \ldots, n$.

Fact 2 Since $f_j(\cdot)$ is continuous and strictly concave, this assertion can be readily shown because payoff function f_j $(j \in \mathcal{N})$ is injective on ϕ^*.

Proposition 1. *The function $g = \ln(f_j(\cdot))$ is strictly concave.*

Proof. Since $f_j(\cdot)$ is a concave and injective function, by applying Lemma 2.1, that is presented in [41], it can be shown that $g = \ln(f_j(\cdot))$ is strictly concave. Thus, the results follows. □

Proof of Theorem 2: Now, taking into account *Fact 1* and *Fact 2* above, and Proposition 3 along Theorem 2.2, that is presented in [41], the proof of the Theorem 2 follows.

Theorem 3. *Let $\Lambda_j = \frac{1}{\sqrt{\mu_j}}\left(\sqrt{\Delta_j} - \frac{1}{2\sqrt{\Delta_j}}\right)$, where $\Delta_j = \mu_j\varphi_j - c_j$. Given that the DCs are ordered in non-decreasing order of $\Lambda_j \leq \Lambda_k$ if $j < k$, the solution to the BP (i.e., NBS: strategy of each player in $\mathcal{G}$) is given by*

$$\phi_j = \begin{cases} \mu_j\left(1 - \frac{1}{2\Delta_j}\right) - \sqrt{\frac{\mu_j}{\Delta_j}} \frac{\sum_{j=1}^{\ell} \mu_j\left(1 - \frac{1}{2\Delta_j}\right) - \lambda_{\sum}}{\sum_{j=1}^{\ell} \sqrt{\frac{\mu_j}{\Delta_j}}}, & j \in [1, \ell] \\ 0, & j \in [\ell + 1, n], \end{cases}$$

where the DC index ℓ is the given as :

$$\ell = \sup\left\{k \leq n : \Lambda_j < \frac{\sum_{j=1}^{k} \sqrt{\frac{\mu_j}{\Delta_j}}}{\sum_{j=1}^{k} \mu_j\left(1 - \frac{1}{2\Delta_j}\right) - \lambda_{\sum}}\right\}.$$

Proof. In supplementary Section 3, available online. □

Remark 4. Theorem 3 implies that the obtained solution, i.e., NBS is Pareto optimal.

4.4 Latency and Energy Aware Load Balancing Scheme (LEWIS)

Once we have established that a unique NBS to the $\mathcal{G}$ exists, we are interested in obtaining a suitable algorithm to compute the NBS. In other words, each player (in $\mathcal{G}$) wants to determine a strategy so that his response time (latency) is optimal while maintaining a certain level of energy consumption. Based on Theorem 3, the procedure of computing the NBS, named as LEWIS, is formally presented in Algorithm 1.

Algorithm 1. LEWIS: Latency and Energy-Aware Load Balancing Scheme

```
Input: Processing rate $\mu_j$ for each DC $j \in \mathcal{N}$;
Initial strategy vector: $\boldsymbol{\phi}^0 = (\phi_1^0, \ldots, \phi_n^0)$
Output: NBS : $\boldsymbol{\phi}^* = (\phi_1^*, \ldots, \phi_n^*)$;
1  begin
2    Sort the DCs in non-decreasing order of $\frac{1}{\sqrt{\mu_j}}\Big(\sqrt{\Delta_j} - \frac{1}{2\sqrt{\Delta_j}}\Big) \le \frac{1}{\sqrt{\mu_j}}\Big(\sqrt{\Delta_j} - \frac{1}{2\sqrt{\Delta_j}}\Big)$ , if $j < k$;
3    $\ell \leftarrow n$
4    while $\frac{1}{\sqrt{\mu_j}(\sqrt{\Delta_j} - \frac{1}{2\sqrt{\Delta_j}})} \ge \frac{\sum_{j=1}^{k}\sqrt{\frac{\mu_j}{\Delta_j}}}{\sum_{j=1}^{k}\mu_j\Big(1-\frac{1}{2\Delta_j}\Big)-\lambda_{\sum}}$ do
5      $\phi_\ell^* \leftarrow 0$;
6      $\ell \leftarrow \ell - 1$
7    end
8    $\Theta \leftarrow \frac{\sum_{j=1}^{\ell}\sqrt{\frac{\mu_j}{\Delta_j}}}{\sum_{j=1}^{\ell}\mu_j\Big(1-\frac{1}{2\Delta_j}\Big)-\lambda_{\sum}}$;
9    for $j = 1$ to $\ell$ do
10     $\phi_j^* \leftarrow \mu_j\Big(1-\frac{1}{2\Delta_j}\Big) - \sqrt{\frac{\mu_j}{\Delta_j}} \times \frac{1}{\Theta}$;
11   end
12   return $\boldsymbol{\phi}^* = (\phi_1^*, \ldots, \phi_n^*)$        ▷ NBS
13 end
```

LEWIS works through three main stages. In the first stage (which includes step 2), the DCs are sorted in non-decreasing order following the values of a mathematical function (shown in step 2). In the second stage (step 4-7), a set of cloud providers who can achieve a performance superior to their initial performance is determined. In the third stage (step 9-11), the strategy (i.e., the amount of workload that each cloud provider should allow at NBS) is computed. On the basis of these three stages, the run time complexity of LEWIS is presented in the following lemma.

Lemma 1. *The time complexity of* LEWIS *is* $\mathcal{O}(n\log n)$.

Proof. The time complexity of the three stages of LEWIS is as follows: stage 1 requires $\mathcal{O}(n\log n)$ and stage 2 and stage 3 requires $\mathcal{O}(n)$. Hence, the overall time complexity of LEWIS is $\mathcal{O}(n\log n)$. □

Lemma 2. *(Correctness of* LEWIS*) The strategy vector* $\boldsymbol{\phi}^* = (\phi_1^*, \ldots, \phi_n^*)$ *computed by* LEWIS *is the NBS of* $\mathcal{G}$.

Proof. In steps 4 to 7 of Algorithm 1, a minimum index, ℓ such that $\frac{1}{\sqrt{\mu_j}(\sqrt{\Delta_j} - \frac{1}{2\sqrt{\Delta_j}})} \ge \frac{\sum_{j=1}^{k}\sqrt{\frac{\mu_j}{\Delta_j}}}{\sum_{j=1}^{k}\mu_j\Big(1-\frac{1}{2\Delta_j}\Big)-\lambda_{\sum}}$, is determined. Applying Theorem 3, $\phi_{\ell+1}, \phi_{\ell+2}, \ldots, \phi_{\ell+n}$ should all set to zero in order to maximize $\mathcal{H}(\cdot)$ (in step 6). Next, in steps 9 to 11, according to Theorem 3, ϕ_j are set to

$$\mu_j\big(1-\frac{1}{2\Delta_j}\big) - \sqrt{\frac{\mu_j}{\Delta_j}}\frac{\sum_{j=1}^{\ell}\mu_j\big(1-\frac{1}{2\Delta_j}\big)-\lambda_{\sum}}{\sum_{j=1}^{\ell}\sqrt{\frac{\mu_j}{\Delta_j}}}, j \in [1, \ell].$$

Thus, the strategy $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)$ computed in Algorithm 1 is guaranteed to be the optimal solution of BP. According to the Theorem 3, this is also the NBS of game $\mathcal{G}$. Hence, results follows. □

Remark 5. Availability of a computing node refers to the probability that the node is found functional at a given point of time [43]. Steady-state availability is the probability that a node is running during any period in which the node is required to be operational [44]. Availability consideration in a load balancing strategy is responsible for dealing with maintenance activities and unexpected failures. Modeling the availability of a node requires a different treatment. It is determined by various factors, including the node's maintenance status, the number of spare devices dedicated for the node, and the presence or absence of antivirus software. In our paper, we assume that the DC's have sufficient computing resources that are available for all service requests made by the users.

Remark 6. Liveness is in a distributed system refers to a set of properties of concurrent systems that require a system to make progress even though its concurrently executing components may have to take turns [45]. In this paper, we are dealing with the bargaining problem where: 1) individuals (cloud providers) have the possibility of collaborating for a mutually profitable agreement, 2) imposing an agreement on an individual without her/his approval is not possible, and 3) there is a conflict of interest about which agreement to conclude. In LEWIS each player computes her/his agreement point and takes a decision where she wants to collude or not. Each player has complete knowledge about others. The bargaining process takes place in a deterministic environment, and each player i reaches an agreement ϕ_i^* without becoming deadlocked. Thus, the overall strategy of the system becomes $\boldsymbol{\phi}^* = (\phi_1^*, \ldots, \phi_n^*)$. Hence, liveness is guaranteed.

Moreover, to execute LEWIS, we assume that there is request allocator (which can be any of cloud provider) that collects the total arrival rate $\lambda_{\sum}$ (over a fixed interval of time) and processing rate μ_j of each DC. The request allocator collects the common knowledge about all cloud providers and executes LEWIS to compute the NBS. Besides even if NBS is computed at a certain moment, it can not always be the same later without proper measures because $\lambda_{\sum}$ changes dynamically. Thus, as any change in the system state occurs, LEWIS is executed periodically to respond the state change by computing a new NBS.

Theorem 4. *The NBS of the* $\mathcal{G}$*, computed by* LEWIS *is proportionally fair.*

Proof. In supplementary material Section 4, available online. □

4.5 Strategies for FEDs

In this section, we compute the load allocation strategies for each FED. We compute the job allocation strategy for each FED using following:

$$\lambda_{ij} = \lambda_i \frac{\phi_j}{\lambda_{\sum}}, \tag{20}$$

where λ_{ij} is the amount of job allocated to DC j by FED i. Obviously, this scheme satisfies the feasibility criteria for each FED, i.e., $\lambda_{ij} \geq 0$, $\sum_{j=1}^{n} \lambda_{ij} = \lambda_i$, and $\sum_{i=1}^{m} \lambda_{ij} = \phi_j < \mu_j$.

4.5.1 Response Time to FEDs

Given $\boldsymbol{\lambda} = (\boldsymbol{\lambda_i}, \ldots, \boldsymbol{\lambda_m})$, the average response time of each FED $i \in \mathcal{M}$, with reference to [17], is determined as:

$$\mathcal{R}_i(\boldsymbol{\lambda}) = \frac{1}{\lambda_i} \sum_{j=1}^{n} \lambda_{ij} \mathcal{D}_j = \frac{1}{\lambda_i} \sum_{j=1}^{n} \frac{\lambda_{ij} c_j}{\mu_j} + \frac{\lambda_{ij}}{\mu_j - \phi_j} \tag{21}$$

The completion of a job, $\mathcal{R}_i^T$ of FED i includes the round-trip network delay, d_{ij} (between FED i to a set of DC j), and average response time $\mathcal{R}_i(\boldsymbol{\lambda})$. To model the network delay for FED i, we define $\boldsymbol{d}_j^{\text{net}}$ the total round-trip delay between FED i and a set of DCs, as $\boldsymbol{d}_j^{\text{net}} = \sum_{i=1}^{m} a_{ij} d_{ij}$, where a_{ij} is an indicator variable which determines whether the FED i allocates the fraction of jobs to DC j or not; If it does so, $a_{ij} = 1$, else $a_{ij} = 0$. Thus, we have

$$a_{ij} = \begin{cases} 1, & \text{if } \lambda_{ij} > 0 \\ 0, & \text{if } \lambda_{ij} = 0 \end{cases} \tag{22}$$

Now, the total response time for FED i is given as:

$$\mathcal{R}_i^T(\boldsymbol{\lambda}) = \mathcal{R}_i + \boldsymbol{d}_j^{\text{net}} = \frac{1}{\lambda_i} \sum_{j=1}^{n} \frac{\lambda_{ij} c_j}{\mu_j} + \frac{\lambda_{ij}}{\mu_j - \phi_j} + \sum_{i=1}^{m} a_{ij} d_{ij} \tag{23}$$

4.5.2 Cost to FEDs

Let the vector p_j be the electricity price of DC j, and let E_j (given in (6)) is the amount of electricity consumed by DC j . Then, given $\boldsymbol{\lambda} = (\boldsymbol{\lambda_i}, \ldots, \boldsymbol{\lambda_m})$ the total cost incurred by FED i is given as:

$$\mathcal{J}_i(\boldsymbol{\lambda}) = \sum_{j=1}^{n} \frac{p_j \lambda_{ij} E_j}{\lambda_i} = \sum_{j=1}^{n} \frac{p_j \lambda_{ij} \phi_j}{\lambda_i \mu_j} \left(\xi_j^{\max} - \xi_j^{\text{idle}} \right) c_j. \tag{24}$$

4.5.3 Fairness to FEDs

In order to characterize the fairness of LEWIS w.r.t. to FEDs, we adopt the following fairness criteria [17]:

$$\text{FI} = \left(\sum_{i=1}^{m} \mathcal{U}_i \right)^2 / m \sum_{i=1}^{m} \mathcal{U}_i^2, \tag{25}$$

where, $\mathcal{U}_i$ denotes the utility of FED i. Here, utility of each FED is a pair of response time and cost. FI is a measure of the equality of utilities of different FEDs. The value of FI lies between 0 and 1. Ideally, if utility values (cost and response time) of each FED are equal, then this metric should evaluate to 1. Now, for given allocation strategy (derived from LEWIS), we provide the following result.

Lemma 3. *The fairness indexes, i.e.,* FI-R *and* FI-C, *evaluate to 1, i.e, for FEDs,* LEWIS *guarantees the fairness in terms of cost and time.*

Proof. The proofs for FI-R =1 and FI-C =1 are as follows.

- *Proof for FI-R =1:*Given $\lambda_{ij} = \lambda_i \frac{\phi_j}{\lambda_{\sum}}$, the response time for $\mathcal{R}_i$ for all FEDs can be expressed as

$$\begin{aligned} \mathcal{R}_i(\boldsymbol{\lambda}) &= \sum_{j=1}^{n} \frac{\lambda_{ij}}{\lambda_i} \left(\frac{c_j}{\mu_j} + \frac{1}{\mu_j - \phi_j} \right) \\ &= \sum_{j=1}^{n} \frac{\phi_j}{\lambda_{\sum}} \left(\frac{c_j}{\mu_j} + \frac{1}{\mu_j - \phi_j} \right). \end{aligned} \tag{26}$$

  Thus, all $\mathcal{R}_i$, $\forall i = 1, \ldots, m$ are equal and this implies that FI-R = 1.

- *Proof for FI-C =1:* Given $\lambda_{ij} = \lambda_i \frac{\phi_j}{\lambda_{\sum}}$, the cost $\mathcal{J}_i$ for all FEDs can be expressed as

$$\begin{aligned} \mathcal{J}_i &= \sum_{j=1}^{n} \frac{p_j \lambda_{ij} \phi_j}{\lambda_i \mu_j} \left(\xi_j^{\max} - \xi_j^{\text{idle}} \right) c_j \\ &= \sum_{j=1}^{n} \frac{p_j (\phi_j)^2}{\lambda_{\sum} \mu_j} \left(\xi_j^{\max} - \xi_j^{\text{idle}} \right) c_j. \end{aligned} \tag{27}$$

  Thus, all $\mathcal{J}_i$, $\forall i = 1, \ldots, m$ are equal and this implies that FI-C = 1. □

5 Experiments and Analysis

In this section, we perform various experiments to validate the effectiveness of the proposed scheme and demonstrate our findings (optimal response time, cost, the effect of demand).

5.1 Experimental Setup

Intending to make our experimental analysis close to realistic experiments, we model an Internet-scale system such as Google within the US.

5.1.1 Data Center Description

On the basis of power availability as mentioned in [22], [39], [46], we consider 12 data centers, each located at the geographic center of a state: New Hampshire (NH), Mississippi (MS), Iowa (IA), Utah (UT), Oklahoma (OK), South Carolina (SC), Oregon (OR), Arizona (AZ), Pennsylvania (PA), Illinois (IL). The number of servers c_j in each DC is proportional to the number of Internet users at the location. The processing rate of each server is 3600 jobs/seconds. Since a DC processes millions of jobs per second, the processing rate of each DC, μ_j, is computed as $\mu_j = c_j * 3600$. Moreover, the electricity price of each state is fixed using the industrial price [47]. The detailed description of DCs along with state-wise electricity cost, is given in Table 2.

TABLE 2
DCs' Configurations

DC locations	NH	MS	IA	UT	OK	SC	OR	AZ	PA	IL
Number of servers	28	42	55	58	60	73	78	132	233	241
EP (cent/kWh)	20.23	11.33	11.72	10.09	9.06	12.27	10.69	11.68	13.63	12.52
PR (MJPS)	0.1008	0.1512	0.198	0.2088	0.216	0.2628	0.2808	0.4752	0.8388	0.8676

EP: Electricity price, PR: Processing rate, kWh: kilowatt-hour, MJPS: Million jobs per second.

TABLE 3
FEDs' Workload Configurations

Client locations	CA	FL	GA	IL	MI	NJ	NY	NC	OH	PA	TX	VA
Relative job arrival rate	0.22	0.10	0.05	0.07	0.05	0.05	0.11	0.05	0.06	0.07	0.12	0.04

5.1.2 Users' Workload Description

To build the workload profiles of users, we consider 12 different states of the US, where a large number of Internet users are present [48], viz. California (CA), Florida (FL), Georgia (GA), Illinois (IL), Michigan (MI), New Jersey (NJ), New York (NY), North Carolina (NC), Ohio (OH), Pennsylvania (PA), Texas (TX), Virginia (VA). The job arrival rate (or demand) of each user location is decided according to the number of total users present in that location (proportional to number of users). The complete configuration of workload profiles for each user location is given in Table 3.

5.1.3 Some Other Factors

The round trip network delay between a user location and a DC is assumed to be the proportional distance between them [46]. To make the round trip delay comparable with processing delay, we fix 0.1 milliseconds for every 2000 km. According to [22], we set $\xi^{max} = 300$ W and $\xi^{\text{idle}} = 100$ W.

5.2 Performance Metrics

In order to evaluate the performance of the LEWIS, we define the following four metrics.

1) *Average time* (Avg_{time}): It measures aggregate response time of the whole system [17], and is defined as:

$$\text{Avg}_{time} = (1/\phi_{\sum}) \times \sum_{j=1}^{n} \phi_j \mathcal{D}_j, \text{where } \phi_{\sum} = \sum_{j=1}^{n} \phi_j.$$

2) *Load imbalance* (Load_{imb}): It measures how evenly the DCs are utilized, and is evaluated as:

$$\text{Load}_{imb} = ((\rho^{max}/\widehat{\rho}) - 1) \times 100\%,$$

where $\rho^{max} = max\{\rho_j : j = 1, \cdots, n\}$ denotes the maximum utilization of a data center and $\widehat{\rho}$ denotes the average utilization of the DCs. Ideally, if each data center is equally utilized, then this metric should evaluate to zero;

3) *Average cost (*Avg_{cost}*):* It measures aggregate cost of the system, and is defined as:

$$\text{Avg}_{cost} = (1/\phi_{\sum}) * \sum_{j=1}^{n} p_j * \phi_j * E_j,$$

where E_j and p_j are the energy consumed and electricity price for DC j, respectively;

4) *Energy efficiency* ($\text{Energy}_{\text{eff}}$): It is a cost-performance ratio, i.e., $\text{Energy}_{\text{eff}} = \text{Avg}_{\text{cost}}/\text{Avg}_{\text{time}}$. If the value of Energy_{eff} of an approach is relatively low then, it would imply that approach is relatively high energy efficient.

5.3 Approaches Used in the Comparison

To judge the relative effectiveness of LEWIS, we consider two baselines and one state-of-the-art approach. These two baselines are approximations of conventional approaches developed for load balancing in distributed systems, and they allow us to make implicit comparisons to previous works. Note that each approach considers the propagation delay and the constraints of the original problem.

5.3.1 Baseline 1

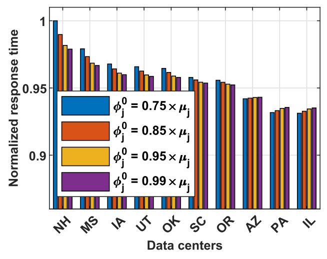
This considers the minimization of job response time and ignores the minimization of energy. This baseline only minimizes the response time; it is referred to as Coop_RT. In Coop_RT , each cloud provider is modeled as a player of the cooperative game, and the NBS is computed by solving the following optimization problem:

$$\mathbf{max} \sum_{j=1}^{n} \ln(\mu_j - \phi_j).$$

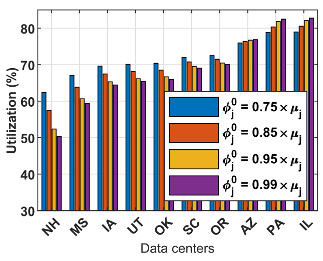
Coop_RT demonstrates the importance of a latency aware load balancing scheme. Furthermore, it allows us to assess the effectiveness of LEWIS against to those such as [13], [14], [15] that use the concept of NBS.

5.3.2 Baseline 2

This baseline, referred to as Coop_{EN}, considers the energy factor while ignoring the latency. In this approach, the problem is modeled as a cooperative game between cloud providers, and the NBS is computed by solving:



(a) Impact of ϕ_j^0 on response time



(b) Impact of ϕ_j^0 on utilization

Fig. 2. The impact of ϕ_j^0 on the performance of LEWIS.

$$\max \sum_{j=1}^{n} \ln(E_j^{max} - c_j(\xi^{max} - \xi^{\text{idle}})\phi_j/\mu_j).$$

5.3.3 State-of-the-art Approach [22]

This approach, called NCG, considers the minimization of energy and time in an integrated manner. The problem is modeled as a cooperative game between the FEDS. We use the same objective functions for players as in the original work [22]. NCG allows us to make a comparison of LEWIS against to those such as [13], [14], [15], [19], [20] that deal with both energy and time.

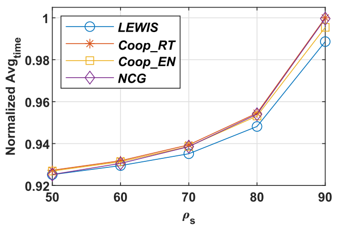
5.4 Sensitivity Analysis

From the formulation shown in Section 4, it is evident that ϕ_j^0 is a vital parameter to derive the NBS, i.e., the performance of LEWIS depends on the value of ϕ_j^0. Therefore, in this set of experiments, the sensitivity of LEWIS to ϕ_j^0 is studied. To carry out the set of experiments, we vary the value of ϕ_j^0 as $0.75 \times \mu_j, 0.85 \times \mu_j, 0.95 \times \mu_j, \text{and } 0.99 \times \mu_j$. we consider the system of 10 DCs and 12 FEDs, whose configuration is given in Tables 2 and 3, respectively. The aggregate job arrival rate $\lambda_{\sum}$ is determined using a total processing rate (i.e., $\sum_{j=1}^{n} \mu_j$) and system utilization (ρ_s) as follows:

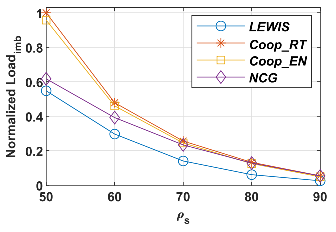
$$\lambda_{\sum} = \rho_s \times \sum_{j=1}^{n} \mu_j \tag{28}$$

Here, we fix $\rho_s = 0.75$. The average processing time ($\mathcal{D}_j$) and the utilization (ρ_j) of each DC versus the change in value ϕ_j^0, as determined by LEWIS, are presented in Fig. 2. In Fig. 2a the response time of all the DCs, for each value of ϕ_j^0, is normalized by dividing maximum response time across all the DCs.

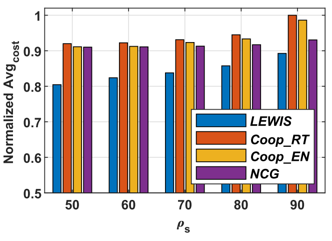
From Fig. 2a, a trend can be observed that, with the increase of value ϕ_j^0, the response time of slower DCs (NH to OR) decreases while for faster DCs (AZ to IL) increases. The utilization of DCs, shown in Fig. 2b, follows a similar trend as in Fig. 2a. This trend could be explained by the fact that: a) the player's bargaining power (i.e., minimum performance requirement) is directly proportional to the value of ϕ_j^0, and b) a higher value of ϕ_j^0 implies higher bargaining power to player j and the better payoff (here, minimum response time). Having observed the patterns shown in Fig. 2, it would be reasonable enough to select $\phi_j^0 = 0.95 \times \mu_j$ as the best suited value for LEWIS given these above mentioned configuration of DCs (Tables 2 and 3)due to the following Reasons:



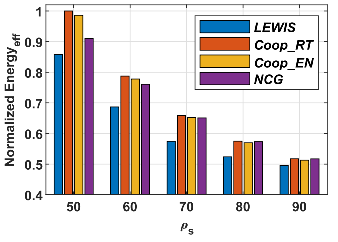
(a) Demand vs average time



(b) Demand vs load imbalance



(c) Demand vs average cost



(d) Demand vs energy efficiency

Fig. 3. Effect of demand on all four approaches.

1) for the majority of the DCs (i.e., 7 out of 10), this value yields the best results (because at $\phi_j^0 = 0.95 \times \mu_j$, the time and utilization for all 7 DCs are better than any other value of ϕ_j^0);
2) the performance loss in the faster DCs can be amortized over the slower ones. For example, the response time and utilization of the fastest one, i.e., IL, increases by $\approx 0.4\%$ and 3.5%, respectively. In contrast, for the slowest one, i.e., NH, these values decrease by $\approx 2\%$ and 12%.

In conclusion, the value of ϕ_j^0 plays an important role in performance of LEWIS, i.e., as the value of ϕ_j^0 increases, the waiting time for the jobs assigned to DC j decreases. Therefore, we shall use the value of $\phi_j^0 = 0.95 \times \mu_j$, for all the experiments in forthcoming sections.

5.5 Performance Comparison

In this section, the relative performance of LEWIS is investigated against three other approaches, namely Coop_RT, Coop_EN, and NCG, using four performance metrics (given in Section 5.2).

5.5.1 Effect of Demand

Demand (or system utilization) is defined as the percentage of the system's aggregate processing rate, requested by the users. In this set of experiments, the value of demand (ρ_s) is varied from 50% to 90% for a system of 10 DCs and 12 FEDs (given in Tables 2 and 3) is considered. The normalized values of different performance metrics obtained by all four approaches are shown in Fig. 3. The normalization of a performance metric value is obtained by dividing the maximum value of that metric across all the approaches. In order to make the discussion of the results (shown in Fig. 3) more concrete, we also present the number of participating cloud providers in NBS for the four approaches over varying demand size in Table 4[2].

2. The actual utilization percentage of each DC for all the approaches, over varying demand size is presented in the Table 1 of supplementary material, available online.

TABLE 4
Number of Participating Cloud Providers in NBS Over Varying Demand (ρ_s)

	Demand (ρ_s)				
Approaches	50%	60%	70%	80%	90%
LEWIS	5	7	9	10	10
Coop_RT	8	9	9	10	10
Coop_EN	8	9	9	10	10
NCG	8	9	10	10	10

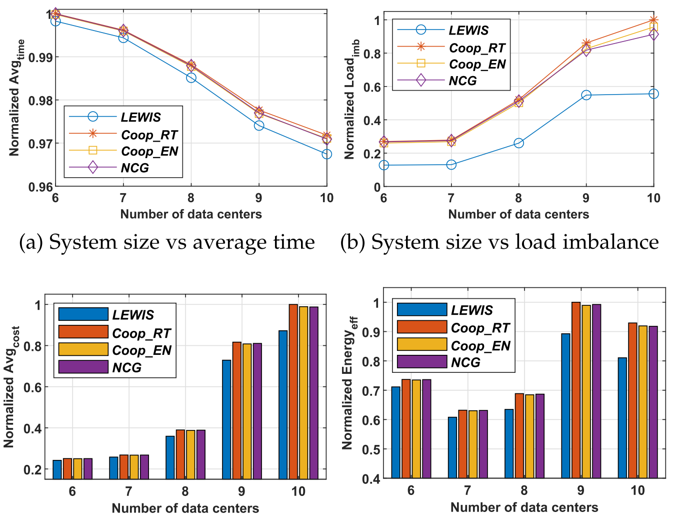
From Fig. 3, one can make the following observations:

1) The value of $\mathsf{Avg}_{\mathsf{time}}$ for all the approaches increases as the demand nears the maximum capacity of the system. This trend can be attributed to the fact that increased demand size implies a higher workload and the long queue length on each DC. When the demand reaches the maximum capacity of the system, the effect of processing time on the $\mathsf{Avg}_{\mathsf{time}}$ becomes less pronounced, and the impact of average waiting time becomes most pronounced. Nevertheless, LEWIS outperforms its counterparts across all types of demand sizes. The reason for the superior performance of LEWIS can be explained from Table 4 and Table 1 (in supplementary material, available online.), where the empirical results demonstrate the following. Unlike others, in case of less demand, LEWIS allows only the fastest DCs to participate in job processing because the slower ones can deteriorate the whole system's performance. In the case of the highest demand, like others, LEWIS allows all DCs to participate, but it utilizes them more fairly than others.
2) In Fig. 3b, it can be seen that LEWIS provides better resource utilization compared with the other approaches, $\mathsf{Load}_{\mathsf{imb}}$ for LEWIS is less than those other approaches. The explanation, given in the latter part of item 1), could be used towards explaining*realizing* this trend in $\mathsf{Load}_{\mathsf{imb}}$ value.
3) In Fig. 3c, it can be seen that the value of $\mathsf{Avg}_{\mathsf{cost}}$ for all the approaches increases with the demand; this is because a higher demand implies a high energy consumption. Nevertheless, LEWIS outperforms others because it utilizes the resources fairly.
4) In Fig. 3d, it can be seen that the value of $\mathsf{Energy}_{\mathsf{eff}}$ for LEWIS subsides (more in comparison to others) with the increase of demand. This trend reveals that LEWIS scales better in presence of high demand.

In conclusion, LEWIS has better scaling behavior than its counterparts for demand, and the LEWIS is the best among its counterparts for all the metrics.

5.5.2 *Effect of System Size*

In Internet-scale distributed systems, increasing or decreasing the system size (here, the number of DCs) affects a load balancing's performance. In order to address this issue of scalability, in this set of experiments, the effectiveness of LEWIS along with its three counterparts, viz., Coop_RT, Coop_EN, and NCG, is investigated over a varying system size. We vary the system size between 6 to 10 (6 DCs: NH to SC, 7 DCs: NH to OR, 8 DCs: NH to AZ, 9 DCs: NH to PA, 10 DCs: NH to IL), while keeping the number of FEDs and value of ρ_s constant, i.e., 12 and 0.75, respectively. The normalized values of different performance metrics obtained by all four approaches are shown in Fig. 4. The normalization process is the same as in Section 5.5.1.



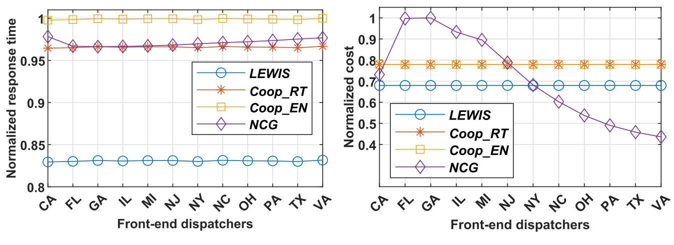
(a) System size vs average time (b) System size vs load imbalance

(c) System size vs average cost (d) System size vs energy efficiency

Fig. 4. Effect of system size on all four approaches.

From Fig. 4, one can make the following observations:

1) In Fig. 4a, Avg_{time} for all the approaches improves (decreases) with the increase of system size; this is because the inclusion of more powerful DCs in the system enhances the processing capacity of the system. The superior performance of LEWIS demonstrates that it is more efficient in utilizing the system's processing capacity compared to its counterparts.
2) In Fig. 3b, $\mathsf{Load}_{\mathsf{imb}}$ increases with the increase of system size; this is because the addition of faster DCs (i.e., AZ, PA, and IL) will increase the heterogeneity of the system. The figure also illustrates the effectiveness of LEWIS with a pronounced difference, specially in the case of larger system size (i.e., number of DCs is 10), with others, i.e., LEWIS lowers the value of Load_{imb} by $\approx 28\%$, $\approx 26\%$, and $\approx 23\%$, in comparison with Coop_RT, Coop_EN, and NCG, in the case of larger system size (i.e., number of DCs is 10).
3) In Fig. 3c, $\mathsf{Avg}_{\mathsf{cost}}$ value for all the approaches increases as the system size increases; this is because the participation of the potent DCs in job processing would increase the energy consumption of the whole system.
4) In Fig. 3d, $\mathsf{Energy}_{\mathsf{eff}}$ varies as system size increases. This variation could be attributed to the change in the electricity price of DCs included in the expansion of system size. For instance, a DC (OR with the price of 10.69 cents/kWh) included expanding the size from 6 to 7, which is cheaper than the DC (AZ with the price of 11.68 cents/kWh) for expanding the size from 7 to 8. Due to this, $\mathsf{Energy}_{\mathsf{eff}}$ at size 7 is less compared to a size 6; on



(a) Expected time of FEDs (b) Expected cost of FEDs

Fig. 5. Performance for different FEDs.

the other hand, $\text{Energy}_{\text{eff}}$ at size 8 is higher than 7. Nevertheless, irrespective of system size, LEWIS is more effective than the others.

In conclusion, irrespective of the system size, LEWIS shows better scalability, in terms of system size, than its counterparts.

5.5.3 *Analysis for Front-end Dispatchers*

In this section, to validate the performance of LEWIS from FEDs' (say users') perspective, it is compared with other approaches in terms of two measures: 1) expected response time of each FED (FI-R) and 2) expected cost of each FED (FI-C). For this experiment, a system of 10 DCs and 12 FEDs is considered, and $\rho_s = 0.75$. The normalized values of the two measures obtained by all four approaches are shown in Fig. 5. The normalization process is the same as in Section 5.5.1.

From Fig. 5, one can make the following observations:

1) Each approach results in the equal expected response time (except for NCG), still LEWIS gives the lowest (best) response time, i.e., it lowers the response time by up to 16% in comparison with Coop_RT and Coop_EN, and by up to 20% in contrast with NCG.
2) LEWIS gives less cost in comparison with Coop_RT and Coop_EN, i.e., it lowers the cost by up to 10% in comparison with both approaches. Even though for some FEDs, NCG gives a lower cost in comparison with LEWIS, still NCG lowers the cost of some FEDs (by up to 35 % in comparison with LEWIS) by increasing the cost of others (by up to 49% in comparison with LEWIS).

Overall, LEWIS not only provides guaranteed fairness but also results in better performance in response time and cost for all the FEDs.

5.6 Summary of the Results

In the wake of the presented results and the discussions above, the salient features of LEWIS are summarized as follows.

1) *Fairness:* In Section 5.5.3, it is shown that LEWIS assures the fairness in terms of response time and cost.
2) *Efficient system-wide performance:* This articulation can be realized with respect to the performance of LEWIS Sections 5.5.1 and 5.5.2, where LEWIS outperforms others over varying system conditions across the performance metrics considered.
3) *Advantageous for Users as well as providers:* Besides the fact that LEWIS guarantees the fairness to the users (irrespective of allocated FED(s), each user will receive equal response time and cost), the results presented in Figs. 5a and 5b affirms that LEWIS is a better suited approach for all the users. Since, LEWIS utilizes the DCs in a balanced manner, each provider will experience a fair cost and optimal energy consumption.

In summary, having realized the features mentioned above, it could be established that LEWIS is an adaptable and feasible approach to get latency and energy-aware load balancing solution in a distributed cloud environment.

6 Conclusion

This paper addressed a latency optimizing energy-aware load balancing problem in a CC system. The problem is modeled as a cooperative game between cloud providers. A solution to the game called NBS is defined analytically. Through rigorous mathematical proofs, the existence of NBS is established in Theorem 1 and characterization of NBS is provided in Theorem 3. An efficient algorithm called LEWIS is proposed to compute the NBS systematically, and it is mathematically established (in Lemmas 1 and 2) that LEWIS yields the Pareto optimal solution in $\mathcal{O}(n \log n)$ time (where n is the number of cloud providers). It is also established theoretically as well as experimentally (in Theorem 4 and Section 5.4 and 5.5.3), that the workload allocation strategy of cloud providers is proportionally fair, and each user will get fairness in terms of response time and energy cost. To assess the solution quality of LEWIS, it is compared against Coop_RT, Coop_EN, and NCG across multiple performance metrics. The results confirm the superior performance of LEWIS over its counterparts, in terms of response time, fairness, and energy optimization. An immediate extension to this work is to consider the cloud system's dynamic properties where users' needs vary over time, following some statistical model.

Acknowledgments

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions.

References

[1] GeSI, "SMARTer 2020: The role of ICT in driving a sustainable future," Tech. Rep., The Climate Group on behalf of the Global eSustainability Initiative, 2012. [Online]. Available: http://gesi.org/SMARTer2020
[2] S. V. Vrbsky, M. Lei, K. Smith, and J. Byrd, "Data replication and power consumption in data grids," in *Proc. IEEE 2ndInt. Conf. Cloud Comput. Technol. Sci.*, 2010, pp. 288–295.
[3] P. Delforge and J. Whitney, "Data center efficiency assessment–scaling up energy efficiency across the data center industry: Evaluating key drivers and barriers," NRDC and Anthesis, New York, NY, USA, IP: 14–08, 2014.
[4] C. Preist and P. Shabajee, "Energy use in the media cloud: Behaviour change, or technofix?," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci.*, 2010, pp. 581–586.
[5] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2008.

[6] Y. C. Lee and A. Y. Zomaya, “Energy efficient utilization of resources in cloud computing systems,” *J. Supercomput.*, vol. 60, no. 2, pp. 268–280, 2012.
[7] L. A. Barroso and U. Hölzle, “The case for energy-proportional computing,” *Computer*, vol. 40, no. 12, pp. 33–37, 2007.
[8] Y. Jiang, “A survey of task allocation and load balancing in distributed systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 585–599, Feb. 2016.
[9] T. Kaur and I. Chana, “Energy efficiency techniques in cloud computing: A survey and taxonomy,” *ACM Comput. Surv.*, vol. 48, no. 2, pp. 1–46, Oct. 2015.
[10] A. Hameed *et al.*, “A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems,” *Computing*, vol. 98, no. 7, pp. 751–774, 2016.
[11] D. Grosu, A. T. Chronopoulos, and Ming-Ying Leung, “Load balancing in distributed systems: An approach using cooperative games,” in *Proc. 16th Int. Parallel Distrib. Process. Symp.*, 2002, pp. 501–510, Art. no. 10.
[12] D. Grosu and A. T. Chronopoulos, “Noncooperative load balancing in distributed systems,” *J. Parallel Distrib. Comput.*, vol. 65, no. 9, pp. 1022–1034, 2005.
[13] R. Subrata, A. Y. Zomaya, and B. Landfeldt, “A cooperative game framework for QoS guided job allocation schemes in grids,” *IEEE Trans. Comput.*, vol. 57, no. 10, pp. 1413–1422, Oct. 2008.
[14] S. Penmatsa and A. T. Chronopoulos, “Game-theoretic static load balancing for distributed systems,” *J. Parallel Distrib. Comput.*, vol. 71, no. 4, pp. 537–555, 2011.
[15] C. Liu, K. Li, Z. Tang, and K. Li, “Bargaining game-based scheduling for performance guarantees in cloud computing,” *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 3, no. 1, pp. 1–25, 2018.
[16] Z. Xiao, D. He, Y. Guo, and J. Du, “Request balancing among users in multiple autonomous cloud provider environments,” *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1140–1148, Feb. 2020.
[17] A. Kishor, R. Niyogi, and B. Veeravalli, “A game-theoretic approach for cost-aware load balancing in distributed systems,” *Future Gener. Comput. Syst.*, vol. 109, pp. 29–44, 2020.
[18] A. Kishor, R. Niyogi, and B. Veeravalli, “Fairness-aware mechanism for load balancing in distributed systems,” *IEEE Trans. Services Comput.*, pp. 1–14, Dec. 2020.
[19] S. U. Khan and I. Ahmad, “A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 3, pp. 346–360, Mar. 2008.
[20] B. Yang, Z. Li, and S. Jiang, “Cooperative game approach for energy-aware load balancing in clouds,” in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl. IEEE Int. Conf. Ubiquitous Comput. Commun.*, 2017, pp. 9–16.
[21] K. Li, C. Liu, K. Li, and A. Y. Zomaya, “A framework of price bidding configurations for resource usage in cloud computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2168–2181, Aug. 2016.
[22] R. Tripathi, S. Vignesh, V. Tamarapalli, A. T. Chronopoulos, and H. Siar, “Non-cooperative power and latency aware load balancing in distributed data centers,” *J. Parallel Distrib. Comput.*, vol. 107, pp. 76–86, 2017.
[23] B. Yang, Z. Li, S. Chen, T. Wang, and K. Li, “Stackelberg game approach for energy-aware resource allocation in data centers,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3646–3658, Dec. 2016.
[24] C. Liu, K. Li, K. Li, and R. Buyya, “A new cloud service mechanism for profit optimizations of a cloud provider and its users,” *IEEE Trans. Cloud Comput.*, vol. 9, no. 1, pp. 14–26, First Quarter 2017.
[25] H. Zhang, Y. Xiao, S. Bu, R. Yu, D. Niyato, and Z. Han, “Distributed resource allocation for data center networks: A hierarchical game approach,” *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 778–789, Third Quarter 2020.
[26] X. León and L. Navarro, “A stackelberg game to derive the limits of energy savings for the allocation of data center resources,” *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 74–83, 2013.
[27] S. Akbar, S. U. R. Malik, S. U. Khan, R. Choo, A. Anjum, and N. Ahmad, “A game-based thermal-aware resource allocation strategy for data centers,” *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 845–853, Third Quarter 2021.
[28] M. Zakarya *et al.*, “epcAware: A game-based, energy, performance and cost efficient resource management technique for multi-access edge computing,” *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2020.3005347.
[29] S. Mazumdar and M. Pranzo, “Power efficient server consolidation for cloud data center,” *Future Gener. Comput. Syst.*, vol. 70, pp. 4–16, 2017.
[30] S.-Y. Hsieh, C.-S. Liu, R. Buyya, and A. Y. Zomaya, “Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers,” *J. Parallel Distrib. Comput.*, vol. 139, pp. 99–109, 2020.
[31] S. K. Mishra *et al.*, “Energy-efficient VM-placement in cloud data center,” *Sustain. Comput. Inform. Syst.*, vol. 20, pp. 48–55, 2018.
[32] J. Cao, K. Li, and I. Stojmenovic, “Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers,” *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 45–58, Jan. 2014.
[33] K. Li, “Optimal task dispatching on multiple heterogeneous multiserver systems with dynamic speed and power management,” *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 167–182, Second Quarter 2017.
[34] G. Xie, J. Jiang, Y. Liu, R. Li, and K. Li, “Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems,” *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1068–1078, Jun. 2017.
[35] M. Mezmaz *et al.*, “A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems,” *J. Parallel Distrib. Comput.*, vol. 71, no. 11, pp. 1497–1508, 2011.
[36] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, and B. Luo, “Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds,” *IEEE Trans. Services Comput.*, vol. 11, no. 4, pp. 713–726, Jul./Aug. 2018.
[37] M. Shojafar, N. Cordeschi, and E. Baccarelli, “Energy-efficient adaptive resource management for real-time vehicular cloud services,” *IEEE Trans. Cloud Comput.*, vol. 7, no. 1, pp. 196–209, First Quarter 2019.
[38] K. Li, J. Mei, and K. Li, “A fund-constrained investment scheme for profit maximization in cloud computing,” *IEEE Trans. Services Comput.*, vol. 11, no. 6, pp. 893–907, Nov./Dec. 2018.
[39] M. Wardat, M. Al-Ayyoub, Y. Jararweh, and A. A. Khreishah, “To build or not to build? addressing the expansion strategies of cloud providers,” in *Proc. IEEE Int. Conf. Future Internet Things Cloud*, 2014, pp. 477–482.
[40] J. F. Nash Jr, “The bargaining problem,” *Econometrica: J. Econometric Soc.*, vol. 8, no. 2, pp. 155–162, 1950.
[41] H. Yaïche, R. R. Mazumdar, and C. Rosenberg, “A game theoretic framework for bandwidth allocation and pricing in broadband networks,” *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 667–678, Oct. 2000.
[42] F. Kelly, “Charging and rate control for elastic traffic,” *Eur. Trans. Telecommun.*, vol. 8, no. 1, pp. 33–37, 1997.
[43] C.-W. Ang and C.-K. Tham, “Analysis and optimization of service availability in a HA cluster with load-dependent machine availability,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 9, pp. 1307–1319, Sep. 2007.
[44] A. M. Johnson Jr, and M. Malek, “Survey of software tools for evaluating reliability, availability, and serviceability,” *ACM Comput. Surv.*, vol. 20, no. 4, pp. 227–269, 1988.
[45] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*. New York, USA: Pearson, 2005.
[46] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. Andrew, “Greening geographical load balancing,” *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 657–671, Apr. 2015.
[47] US Department of Energy, “US Energy information,” Accessed: Apr. 2020. [Online]. Available: https://www.eia.gov/totalenergy/data/monthly/index.php
[48] Internet Worlds Stats. Accessed: Apr. 2020. [Online]. Available: https://www.internetworldstats.com/unitedstates.htm#AZ

Avadh Kishor received the MTech degree in computer science and engineering from the Indian Institute of Information Technology and Management (IIITM), Gwalior, India, in 2015, and the PhD degree in computer science and engineering from the Indian Institute of Technology (IIT), Roorkee, India, in 2020. He is currently an assistant professor with the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology (TIET) Patiala. His research interests include distributed systems, multi-agent systems and algorithmic game theory. He is a member of ACM.

Rajdeep Niyogi received the PhD degree in computer science and engineering from the Indian Institute of Technology (IIT), Kharagpur, India, in 2004. He is currently a professor with the Department of Computer Science and Engineering, Indian Institute of Technology (IIT) Roorkee, India. His research interests include automated planning, algorithmic game theory, distributed systems, and applications of logic and automata theory. He is a member of ACM and ERCIM.

Anthony Theodore Chronopoulos (Senior Member, IEEE) received the PhD degree in computer science from the University of Illinois at Urbana-Champaign, in 1987. He is currently a full professor with the Department of Computer Science, University of Texas, San Antonio, USA , and a visiting professor with the Department of Computer Engineering & Informatics, University of Patras, Greece. He is the author of 100 journal and 74 peer-reviewed conference proceedings publications in the areas of Parallel and Distributed computing, grid and cloud computing, machine learning, numerical and scientific computing, computer networks and wireless communications, applications to science and engineering. He is the elected member of the European Academy of Sciences and Arts (Euro-Acad) (2021), fellow of the Institution of Engineering and Technology (FIET) (2017), and ACM senior member (2017).

Albert Y. Zomaya (Fellow, IEEE) is the chair professor of high performance computing & networking with the School of Information Technologies, University of Sydney, and he also serves as the director of the Centre for Distributed and High Performance Computing. His research interests include parallel and distributed computing and complex systems, he has published more than 600 scientific papers and articles and is author, co-author or editor of more than 20 books. He is the founding editor in chief of the *IEEE Transactions on Sustainable Computing* and serves as an associate editor for more than 20 leading journals. He served as an editor in chief for the *IEEE Transactions on Computers* (2011–2014). He is the recipient of the *IEEE Technical Committee on Parallel Processing Outstanding Service* Award (2011), the *IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing* (2011), and the *IEEE Computer Society Technical Achievement* Award (2014). He is a chartered engineer, a fellow of the AAAS, and IET.