2016 IEEE 22nd International Conference on Parallel and Distributed Systems

Game Theoretic Energy Allocation for Renewable Powered In-Situ Server Systems

Junlong Zhou†, Jianfei Chen†, Kun Cao†, Tongquan Wei†§, and Mingsong Chen‡
†Department of Computer Science and Technology, East China Normal University, Shanghai 200241, China
‡Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200241, China
§Corresponding email: tqwei@cs.ecnu.edu.cn.

***Abstract*—In-situ server systems are deployed in very special operating environment to handle in-situ workloads that are normally generated from environmentally sensitive areas or remote places that lack established utility infrastructure. This very special operating environment of in-situ servers urges such systems to be 100 percent powered by renewable energy. However, existing energy management schemes assume a hybrid supply of grid and renewable energy, hence are not well suited for 100 percent renewable powered in-situ server systems. In this paper, we tackle the problem of allocating harvested energy to 100 percent renewable powered server systems for optimizing both the overall system throughput and throughput of individual servers. From a game theoretic perspective, we model the energy allocation problem as a cooperative game among multiple servers and derive a Nash bargaining solution. Based on the Nash bargaining solution, we then propose a heuristic algorithm that determines the energy allocation strategies according to system energy states. Experimental results show that our proposed game theoretic approach achieves a high throughput from perspectives of both the overall system and individual servers.**

***Index Terms*—Renewable energy allocation, In-situ server, Game theory, Nash bargaining solution**

I. Introduction

As the essential infrastructures for large-scale computing, server clusters that attract considerable attention in both academia and industry in past decades have been used in delivering various computing and information technology (IT) services. These cluster-enabling services consume a tremendous amount of energy and incur a huge electricity bill. Consequently, renewable energy powered server clusters have become promising alternatives to conventional server clusters, and are expected to partially tackle the dual challenges of energy shortage and environmental issues. It has been reported that major IT companies like Google, Apple, and Facebook are reaching 100 percent renewable across all data centers [1].

In this paper we concentrate on energy management for in-situ server systems running on 100 percent renewable energy. Examples of in-situ server systems can be found in many data-driven applications such as oil/gas exploration [2], rural geographical surveying [3], astronomy observing in remote area [4], and video surveillance for wildlife behavioral studies [5]. In these in-situ applications, a group of inexpensive commodity servers are placed near data source for pre-processing collected data sets. In particular, these applications do not have hard real-time requirements, hence 100 percent intermittent renewable energy can be used to process delay-tolerant data sets [6].

Most existing works on renewable energy-aware power management concentrate on improving overall system energy utilization and/or throughput for data centers. Li et al. [7] have investigated the optimal solar power allocation on multi-core systems for maximally harvesting the solar energy and improving the overall throughput. Sharma et al. [8] designed a mechanism called blinking to manage server clusters running on intermittent power. A fast power state switching is utilized to match server power demand to intermittent power budget and a performance tradeoff between distributed cache's hit rate and access fairness is obtained. Goiri et al. [9]–[11] designed job scheduling schemes for data centers partially powered by solar energy. The presented schemes select the energy source to use, maximize the use of green energy, limit grid electricity costs, and avoid deadline violations. The major concern in these works is that intermittent renewable energy fluctuates with time and thus various designs need to be tailored such that the system energy demand matches the renewable energy budget. A fundamentally different design is presented in [12] that enables the renewable energy supply to follow the data center power demand by using the load following power provisioning technique.

Although the aforementioned proposals improve overall system energy efficiency from different perspectives, they make a common assumption that both renewable energy and grid electricity are adopted to maintain system functionalities. Thus, these energy management schemes are not well suited for data centers 100 percent powered by renewable energy. Image a scenario at a fine granularity level where 100 percent green powered remote servers host application software accessed by users. In such a scenario, user requests or workloads are distributed to application servers by a load balancer of the service provider. When the renewable energy is scarce, individual application servers play a game to compete for the energy available, aiming to enhance the probability of finishing their own workloads in time. Meanwhile, the service provider anticipates a high system throughput in the sense that as many overall workloads as possible can be executed using the limited renewable energy. It is clear to see that allocating scarce renewable energy to multiple servers is essentially a resource sharing problem, which can often be solved using Nash bargaining approach.

Nash bargaining is a powerful game theoretic technique that has been widely used for resource sharing problems [13], [14]. A number of recent works have investigated the application of game theoretic techniques to model the problems of data center bandwidth sharing [15], QoS-guided task allocation [16], energy-aware task scheduling [17], and service reservation [18]. Specifically, the bandwidth sharing problem [15], the QoS-guided task allocation problem [16], and the energy consumption scheduling problem [17] are all modeled as a cooperative game and the Nash bargaining solution to each game is derived. Based on the derived solutions, algorithms are then designed to realize corresponding optimization objectives. Unlike [15]–[17], the service reservation problem studied in [18] is formulated into a non-cooperative game among multiple users. The problem is solved by using the variational inequality theory and a Nash equilibrium solution to the formulated game is generated by using a proximal algorithm. Ge et al. [19] proposed a game-theoretic approach to minimize the overall energy consumption of a mobile computing system and developed an efficient algorithm that can achieve the Nash equilibrium in polynomial time. However, none of these works deal with the intermittent energy allocation in a system harvesting energy from ambient environments.

In this paper, we propose a game theoretic approach to tackle the energy allocation problem for in-situ servers systems powered by 100 percent renewable energy. In particular, we model the allocation problem as a cooperative game among multiple servers and derive a Nash bargaining solution to allocating the shared renewable sources.

This work was partially supported by Shanghai Municipal Natural Science Foundation (Grant No. 16ZR1409000), Natural Science Foundation of China (Grant No. 91418203 and 61672230), and East China Normal University Outstanding Doctoral Dissertation Cultivation Plan of Action (Grant No. PY2015047).

1521-9097/16 $31.00 © 2016 IEEE
DOI 10.1109/ICPADS.2016.97

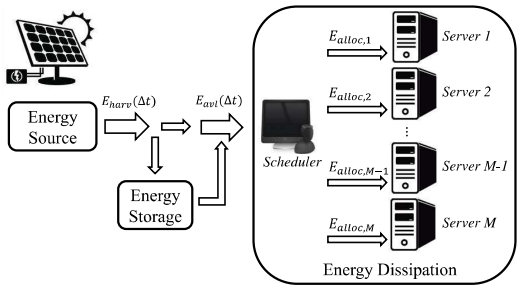


Figure 1: Architecture of a renewable energy powered system.

Finally, we propose a heuristic algorithm that determines the energy allocation strategies according to system energy states. Extensive simulation experiments validate the efficacy of our scheme.

The remainder of the paper is organized as follows. Section II presents the models and problem definition, Section III introduces the cooperative game framework, and Section IV describes the proposed energy allocation scheme. Section V verifies the effectiveness of the proposed scheme and Section VI concludes the paper.

II. Models and Problem Definition

Consider a renewable energy powered in-situ system that consists of three key modules, that is, energy source module, energy storage module, and energy dissipation module. As shown in Fig. 1, the energy source module automatically scavenges renewable energy from external environments, and converts the renewable generation (i.e., solar energy) to electrical energy. The energy storage module, which is typically in the form of a super capacitor or a battery, serves as a buffer against the uncertainty in harvested energy. A server cluster consisting of a scheduler and M heterogeneous in-situ servers is considered as the energy dissipation module. Since most of current in-situ systems are micro data centers processing none-critical data, the system energy consumption would not be high and the temporary system halt is allowed [6]. Therefore, we can achieve an almost sustainable running of such in-situ systems when no solar energy is harvested (e.g., at night) by assuming an energy storage of large capacity. As part of future work, we will extend our work to in-situ systems where critical data sets are supposed to be processed in real-time and temporary system halt is not allowed.

A. Energy Supply Model

Let $P_{harv}(t)$ denote the harvesting power and $E_{harv}(\Delta t)$ denote the energy harvested from the environments during time interval $[t, t+\Delta t]$, then $E_{harv}(\Delta t)$ is expressed as

$$E_{harv}(\Delta t) = \int_{t}^{t+\Delta t} P_{harv}(t)dt. \tag{1}$$

The harvested energy supports the computing system to satisfy the requests made by customers. If the harvested energy exceeds the energy demands of the system to provide a required service level, the energy surplus is stored in the energy storage module. Otherwise, there is no energy surplus and all harvested energy is allocated by the scheduler to individual servers. Thus, the energy available for allocation in the system during a time interval Δt is the sum of the energy harvested by energy source module during the current time interval and the energy surplus of previous time intervals stored in energy storage module. Specifically, let $E_{avl}(\Delta t)$ denote the system available supply energy during time interval $[t, t+\Delta t]$, $E(t)$ denote the energy stored in the battery at time instance t, and $E_{alloc,i}$ $(1 \leq i \leq M)$ denote the energy allocated to server i, then we have

$$E_{avl}(\Delta t) = E_{harv}(\Delta t) + E(t) \tag{2}$$

$$E_{alloc,1} + E_{alloc,2} + \cdots + E_{alloc,M} = E_{avl}(\Delta t). \tag{3}$$

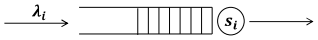


Figure 2: The M/M/1 queue of server i.

Table I: Parameters of two heterogeneous processors [25].

Processor	P_{sta}	C_{eff}	v	s
1	65.2	2.15×10^{-8}	0.89	1.8×10^{9}
2	37.2	4.08×10^{-8}	0.98	2.0×10^{9}

B. Task Model

We assume the arrival of tasks at each server as a Poisson process and the service times of servers have exponential distribution. Such assumption is based on other studies [20], [21] and verified by real-trace studies as in [22], [23]. Tasks considered in this model are independent, which means no dependency or communication among tasks. Let λ_i represent the average task arrival rate of server i and s_i represent the processing speed of server i, respectively. The task processing procedure of a server is thus described by an M/M/1 queuing model [24], as illustrated in Fig. 2. External tasks arrive with a rate λ_i at server i, wait in a queue with an infinite capacity, and get processed with a speed s_i. A task is denoted by τ. Execution requirements of tasks are quantified by the number of cycles to be executed, which are independent and approximately follow normal distributions [25]. Thus we let $N(c_i, \sigma_i^2)$ represent the normal distribution of execution cycles of tasks on server i, where c_i is the expected value and σ_i is the standard variance.

Since the computing system considered in this paper is powered by renewable energy sources, the energy available may not meet the energy demands for task execution. Thus, the number of tasks executing on a server is the smaller value between the number of arrival tasks and the number of tasks that can be processed by the server. Let $N_i(\Delta t)$ be the number of tasks executing on server i during Δt, and it is given by

$$N_i(\Delta t) = \min\left(\int_{t}^{t+\Delta t} \lambda_i dt, \frac{s_i \Delta t}{c_i}\right), \tag{4}$$

where $\int_{t}^{t+\Delta t} \lambda_i dt$ is the number of arrival tasks during Δt and $\frac{s_i \Delta t}{c_i}$ is the number of tasks that could be handled by server i during Δt at processing speed s_i.

C. Energy Dissipation Model

A typical example of energy dissipation is the execution of tasks on the server processor. The server processor is an CMOS device, and the power consumption of an CMOS device can be modeled as the sum of static power consumption P_{sta} and dynamic power consumption P_{dyn} [26], that is,

$$P_{cons} = P_{sta} + P_{dyn}. \tag{5}$$

P_{sta} is independent of switching activity and maintains the circuit basic state. P_{dyn} is related to processor switching activity and can be formulated as a function of supply voltage v and processing speed s, that is, $P_{dyn} \propto v^2 s$. Since $v \propto s$, the overall processor power consumption of server i is

$$P_{cons,i} = P_{sta,i} + P_{dyn,i} = P_{sta,i} + C_{eff,i} s_i^3, \tag{6}$$

where the static power dissipation $P_{sta,i}$ is a constant and $C_{eff,i}$ is the processor effective capacitance. Parameters of two example heterogeneous processors are shown in Table I.

The energy consumed by processor of server i when executing a task instance τ at processing speed s_i, denoted by $E_{cons}(\tau, s_i)$, is the product of power consumption and task execution time, and can be approximately estimated as

$$E_{cons}(\tau, s_i) = (P_{sta,i} + C_{eff,i} s_i^3) \times \frac{c_i}{s_i} = \frac{P_{sta,i} c_i}{s_i} + C_{eff,i} s_i^2 c_i. \tag{7}$$

Then the processor energy consumption of server i during time interval Δt is given by

$$E_{cons,i}(\Delta t) = N_i(\Delta t) \times E_{cons}(\tau, s_i), \tag{8}$$

and the energy demand of M servers is formulated as

$$E_{dem}(\Delta t) = \sum_{i=1}^{M} E_{cons,i}(\Delta t). \tag{9}$$

Due to the intermittent nature of renewable energy sources, the energy available to support system operation varies in a predefined time domain (i.e., Δt). To analyze the system at a fine granularity, the operation of the system with respect to energy is divided into two states: high energy state and low energy state. Specifically, if the energy available during the scheduling horizon is plentiful for the system to finish the workloads of all servers, that is,

$$E_{avl}(\Delta t) \geq E_{dem}(\Delta t), \tag{10}$$

the system is deemed to be in high energy state; otherwise, in low energy state. Accordingly, the scheduler needs two energy allocation strategies to handle the uncertainty in energy availability. Designing an energy allocation strategy for high energy state is easy since the scheduler only needs to assign the amount of energy to individual servers as they require. However, designing an energy allocation strategy for low energy state is challenging. This is because individual servers compete for insufficient energy to maximize their respective throughput. Meanwhile, from the perspective of the system, it is desirable for servers to cooperate to maximize the overall system throughput. Therefore, our goal is to design an efficient energy allocation strategy for low energy state to cope with competition and cooperation among servers.

D. Problem Definition

As explained above, we focus on designing an energy allocation strategy for 100 percent renewable powered server system. We assume a scenario that users have access to application software deployed on 100 percent green powered remote servers, and user requests or workloads have been distributed to application servers by a load balancer of the service provider. When the system is in low energy state, that is, the renewable energy is scarce, individual application servers play a game to compete for the energy available, expecting to acquire enough energy and timely complete their own workloads. Simultaneously, the service provider also anticipates a high system throughput under the limited renewable energy, which can be achieved via cooperation among multiple application servers. Therefore, to satisfy the requirements of both service provider and users, an energy allocation strategy that optimizes both the system overall throughput and throughput of individual servers is needed.

Problem Definition: Given a server system that is 100 percent powered by renewable energy and assuming that user requests or workloads have been distributed to application servers by a load balancer, maximize both the system overall throughput and throughput of individual servers when renewable energy is insufficient.

III. Our Cooperative Game Framework

The problem of allocating harvested energy to multiple servers can be modeled as a cooperative game. Nash bargaining is a powerful technique that has been widely used for resource sharing problems [13]. In this work, we propose a game theoretic framework to address the energy allocation problem, as described in Section II-D. Below, we first introduce the concepts of cooperative game theory, then model the energy allocation game among multiple servers, and finally derive the optimal Nash bargaining solution to the cooperative game.

A. Cooperative Game Theory Concepts

In this subsection, we briefly summarize basic definitions and concepts of cooperative game theory and Nash bargaining solution which are used in the sequel. For further details, readers are suggested to refer to [27] and [28].

A cooperative game consists of: (i) M players. (ii) A nonempty, closed, and convex set, denoted by $\mathcal{X} \subset \mathcal{R}^M$, which is the set of strategies of the M players. (iii) A performance function f_i of each player i, where f_i is a function from $\mathcal{X}$ to $\mathcal{R}$ and is upper-bounded. (iv) A minimum value of f_i, denoted by μ_i^0, is the minimal performance required for the player to enter the game without any cooperation. The vector $\mu^0 = (\mu_1^0, \mu_2^0, \cdots, \mu_M^0)$ is called the initial agreement point.

In the context of our allocation of shared energy supply, $\mathcal{X}$ represents the vector space of allocated energy. The initial performance of each server (player) is a minimum guarantee that the system must provide the server to enable a predictable quality of service. Therefore, we will assume that each server involved in the game can achieve its initial performance. In other words, there exists at least one vector $x \in \mathcal{X}$, whose performance vector $f(x) = (f_1(x), f_2(x), \cdots, f_M(x))$ is superior or equal to the initial performance vector μ^0. Let $U \subset \mathcal{R}^M$ be a nonempty convex closed and upper-bounded set, which is the set of achievable performance in our context, and let $\mu^0 \in \mathcal{R}^M$. Since it is assumed that each player involved in the game can achieve its initial performance at least, we have $U_0 = \{\mu \in U \mid \mu \geq \mu^0\} = \varnothing$, where U_0 is the set of performance vectors that is no less than the initial agreement point. The above cooperative game is in general resolved by Nash bargaining, and the generated solutions to the cooperative game are called Nash bargaining solutions.

Nash bargaining solution (NBS) [27]: A mapping $\mathcal{S} : (U, \mu^0) \to \mathcal{R}^M$ is an NBS if $S(U, \mu^0) \in U_0$, and $S(U, \mu^0)$ is Pareto optimal and satisfies the fairness axioms [28].

After giving the definition of NBS, we now introduce how to derive the NBS. Let $\mathcal{X}_0 = \{x \in \mathcal{X} \mid f(x) \geq \mu^0\}$ represent the set of strategies enabling the players to achieve at least their initial performance, which corresponds to the performance set U_0. Let $\mathcal{J} = \{j \in \{1, 2, \cdots, M\} \mid x \in \mathcal{X}_0, f_j(x) > \mu_j^0\}$ represent the set of players who are able to achieve a performance strictly superior to their initial performance. In addition, suppose that for each $j \in \mathcal{J}$, f_j is one-to-one on $\mathcal{X}_0$. Under these definitions and assumption, there exists a unique Nash bargaining solution μ^* with its strategy $x^* = \mathrm{f}^{-1}(\mu^*)$, which is derived by solving the following optimization problem [13]:

$$P_{\mathcal{J}}: \quad \max_x \prod_{j \in \mathcal{J}} (f_j(x) - \mu_j^0) \quad x \in \mathcal{X}_0. \tag{11}$$

In the Nash bargaining problem, multiple players (typically no less than two) enter the game with an initial performance requirement and a performance function. The players cooperate in the game to achieve a win-win solution, which maximizes the performance gains given in (11) and leads to a relative fairness among players. The game also ensures a minimum service guarantee by complying with the constraint of initial performance agreement. This corresponds to design requirements of the concerned system, where the overall system throughput is maximized and throughput of individual servers is lower-bounded. By taking the logarithm of the objective function, an equivalent optimization problem can be derived and is

$$P'_{\mathcal{J}}: \quad \max_x \sum_{j \in \mathcal{J}} \ln(f_j(x) - \mu_j^0) \quad x \in \mathcal{X}_0. \tag{12}$$

$P'_{\mathcal{J}}$ is a convex optimization problem and has a unique solution. The unique solution of $(P'_{\mathcal{J}})$ is the bargaining solution.

B. Energy Allocation as a Cooperative Game

We consider a cooperative game in which each server is a player and the M servers are competing for the shared available energy $E_{avl}(\Delta t)$ in the system. In the context of energy allocation to maximize the overall system throughput and throughput of individual servers, and satisfy the throughput requirement of individual servers, we define the performance function of server i as the throughput of

the server under the budget of its allocated energy $E_{alloc,i}$, which is the number of tasks on the server and formulated as

$$f(E_{alloc,i}) = \frac{E_{alloc,i}}{E_{cons}(\tau, s_i)} = \frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i}, \tag{13}$$

where $E_{cons}(\tau, s_i)$ is the energy consumed by executing task τ at processing speed s_i. Suppose that each server i has an initial performance requirement μ_i^0, which corresponds to the lowest throughput of server i that needs to be guaranteed. We also assume that the M servers are able to achieve performance strictly higher than their initial performance.

Associated with the above cooperative game, the problem of maximizing the throughout of the system and satisfying the throughout requirement of individual servers can be described as follows. Given the shared available energy $E_{avl}(\Delta t)$ in a time interval of Δt and the base throughput requirement $(\mu_1^0, \mu_2^0, \cdots, \mu_M^0)$ of M servers, the M servers cooperate in the game to obtain a maximized overall system throughput by agreeing with an optimal point of energy allocation $(E_{alloc,1}, E_{alloc,2}, \cdots, E_{alloc,M})$. In other words, the optimization problem can be formulated as

$$\textbf{P1}: \quad \max \quad \prod_{i=1}^{M} \left(\frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} - \mu_i^0\right) \tag{14}$$

$$\text{s.t.} \quad \sum_{i=1}^{M} E_{alloc,i} = E_{avl}(\Delta t), \tag{15}$$

$$\frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} \leq N_i(\Delta t), \tag{16}$$

$$\mu_i^0 \leq \frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i}, \tag{17}$$

where (15) indicates all of the available energy can be allocated to servers for task execution, (16) indicates the energy allocated to each server cannot exceed the energy of executing tasks on the server processor, and (17) indicates the throughput achieved by each server is higher than its base requirement.

In the above formulation, $\max \prod_{i=1}^{M} (\frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} - \mu_i^0)$ is selected as the objective rather than $\max \sum_{i=1}^{M} (\frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} - \mu_i^0)$. This is because the former not only demonstrates the capability of the scheduler for maximizing system throughput, but also shows the expectation of the M servers for maximizing their respective throughput. According to the analysis given in the end of Section III-A, the objective in (14) is equivalent to $\max \sum_{i=1}^{M} \ln(\frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} - \mu_i^0)$, which can be converted into

$$- \quad \min \quad \sum_{i=1}^{M} \ln\left(\frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} - \mu_i^0\right). \tag{18}$$

P1 is an optimization problem that attempts to maximize a function subject to multiple constraints, and many methods can be used to solve the problem. Since Lagrange multiplier is powerful for solving this type of problem with low computation complexity, we adopt it to obtain the best solution to the problem. The Lagrangian of **P1** problem is expressed as

$$\begin{aligned}\mathcal{L}(E_{alloc,i}, \alpha, \beta_i, \gamma_i) = &- \sum_{i=1}^{M} \ln\left(\frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} - \mu_i^0\right) \\ &+ \alpha\left(\sum_{i=1}^{M} E_{alloc,i} - E_{avl}(\Delta t)\right) + \sum_{i=1}^{M} \beta_i \left(\frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i}\right. \\ &\left. - N_i(\Delta t)\right) + \sum_{i=1}^{M} \gamma_i \left(\mu_i^0 - \frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i}\right),\end{aligned} \tag{19}$$

where $\alpha \in \mathbb{R}$, and $\beta_i, \gamma_i \geq 0$ hold for $\forall i = 1, 2, \cdots, M$. They are the Lagrange multipliers associated with the constraints given in (15), (16), and (17).

It is clear that the optimal solution is derived when the derivative of $\mathcal{L}(E_{alloc,i}, \alpha, \beta_i, \gamma_i)$ with respect to $E_{alloc,i}$ equals to zero. In this case, the expression

$$\begin{aligned}\nabla\mathcal{L}(E^*_{aloc,i}, \alpha^*, \beta_i^*, \gamma_i^*) = 0 \Longleftrightarrow &- \frac{1}{E^*_{alloc,i} - E^0_{alloc,i}} + \alpha^* \\ &+ \frac{\beta_i^*}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} - \frac{\gamma_i^*}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} = 0,\end{aligned} \tag{20}$$

and the Karush-Kuhn-Tucker (KKT) conditions [29]

$$\alpha^*\left(\sum_{i=1}^{M} E_{alloc,i} - E_{avl}(\Delta t)\right) = 0, \ \alpha^* \in \mathbb{R}, \tag{21}$$

$$\beta_i^*\left(\frac{E^*_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} - N_i(\Delta t)\right) = 0, \ \beta_i^* \geq 0, \tag{22}$$

$$\gamma_i^*\left(\mu_i^0 - \frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i}\right) = 0, \ \gamma_i^* \geq 0, \tag{23}$$

hold. Therefore, the best solution to **P1** problem can be derived from (20), and it is given by

$$E^*_{alloc,i} = \frac{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i}{\alpha^*\left(\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i\right) + \beta_i^* - \gamma_i^*} + E^0_{alloc,i}, \tag{24}$$

where α^*, β_i^*, γ_i^* are the optimal Lagrange multipliers and $E^0_{alloc,i}$ is the energy required to achieve initial performance μ_i^0. The best solution $E^*_{alloc,i}$ is in fact the optimal energy allocated to server i and can be calculated once the optimal Lagrange multipliers given in (21), (22), (23) are obtained. Note that $\mu_i^0 = \frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i}$ is a special case of (23) that the performance of server i equals to its initial value μ_i^0. Since this paper focuses on a general situation that $\mu_i^0 < \frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i}$, then we have $\gamma_i^* = 0$ from (23).

As indicated in (24), the key of obtaining the best solution is to derive the optimal Lagrange multipliers of each server, which is generally achieved by converting the optimization problem into its dual problem and solving the dual problem via gradient projection.

C. Derive the Optimal Nash Bargaining Solution

The original optimization problem **P1** is a typical convex optimization problem that can be deemed as a primal problem and has a corresponding dual problem. The solution to the dual problem provides a lower bound to the solution of the primal problem, and the difference between two solutions is called the duality gap. Especially, when the duality gap equals to zero, the optimal solution of the primal problem is given by the dual problem. Hence, we first derive the Lagrange dual problem that corresponds to the primal problem with no duality gap, then solve the dual problem using gradient project.

The Dual Lagrangian function is defined as the minimum value of Lagrangian over argument $E_{alloc,i}$, that is,

$$d(\alpha, \beta_i, \gamma_i) = \inf_{E_{alloc,i}} \mathcal{L}(E_{alloc,i}, \alpha, \beta_i, \gamma_i). \tag{25}$$

It yields lower bounds on each optimal $E^*_{alloc,i}$ of Lagrangian, which solves (18). Since the infimum of Lagrangian occurs where the gradient equals to zero, we have (24). Then, substituting (24) and $\gamma_i^* = 0$ into (19), the formulation of dual function can be obtained and is expressed as

$$\begin{aligned}d(\alpha, \beta_i) = &\sum_{i=1}^{M} \ln\left(\alpha\left(\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i\right) + \beta_i\right) - \\ &\alpha E_{avl}(\Delta t) - \sum_{i=1}^{M} \beta_i N_i(\Delta t) + \sum_{i=1}^{M} \alpha E^0_{alloc,i} \\ &+ \sum_{i=1}^{M} \frac{\beta_i E^0_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} + M.\end{aligned} \tag{26}$$

It is obvious that there exists such $E_{alloc,i}$ that the equalities $\frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i} < N_i(\Delta t)$ and $\mu_i^{init} < \frac{E_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i} + C_{eff,i}s_i^2 c_i}$ hold for $\forall i = 1, 2, \cdots, M$. This indicates that constraints (16) and (17)

hold with strict inequalities, which implies that the Slater's condition holds. Since the **P1** problem is convex and the Slater's condition holds, the KKT saddle point conditions are satisfied and become sufficient and necessary for strong duality [29]. Thus, there is no duality gap and the dual problem has at least one optimal solution.

In summary, the dual problem corresponding to the primal problem **P1** with no duality gap is depicted as

$$\begin{aligned}\min\ & d(\alpha,\beta_i)^* = -\sum_{i=1}^{M}\ln(\alpha(\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i)+\beta_i)\\ &+\alpha E_{avl}(\Delta t)+\sum_{i=1}^{M}\beta_i N_i(\Delta t)-\sum_{i=1}^{M}\alpha E^0_{alloc,i}\\ &-\sum_{i=1}^{M}\frac{\beta_i E^0_{alloc,i}}{\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i}-M \qquad (27)\end{aligned}$$

$$\text{s.t.}\quad \beta_i \geq 0, \qquad (28)$$

$$\begin{aligned}&\frac{\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i}{\alpha(\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i)+\beta_i}+E^0_{alloc,i}\\ &\leq(\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i)N_i(\Delta t), \qquad (29)\end{aligned}$$

where $d(\alpha,\beta_i)^* = -d(\alpha,\beta_i)$, and (29) indicates the constraint that the energy assigned to server i can't exceed the maximum energy it requires. Note that the dual problem is an optimization problem with liner constraints, which motivates us to address the dual problem by using gradient project.

Gradient project [29] is widely used to solve the optimization problem with linear constraints. However, due to the greedy search used in gradient project to find the optimal solution, the generated solution may fall into local optimality. Thus, the analysis to verify the solution derived by gradient project is global optimal becomes a necessity. Considering the property that a local optimal solution of a convex optimization function is globally optimal, we only need to check whether the dual function is convex or not. A function is convex iff the domain of the function is convex and the Hessian matrix of the function is positive semi-definite [29]. Obviously, the domain of our dual function $d(\alpha,\beta_i)^*$ is convex. The Hessian matrix of $d(\alpha,\beta_i)^*$ is positive semi-definite since all elements in the matrix are no less than zero, as shown below.

$$\begin{cases}\frac{\partial d^{*2}}{\partial\alpha^2}=\frac{(\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i)^2}{(\alpha(\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i)+\beta_i)^2}\geq 0, & \forall i=1,\cdots,M\\ \frac{\partial d^{*2}}{\partial\alpha\partial\beta_i}=\frac{\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i}{(\alpha(\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i)+\beta_i)^2}>0, & \forall i=1,\cdots,M\\ \frac{\partial d^{*2}}{\partial\beta_i\partial\alpha}=\frac{\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i}{(\alpha(\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i)+\beta_i)^2}>0, & \forall i=1,\cdots,M\\ \frac{\partial d^{*2}}{\partial\beta_i^2}=\frac{1}{(\alpha(\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i)+\beta_i)^2}>0, & \forall i=1,\cdots,M\\ \frac{\partial d^{*2}}{\partial\beta_i\partial\beta_j}=0. & \forall i,j=1,\cdots,M\end{cases}$$

As analyzed above, the dual function $d(\alpha,\beta_i)^*$ is convex. Thus, its solution produced by gradient project is globally optimal, which is also the optimal Nash bargaining solution to the primal problem.

IV. Our Proposed Energy Allocation

As described in Section III, a cooperative game is utilized to model the energy allocation problem for renewable energy powered computing systems in low energy state, and a Nash bargaining solution is derived to develop our proposed energy allocation scheme. Since our energy allocation scheme is supposed to run at discrete time instances, we first calculate the scheduling interval, then present our scheme in detail.

A. The Calculation of Schedule Interval

Before presenting our energy allocation scheme, the schedule interval of the scheme is derived at first. Since harvesting power and external task arrivals are not constant, system state including energy state and task state is varying over time. It is rational to adjust energy allocation scheme according to system state, which inspires us to adopt the duration of system stable state as the time interval of energy allocation.

Suppose Δt is the duration of stable state of system from present moment, the energy of the whole system collected in Δt is calculated as $E_{harv}(\Delta t) = \int_t^{t+\Delta t} P_{harv}(t)dt$, where $P_{harv}(t)$ is the curve of harvesting power. Let $E_{min}(\Delta t) = M \times \min\{(\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i)\times N_i(\Delta t)\mid i=1,2,\cdots,M\}$ and $E_{max}(\Delta t) = M\times\max\{(\frac{P_{sta,i}c_i}{s_i}+C_{eff,i}s_i^2c_i)\times N_i(\Delta t)\mid i=1,2,\cdots,M\}$, where $N_i(\Delta t)$ is the number of tasks executing on server i, as given in (4). Obviously, $E_{harv}(\Delta t)$ must be between $E_{min}(\Delta t)$ and $E_{max}(\Delta t)$ when the system exactly consumes up all the harvested energy for task execution. Given these, we define the system is stable when $E_{harv}(\Delta t)$ is between $E_{min}(\Delta t)$ and $E_{max}(\Delta t)$. Accordingly, the duration of stable state is the time interval before $E_{harv}(\Delta t)$ beyond the range from $E_{min}(\Delta t)$ to $E_{max}(\Delta t)$, which is given by

$$\{\Delta t \mid E_{harv}(t+\Delta t)\in(E_{min}(t+\Delta t),E_{max}(t+\Delta t))\}. \qquad (30)$$

The Δt derived in such a way could neither be too small nor too large, which ensures the overhead of invoking the algorithm wouldn't be high and the estimation of energy harvested wouldn't be imprecise.

Algorithm 1: Assign System Available Energy to Servers

Input: $\lambda_1,\lambda_2,\cdots,\lambda_M$ & $s_1,s_2,\cdots,s_M$ & $P_{harv}(t)$
Output: $E_{alloc,1},E_{alloc,2},\cdots,E_{alloc,M}$

1 calculate the system available energy $E_{avl}(\Delta t)$ using (1), (2), (30);
2 initialize the system demand energy E_{dem} to 0;
3 **for** $i=1$ to M **do**
4 derive the number $N_i(\Delta t)$ of tasks arrived at server i based on Δt, λ_i, s_i, c_i, and (4);
5 compute the energy $E_{cons,i}(\Delta t)$ consumed by server i based on $N_i(\Delta t)$, s_i, c_i, and (8);
6 update the E_{dem} by $E_{dem}=E_{dem}+E_{cons,i}(\Delta t)$;
7 **end**
8 **if** $E_{dem}\leq E_{avl}(\Delta t)$ **then**
  /* system is in high energy state */
9 **for** $i=1$ to M **do**
10 $E_{alloc,i}=E_{cons,i}(\Delta t)$;
11 **end**
12 **end**
13 **else**
  /* system is in low energy state */
14 **for** $i=1$ to M **do**
15 obtain the optimal energy $E^*_{alloc,i}$ allocated to server i by (24);
16 $E_{alloc,i}=E^*_{alloc,i}$;
17 **end**
18 **end**

B. The Assignment of System Available Energy

As modeled in Section II, the multiserver system has two energy states: high energy state and low energy state. When the system is in high energy state, the scheduler assigns the amount of energy to servers as they require. When the system is in low energy state, the scheduler assigns the amount of energy to servers, which is determined through a cooperative Nash bargaining game. In order to address the energy allocation issue in two states, we first compare the system energy available and the system energy requested to identify which state the system is in, then derive the energy assigned to each server as described above.

Our proposed energy allocation scheme given in Alg. 1 operates as follows. It takes as input task arrival rates $\lambda_1,\lambda_2,\cdots,\lambda_M$ and processing speed $s_1,s_2,\cdots,s_M$ of M servers, and harvesting power $P_{harv}(t)$. Lines 1-2 of the algorithm calculate the system available energy $E_{avl}(\Delta t)$ using (1), (2), (30), and initialize the system energy

consumption E_{cons} to 0. Lines 3-7 estimate the energy consumed by M servers to complete their workloads. When the system energy demand is no greater than the system energy available (i.e., $E_{dem} \leq E_{avl}(\Delta t)$), indicating the system energy is in high energy state, the scheduler can satisfy the energy requirements of M servers by $E_{alloc,i} = E_{cons,i}(\Delta t)$ (lines 8-12). When the system energy demand exceeds the system energy available (i.e., $E_{dem} > E_{avl}(\Delta t)$), indicating the system is in low energy state, the M servers compete for the shared supply energy as a cooperative Nash bargaining game. The energy $E_{alloc,i}$ assigned to each server is set to $E^*_{alloc,i}$, which is the optimal solution generated by the game (lines 13-18).

V. Numerical Results

A. Experimental Settings

Extensive simulations have been implemented to validate the effectiveness of the proposed energy allocation scheme. The objective of the proposed scheme is to optimize both the system overall throughput and throughput of individual servers when renewable energy is insufficient. We have demonstrated in Section III that the proposed energy allocation scheme leads to a relatively fair renewable distribution among application servers. For the sake of easy presentation, we use efficiency to show the requirement of optimizing the system overall throughput via a cooperative game, while use fairness to indicate the requirement of maximizing the throughput of individual servers via a competitive game. Obviously, the higher the system throughput, the more efficient the proposed scheme, and the smaller the variance of throughput achieved by individual servers, the more fair the proposed scheme. Give these, we define a metric that jointly takes into account efficiency and fairness to quantify the performance of the proposed scheme, which is formulated as

$$Metric_P = \frac{Metric_E}{Metric_F}, \tag{31}$$

where the efficiency metric $Metric_E$ is calculated as the throughput achieved by the whole system and the fairness metric $Metric_F$ is calculated as the variance of throughput achieved by servers. From (31), it is clear that larger $Metric_P$ indicates better efficiency and fairness.

Three sets of simulations are carried out to validate the effectiveness of the proposed scheme, including

1) Uniqueness Verification of NBS: we conduct the simulation under varying system setups and initial values of gradient projection to show the solutions derived by our algorithm can converge to a global optimal solution, that is, the unique NBS generated by the cooperative game.
2) Performance Evaluation of the Propose Scheme: we compare the performance of the proposed scheme with that of a benchmarking method, referred to as Naive, under a fixed workload and varying energy supplies, or under varying workloads and a fixed energy supply.
3) Validation of Efficiency and Fairness: we compare the efficiency and fairness of the proposed scheme with that of two benchmarking methods MT (Max Throughput) and AVG (Average).

As mentioned above, three benchmarking schemes are used in the second and third sets of simulations. In the first benchmarking scheme, the system energy supply is prorated to servers according to the ratio of the energy required by each server to the energy required by the whole system. This is a simple proportionate allocation scheme, thus we call it Naive. In the second benchmarking scheme, the server consuming less energy for completing a given workload has a higher priority to receive energy from the scheduler. If energy allocation is performed in this way, the system throughput under a limited energy budget is maximized, thus we call it MT. Obviously, this scheme only pursues high efficiency without considering fairness among servers. On the contrary, the third benchmarking scheme that only strives for fairness without considering efficiency equally assigns the system available energy to servers, thus we call it AVG.

Table II: Task parameters of benchmarks

Benchmark	Expected execution time	Standard derivation
Mpegplay	113.4	38.9
Madplay	43.1	34.8
Tmndec	89.5	34.7
Toast	5.6	4.7

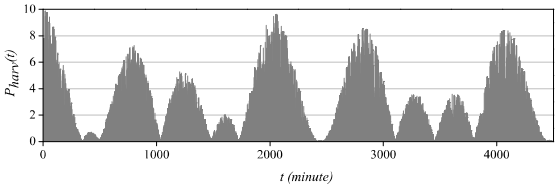


Figure 3: Example of the obtained solar power trace [31].

Three simulated systems that respectively consist of 5 servers ($M = 5$), 10 servers ($M = 10$), and 15 servers ($M = 15$) are utilized in the experiments. The simulated processors in the three systems are modeled based on the following cores: Intel Core Duo, Intel Xeon, AMD Athlon, TI DSP, and SPARC64. Four practical benchmarks Mpegplay, Madplay, Tmndec, and Toast are employed to generate tasks for evaluations, and their task parameters including expected execution time and standard derivation are shown in Table II. The workload of the three systems, quantified by the number N of tasks on the processor, are in the range of $[200, 300]$, $[600, 1100]$, $[2000, 3000]$, respectively. We adopt the task assignment [25] to model the arrival of tasks at servers. The most common and accessible energy, solar energy, is selected as the renewable generation, and the trace of harvesting power $P_{harv}(t)$ is generated according to [30], [31]

$$P_{harv}(t) = \left| 10 \cdot \Theta(t) \cdot \cos(\frac{t}{70\pi}) \cdot \cos(\frac{t}{100\pi}) \right|, \tag{32}$$

where $\Theta(t)$ is a random variable with variance of 1 and mean of 0. Fig. 3 demonstrates the obtained power trace in a frame of about 4500 time units.

B. Simulation Results

1) Validate the Uniqueness of NBS: If the OPT problem has a global optimal solution, the energy allocation cooperative game can obtain a unique Nash bargaining solution. Since a greedy approach

Table III: The number of iterations taken by the proposed scheme to reach a converged solution.

System setup \ Initial value	1	2	3	4	5	6	7	8
$M = 5$, $E_{avl} = 2.0e5$	6	8	6	6	7	3	4	2
$M = 10$, $E_{avl} = 5.0e5$	23	25	28	13	8	6	6	6
$M = 15$, $E_{avl} = 1.2e6$	25	26	34	29	19	20	14	14

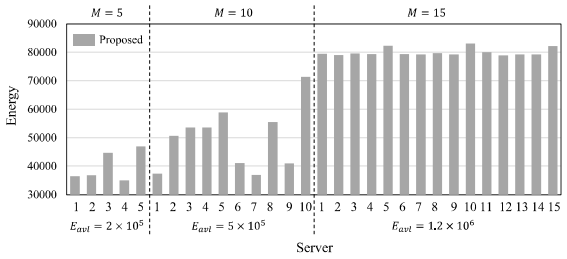
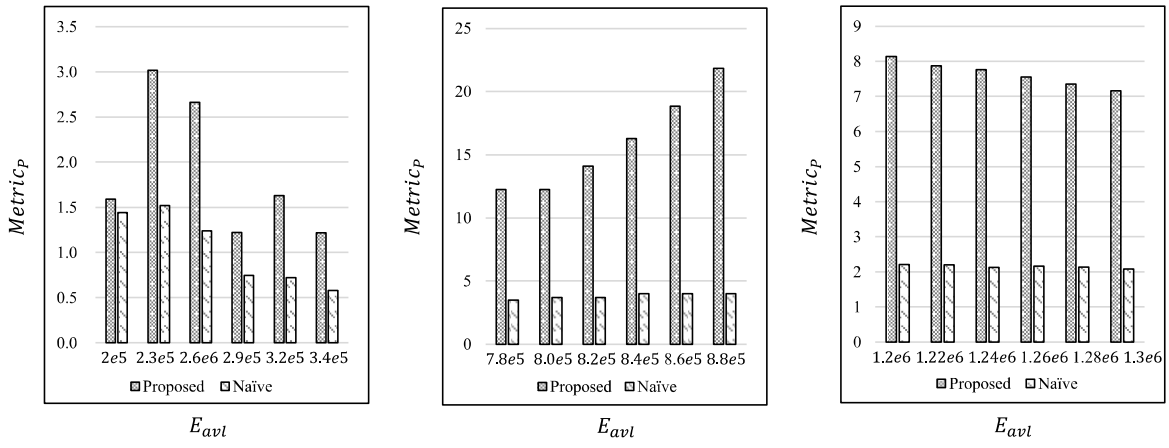
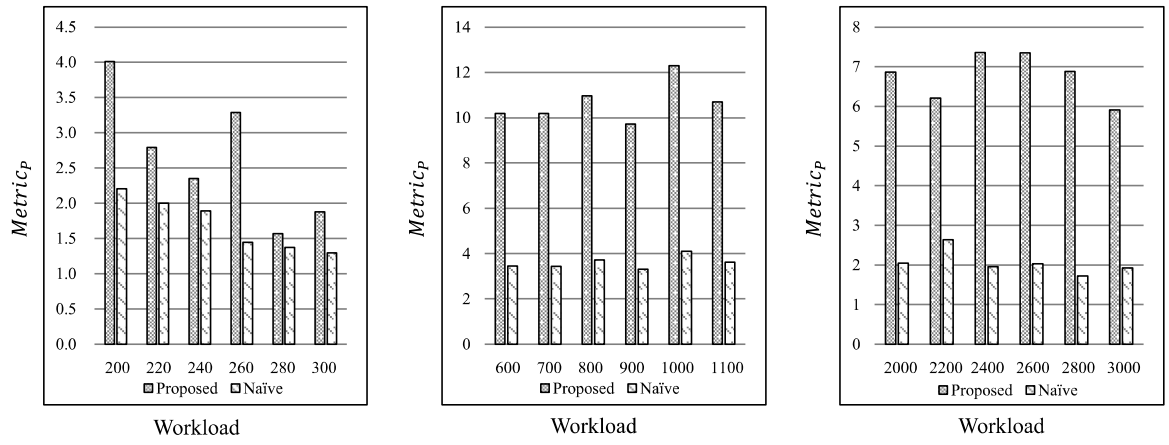


Figure 4: Energy allocation scheme of three systems under the energy supply E_{avl} of 2×10^5, 5×10^5, and 1.2×10^6.



(a) $M = 5$, Workload $= 200$ (b) $M = 10$, Workload $= 600$ (c) $M = 15$, Workload $= 2000$

Figure 5: Compare the proposed energy allocation scheme with the benchmarking algorithm Naive in performance $Metric_P$ under a fix workload and varying energy supplies E_{avl}.



(a) $M = 5, E_{avl} = 2 \times 10^5$ (b) $M = 10, E_{avl} = 5 \times 10^5$ (c) $M = 15, E_{avl} = 1.2 \times 10^6$

Figure 6: Compare the proposed energy allocation scheme with the benchmarking algorithm Naive in performance $Metric_P$ under a fix energy supply E_{avl} and varying workloads.

(i.e., gradient projection) is used in the proposed algorithm to solve the OPT problem, we need to verify if the solutions generated by the proposed algorithm can converge to a global optimal solution. The verification can be readily achieved by checking if the algorithm can stop and derive a converged solution after a number of iterations regardless of the initial value of gradient projection adopted.

To have a comprehensive study, we investigate the iteration number of the proposed algorithm when using eight initial values and three system settings. Since the selection of initial value has no effect on gradient projection, the eight initial values are casually set to 1, 2, 3, 4, 5, 6, 7, and 8. The three system settings used in the simulation are ($M = 5$, $E_{avl} = 2 \times 10^5$), ($M = 10$, $E_{avl} = 5 \times 10^5$), and ($M = 15$, $E_{avl} = 1.2 \times 10^6$). Table III demonstrates the number of iterations taken by the proposed algorithm to reach a converged solution. As shown in the table, for any given initial value and system setting, a converged solution can be obtained by the proposed algorithm after a number of iterations. The converged solution is exactly the optimal energy allocation scheme. As an example, Fig. 4 presents the energy allocation scheme for three systems under the energy supply E_{avl} of 2×10^5, 5×10^5, and 1.2×10^6.

2) Evaluate the Performance: Two comparisons are performed to evaluate the performance of the proposed scheme. We first compare the proposed scheme with the benchmarking method Naive under a fixed workload and varying energy supplies. Specifically, when the system consists of 5 servers and the workload is 200, the system energy supply is varied as 2×10^5, 2.3×10^5, 2.6×10^5, 2.9×10^5, 3.2×10^5, and 3.4×10^5. When the system consists of 10 servers and the workload is 600, the system energy supply is varied as 7.8×10^5, 8.0×10^5, 8.2×10^5, 8.4×10^5, 8.6×10^5, and 8.8×10^5. When the system consists of 15 servers and the workload is 2000, the system energy supply is varied as 1.2×10^6, 1.22×10^6, 1.24×10^6, 1.26×10^6, 1.28×10^6, and 1.30×10^6.

The performance characterized by $Metric_P$ of the proposed scheme and Naive method under a fixed workload and varying energy supplies are demonstrated in Fig. 5. It has been shown in the figure that the proposed scheme outperforms the Naive method. The performance of the proposed scheme can be up to 126.3%, 443.4%, and 280.6% better than that of the Naive method for 5-server system, 10-server system, and 15-server system, respectively. For example, when $M = 10$ and $E_{avl} = 8.8 \times 10^5$, the $Metric_P$ of the proposed scheme and Naive method are 21.838 and 4.019, thus the performance upgradation is $\frac{21.838 - 4.019}{4.019} \times 100\% = 443.4\%$.

We then compare the proposed scheme with the benchmarking method Naive under a fix energy supply and varying workloads. For

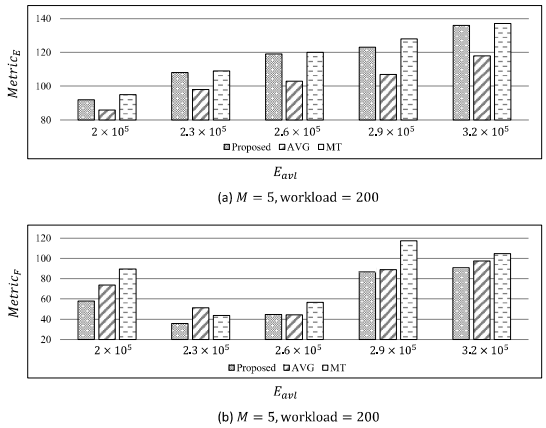


Figure 7: Compare the proposed scheme with benchmarking algorithms MT and AVG in efficiency and fairness.

5-server system, the energy supply E_{avl} is set to 2×10^5, and the workload is varied as 200, 220, 240, 260, 280, and 300. For 10-server system, the energy supply E_{avl} is set to 5×10^5, and the workload is varied as 600, 700, 800, 900, 1000, and 1100. For 15-server system, the energy supply E_{avl} is set to 1.2×10^6, and the workload is varied as 2000, 2200, 2400, 2600, 2800, and 3000.

Fig. 6 presents the performance $Metric_P$ of the proposed scheme and Naive method under a fixed energy supply and varying workloads. As can be seen in the figure, the proposed scheme has a better performance than the Naive method. In the simulated 5-server, 10-server, and 15-server system, the performance of the proposed scheme can be up to 127%, 200%, and 299.5% better than that of the Naive method. For instance, when $M = 15$ and workload is 2800, the $Metric_P$ of the proposed scheme and Naive method are 6.880 and 1.722, thus the performance upgradation is $\frac{6.880-1.722}{1.722} \times 100\% = 299.5\%$

3) Validate the Efficiency and Fairness: We compare the efficiency and fairness of the proposed scheme with that of two benchmarking methods MT and AVG under a fixed workload and varying energy supplies. As introduced in Section V-A, the efficiency is characterized by $Metric_E$ and holds for that the higher the $Metric_E$, the more efficient the scheme. The fairness is characterized by $Metric_F$ and holds for that the lower the $Metric_F$, the more fair the scheme. In this simulation, the server number M is set to 5, the workload is set to 200, and the energy supply is varied as 2×10^5, 2.3×10^5, 2.6×10^5, 2.9×10^5, and 3.2×10^5.

The efficiency and fairness of the proposed scheme and two benchmarking methods MT and AVG for the 5-server system under a fixed workload and varying energy supplies are demonstrated in Fig. 7. As shown in the figure, unlike the two methods MT and AVG that only concern one side of efficiency and fairness, the proposed scheme can achieve a balance between efficiency and fairness. This benefits from its consideration of both sides. The same conclusion can be drawn for the 10-server and 15-server system, whose results are not shown in the paper due to page limit.

VI. Conclusions

In this paper, we investigate the problem of throughput-aware energy allocation for heterogeneous cluster servers with renewable generation. The objectives of energy allocation are to achieve high system overall throughput and throughput of individual servers. To achieve the objectives, we take a first step towards using game theoretic approaches to shared renewable energy powered server clusters, by applying cooperative game framework and designing a heuristic algorithm that generates the Nash bargaining solution. Based on the generated Nash bargaining solution, an efficient and fair energy allocation scheme is derived. Extensive simulations are carried out to validate the uniqueness of the Nash bargaining solution and the effectiveness of our scheme. Simulation results show that our theoretic scheme can achieve a high throughput for both of the overall system and individual servers.

References

[1] https://www.greenpeace.de/sites/www.greenpeace.de/files/publications/clicking-clean-20151905.pdf
[2] "Offshore oil and gas supply," *Working Document of the National Petroleum Council*, 2011.
[3] "The changing geospatial landscape," *A Report of the National Geo-spatial Advisory Committee*, 2009.
[4] "How big data is changing astronomy (again)," *The Atlantic*, 2012.
[5] W. Cox, M. Pruett, T. Benson, S. Chiavacci, and F. Thompson III, "Development of camera technology for monitoring nests," *USGS Northern Prairie Wildlife Research Center*, 2012.
[6] C. Li, Y. Hu, L. Liu, J. Gu, M. Song, X. Liang, J. Yuan, and T. Li, "Towards sustainable in-situ server systems in the big data era," in *Proc. Int. Symp. Computer Architecture (ISCA)*, pp. 14-26, 2015.
[7] C. Li, W. Zhang, C. Cho, and T. Li, "SolarCore: Solar energy driven multi-core architecture power management," in *Proc. Int. Symp. High Performance Computer Architecture (HPCA)*, pp. 2015-216, 2011.
[8] N. Sharma, S. Barker, D. Irwin, and P. Shenoy, "Blink: Managing server clusters on intermittent power," *ACM SIGPLAN Notices*, pp. 185-198, 2011.
[9] Ł. Goiri, R. Beauchea, K. Le, T. Nguyen, M. Haque, J. Guitart, J. Torres, and R. Bianchini, "GreenSlot: scheduling energy consumption in green data centers," in *Proc. Int. Conf. High Performance Computing, Networking, Storage, and Analysis*, 2011.
[10] Ł. Goiri, W. Katsak, K. Le, T. Nguyen, and R. Bianchini, "Parasol and greenSwitch: managing datacenters powered by renewable energy," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 1, pp. 51-64, 2013.
[11] Ł. Goiri, K. Le, T. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "GreenHadoop: leveraging green energy in data-processing frameworks," in *Proc. ACM Euro. Conf. Computer Systems*, pp.57-70, 2012.
[12] C. Li, R. Zhou, and T. Li, "Enabling distributed generation powered sustainable high-performance data center," in *Proc. Int. Symp. High Performance Computer Architecture (HPCA)*, pp.35-46, 2013.
[13] H. Yaiche, R. Mazumdar, and C. Rosenberg, "A game theoretic framework for bandwidth allocation and pricing in broadband networks," *IEEE/ACM Trans. Networking*, vol. 8, no. 5, pp. 667-678, 2000.
[14] M. Osborne and A. Rubinstein, "A course in game theory," *MIT press*, 1994.
[15] J. Guo, F. Liu, D. Zeng, J. Lui, and H. Jin, "A cooperative game based allocation for sharing data center networks," in *Proc. Int. Conf. Computer Communications*, pp. 2139-2147, 2013.
[16] R. Subrata, A. Zomaya, and B. Landfeldt, "A cooperative game framework for QoS guided job allocation schemes in grids," *IEEE Trans. Computers*, vol. 57, no. 10, pp. 1413-1422, 2008.
[17] A. Mohsenian-Rad, V. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Trans. Smart Grid*, vol. 1, no. 3, pp. 320-331, 2010.
[18] C. Liu, K. Li, C. Xu, and K. Li, "Strategy configurations of multiple users competition for cloud service reservation," *IEEE Trans. Parallel and Distributed Systems*, 2015.
[19] Y. Ge, Y. Zhang, Q. Qiu, and Y. Lu, "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *Proc. Int. Symp. Low Power Electronics and Design*, pp. 279-284, 2012.
[20] X. Fan, W. Weber, and L. Barroso, "Power provisioning for a warehouse-sized computer," in *Proc. Int. Symp. Computer Architecture*, pp. 13-23, 2007.
[21] L. Gu, D. Zeng, A. Barnawi, S. Guo, and I. Stojmenovic, "Optimal task placement with QoS constraints in geo-distributed data centers using DVFS," *IEEE Trans. Computers*, vol. 64, no. 7, pp. 2049-2059, 2014.
[22] I. Marshall and C. Roadknight, "Linking cache performance to user behaviour," *Computer Networks and ISDN Systems*, vol. 30, no. 223, pp. 2123-2130, 1998.
[23] S. Gunduz and M. Ozsu, "A poisson model for user accesses to web pages," *Computer and Information Sciences - ISCIS 2003, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg*, vol. 2869, pp. 332-339, 2003.
[24] T. Robertazzi, "Computer networks and systems: queueing theory and performance evaluation," *Telecommunication networks and computer systems, Springer*, 2000.
[25] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Trans. Parallel and Distributed Systems*, pp. 2867-2876, 2014.
[26] Y. Lee and A. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions," *IEEE Trans. Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1374-1381, 2011.
[27] A. Muthoo, "Bargaining theory with applications," *Cambridge University Press*, 1999.
[28] J. Nash, "The bargaining problem," *Econometrica*, vol. 18, pp. 155-162, 1950.
[29] S. Boyd and L. Vandenberghe, "Convex Optimization," *Cambridge University Press*, 2004.
[30] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Real-time scheduling for energy harvesting sensor nodes," *Real-Time Syst. J.*, vol. 37, no. 3, pp. 233-260, 2007.
[31] J. Chen, T. Wei, and J. Liang, "State-aware dynamic frequency selection scheme for energy-harvesting real-time systems," *IEEE Trans. Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 8, pp. 1679-1692, 2013.