An Instability-Resilient Renewable Energy Allocation System for a Cloud Datacenter

Haiying Shen, *Senior Member, IEEE*, Haoyu Wang, Jiechao Gao, and Rajkumar Buyya, *Fellow, IEEE*

***Abstract*—Renewable energy supply is a promising solution for datacenters' increasing electricity monetary cost, energy consumption and harmful gas emissions. However, due to the instability of renewable energy, insufficient renewable energy supply may lead to the use of stored energy or brown energy. To handle this problem, in this paper, we propose an instability-resilient renewable energy allocation system. We define a job's service-level-objective (SLO) as the successful running probability by only using supplied renewable energy. The system allocates jobs with the same SLO level to the same physical machine (PM) group, and powers each PM group with renewable energy generators that have probability no less than its SLO to produce the amount no less than its energy demand. We use a deep learning technique to predict the probability of producing the amount no less than each value of each renewable energy source, and predict the energy demands of each PM area. We formulate an optimization problem to match renewable energy resources with different instabilities to different PM groups for supply, and use reinforcement learning method and linear programming method to solve it. We further propose an energy-driven computing resource assignment method, which adjusts the amount of computing resource of each job based on job deadline and failure probability in each PM group, and a failure prediction based energy saving method. Real trace driven experiments show that our methods achieve much lower SLO violations, total energy monetary cost and total carbon emission compared to other methods and the effectiveness of individual methods.**

***Index Terms*—Cloud datacenter, machine learning prediction, renewable energy scheduling.**

I. Introduction

OVER the past years, more and more Internet services (e.g., e-commerce, content distribution, gaming, and social networking) have been deployed over the cloud datacenters, which are reliable, elastic, and cost-effective. Consequently, the size and energy consumption of datacenters have been increasing significantly. As a result, datacenters' increasing electricity monetary cost, energy consumption and energy harmful gas emissions have become a severe problem to the society. Some of the world's largest datacenters require more than 100 megawatts (MW) of power capacity, which is enough to power around 80,000 U.S. households [1]. In 2020, the US datacenter industry consumed around 196 to 400 terawatt-hours (TWh), which is equivalent to 1%–2% of worldwide annual data center energy consumption [2]. A large amount of datacenters around the world are powered by electricity generated by brown energy such as fuel fossil, coal and oil. On a global level, datacenters contribute to 0.3% of all global CO2 emissions [3].

To solve this problem, governments in many countries began to set up laws and regulations to brown energy utilization. The government and the Environmental Protection Agency (EPA) will punish the datacenters with severe fines based on the carbon emission [4]. As a solution, cloud service providers start using renewable energy such as solar, wind and hydro to power the cloud datacenters. For example, Microsoft partnered with Swedish company Vattenfall to build and deliver a large-scale 24/7 renewable energy matching solution at the new datacenter region on an hourly basis to ensure that every megawatt hour (MWh) of energy consumed at the datacenter is matched with a MWh of renewable energy generation that was generated during the same hour of consumption [5].

In the future, a system with thousands of active consumers who own solar and wind energy generators and have an ability to sell the renewable energy is envisioned [6]. Consider the scenario with geo-distributed renewable energy sources (or generators), previous research methods focus on how to schedule jobs to the datacenters to use these renewable energy resources to minimize either energy monetary cost or carbon emission while satisfying time latency constraints of jobs [7], [8], [9], [10], [11], [12]. These methods predict the amount of generated renewable energy from each generator powering a specific datacenter, and the energy demands from jobs, and then schedule the jobs to different datacenters to achieve the goals.

However, renewable energy resources are featured by instability. For example, the amount of produced solar energy depends on solar irradiance, and the amount of produced wind energy depends on wind turbines; both factors depend on the time of a day, the season, the climate and so on. The energy amount instability brings about a new challenge since insufficient renewable energy supply may lead to the use of stored energy or brown energy, which increases power monetary cost and/or carbon emissions. Though the previous works attempt to more accurately predict the amount of produced renewable energy, due to the energy instability, sufficient renewable energy supply cannot be always guaranteed. Thus, it is important to study the problem about how

Manuscript received 24 July 2021; revised 24 December 2022; accepted 8 January 2023. Date of publication 25 January 2023; date of current version 31 January 2023. This research was supported in part by U.S. NSF under Grants NSF-2206522, NSF-1822965, NSF-1827674, NSF-1733596, and NSF-1724845, in part by Microsoft Research Faculty Fellowship under Grant 8300751, and in part by AWS Machine Learning Research Awards. Recommended for acceptance by A. Orgerie. *(Corresponding author: Haiying Shen.)*

Haiying Shen, Haoyu Wang, and Jiechao Gao are with the Department of Computer Science, University of Virginia, Charlottesville, VA 22903 USA (e-mail: hw8c@virginia.edu; hs6ms@virginia.edu; jg5ycn@virginia.edu).

Rajkumar Buyya is with the School of Computing and Information Systems, University of Melbourne, Parkville, VIC 3010, Australia (e-mail: rbuyya@unimelb.edu.au).

Digital Object Identifier 10.1109/TPDS.2023.3235957

to choose renewable energy generators to power a datacenter to mitigate the adverse effects due to overestimation of the amount of produced renewable resources from the generators. The adverse effects include the power monetary cost increase and/or harmful gas emission increase caused by using stored energy and brown energy.

This article aims to handle this problem. For this purpose, we define a job's service-level-objective (SLO) as the successful running probability by only using supplied renewable energy. To reduce the SLO violations due to overestimation of the amount renewable resources, we propose an instability-resilient renewable energy allocation system system. It allocates jobs with the same SLO level to the same physical machine (PM) group, and power the PM group with an SLO $= x\%$ and predicted energy demand of ykWh with the renewable energy generator that is predicted to generate no less than ykWh amount with no less than $x\%$ probability at each time slot in the next time period.

Our system runs after each time period and schedule which energy generator will supply energy to which PM area in the next time period in order to minimize the number of SLO violations (due to insufficient supplied renewable energy), total energy monetary cost and total carbon emission. It has the following steps:

- First, we use long short term memory (LSTM) deep learning model to predict the tail distribution of the amount of generated renewable energy of each energy source represented, i.e., the probability that it will generate no less than each certain amount of energy at each time slot of the next time period. Second, we use LSTM to predict the energy demand of each PM area.
- Third, we allocate renewable energy generators to the PM areas based on the aforementioned rule. That is, if an energy source's generated renewable energy amount is no less than the PM area energy demand, and the probability of producing that amount is no less than the SLO value of the PM areas, it can be a candidate to be assigned to the PM area. Specifically, we use a reinforcement learning (RL) method and a linear programming method to solve the aforementioned problem.

An early version of this work was presented in [13]. In this early work, we assumed that the resource supply prediction is correct. However, since many factors affect the amount of the generated renewable energy, the actual amount may vary from the predicted value. To handle this problem, in this article, we propose an energy-driven computing resource assignment method (ECRA), which adjusts the amount of computing resource of each job based on job deadline, SLO and failure probability. In addition, to avoid unnecessary energy consumption to avoid the adverse effect of insufficient energy supply, we propose a failure prediction based energy saving method (FPES). The additional contribution is summarized as below:

- *Energy-driven computing resource assignment (ECRA):* In order to further handle the renewable energy overage and shortage (caused by the supply instability) while avoiding SLO violation, ECRA reduces the computing resources of the jobs with loose deadlines and SLOs when an energy shortage happens and assign more computing resource to the jobs with strict deadlines and SLOs, and low failure probability when an energy overage happens.
- *Failure prediction based energy saving (FPES):* When a job fails, it restarts from the previous checkpoint. Then, the computing resources and energy resources used during the time from the previous checkpoint and the failure time are wasted. In order to avoid such resource and energy waste caused by job failures, FPES predicts each job's failure probability using bi-directional LSTM [14] and restarts the job right after the checkpoint before the failure if the failure is not right after a check point. This way, it reduces energy consumption for unnecessary job execution.

We conduct comprehensive real trace-driven experiments to compare our methods with other three methods in terms of SLO satisfactory ratio, total energy monetary cost and total carbon emission. The experimental results show that our methods can achieve much lower number of SLO violations, total energy monetary cost and total carbon emission compared to the other methods and the effectiveness of the individual methods. Specifically, for 100 servers in a month, RL uses around 60% less brown energy, generates around 7-8% higher uninterrupted PM area ratio and SLO satisfaction ratio, causes $150000 (19%) less monetary cost and 0.2Tons (9%) less carbon emission. In addition, ECRA reduces around 50% brown energy of RL, and FPES further can reduce around 50% brown energy. ECRA increases around 0.7% uninterrupted PM area ratio and around 0.2% SLO satisfaction ratio of RL. ECRA reduces $20000 (3%) monetary cost and 0.4Tons (20%) carbon emission of RL, and FPES further can reduce $40000 (6%) monetary cost and 0.2Tons (13%) carbon emission. We distributed our source code [15].

The rest of the article is organized as follows. Section II presents the related work. Section III presents the background and our research problem. Section IV presents the basic instability-resilient renewable energy allocation system. Sections V and VI present the energy-driven computing resource assignment method and the failure prediction based energy saving method, respectively. Section VII presents the performance evaluation of our system. Section VIII concludes the article with remarks on our future work.

II. Related Work

Energy-Efficient Resource Management: Reducing the number of running servers is a common approach to reduce energy consumption of a datacenter. Chen et al. [16] proposed algorithms to minimize the number of running servers via dynamically distributing workload to the servers, which saves up to 30% energy. Heller et al. [17] introduced ElasticTree, an energy manager with a focus on the datacenter network elements (links and switches). It monitors traffic conditions in the datacenter, and simply turns off the switches and links if they are not needed. Lin et al. [18] proposed an Energy-Efficient Adaptive File Replication System (EAFR). EAFR decreases the number of replicas for cold files without compromising their read efficiency, stores the cold files to servers and put these servers to the sleep mode to save energy. Dabbagh et al. [19] developed a framework for predicting future virtual machine requests and associated resource requirements. It puts unneeded machines into the sleep mode to reduce energy consumption. These methods focus on reducing the energy use while we focus on the renewable energy allocation. We can use these methods to further save renewable energy for a datacenter.

Renewable Energy Management: Given a number of geo-distributed datacenters, with each datacenter being powered by certain renewable energy sources, several methods have been proposed to reduce total energy monetary or carbon emission. The method in [10] aims to minimize monetary cost while giving higher priority to using renewable energy via rescheduling (or migrating) jobs between datacenters using RL based on neural network (NN) model. The method in [7] uses an integer linear programming method to allocate jobs to the different datacenters to minimize the carbon emissions of the datacenters by using renewable energy while satisfying a few requirements. This method uses a pattern-based method to predict the amount of generated renewable energy in each energy source. Liu et al. [8] proposed an integrated workload management system for one datacenter. Since different renewable energy sources have dynamic energy generation and price through time, the system tries to minimize the monetary cost by scheduling jobs to different time slots while satisfying job processing time constraint and using solar energy as much as possible since it is easy to predict. It uses the k-nearest neighbor (k-NN) method to predict renewable energy and energy demand of each node based on historical data. De Courchelle et al. [11] proposed a job scheduling method for a datacenter aiming to use renewable energy as much as possible. Gu et al. [9] aim to minimize the brown energy usage via task allocation and renewable energy scheduling in edge computing using a mixed integer linear programming. Xu et al. [20] proposed a job reallocation based method, which adjusts the number of jobs among multiple cloud datacenters, aiming to reduce carbon emission caused by brown energy and maximize the renewable energy (solar energy) usage as much as possible. Liu et al. [21] proposed a renewable energy matching system, aiming to achieve lower monetary cost and carbon emission using mixed linear programming. Nayak et al. [22] proposed user requests scheduling system, which assigns different user requests to different datacenter, aiming to reduce the user requests' completion time and also renewable energy cost.

Due to renewable energy instability, the above works may overestimate the amount of produced renewable resource from a generator. Different from the above works, our work handles how to choose renewable energy generators to power a datacenter to mitigate the adverse effect on the datacenter jobs from overestimation of the amount of produced renewable resources from the generators. It can complement other renewable energy management methods to reduce the number of SLO violations due to insufficient renewable energy supply.

SLO-aware Resource Scheduling: A significant amount of previous research focuses on achieving job SLO guarantee, where the SLO usually reflects the job latency. Wen et al. [23] proposed StepConf, SLO-aware dynamic resource configuration for serverless function workflows. Safaryan et al. [24] designed a tool called SLAM to solve the issue of minimizing cost and meeting SLO requirements for an application consisting of many FaaS functions. SLAM determines the optimal memory configuration for the given serverless application. Shukla et al. [25] presented analysis of the impact of cluster heterogeneity on the achieved server utilization and energy footprint to meet the SLO of latency-critical services. Zhang et al. [26] proposes MArk (Model Ark), a general-purpose machine learning (ML) inference serving system, to tackle the dual challenge of response-time SLO compliance and serving cost effectiveness. Shukla et al. [27] proposed a user-centric End-to-end Service Level Objective (ESLO) that guarantees stricter bounds on end-to-end delay and thereby achieving a higher QoE. The authors showed how the variability in the external network delay can be both addressed and leveraged to meet the ESLO and improve server utilization, and proposed ESLO-aware infrastructure. Alsadie et al. [28] presented a dynamic threshold-based fuzzy approach (DTFA) to detect overloaded and underutilized PMs and the Lowest Interdependence Factor Exponent Multiple Resources predictive (LIFE-MP) approach for placing virtual machines (VMs) on PMs. Ramesh et al. [29] used ML model to predict the resource utilization of VM and PM for load balancing to satisfy SLOs in the cloud. Cortez et al. [30] characterized Azure's VM workload to demonstrate how the VM characteristics can be utilized for better resource management. Hua et al. [31] used various forms of LSTM for time series forecasting for resource management. Chen et al. [32] proposed a deep Learning based Prediction Algorithm (L-PAW) to achieve adaptive and accurate prediction of workloads that are highly variable, thereby resulting in lower resource wastage and lower SLO violations. Kumar et al. [33] used LSTM based workload prediction model for efficient resource scaling and energy consumption. Ding et al. [34] used predicted resource utilization and Performance-to-power Ratio (PPR) of heterogeneous hosts in order to ensure the balance of workload and energy. The methods in [35], [36], [37] use an RL or Markov decision process (MDP) algorithm for resource management and VM placement to minimize energy consumption and/or achieve load balance.

Different from these SLO-aware methods, our defined SLO is for successful job execution using the renewable energy. We use the previous methods to guarantee job deadline SLO, and our method is orthogonal to these previous SLO-aware methods.

III. Background and Research Problem

Different renewable energy resources are influenced by different natural features [38], [39]. Solar energy is influenced by solar irradiance, and wind energy is influenced by wind speed. The renewable energy resources are featured by instability due to the environment and climate change. For example, in summer sunny days, solar can generate more stable energy resource than wind, and it can generate more energy at the daytime than that at night. In winter cloudy days, wind can generate more stable energy resource than solar. This instability feature brings a challenge when we use renewable energies as energy supply in local datacenters. To handle this problem, previous research attempts to

TABLE I
NOTATIONS

Notation	Description
G_k	the k^{th} renewable energy source (or generator)
$g_{G_k,t}$	predicted energy generation amount of G_k at time t
P_{G_k,t_n}	tail distribution of G_k at t_n
$p(g^i_{G_k,t_n})$	probability that the amount of the generated energy is $\geq g^i_{G_k,t_n}$
P_{G_k}	tail distribution of G_k at each time in the next time period
$\mathcal{P}_G$	the vector of P_{G_k} of all generators
$\mathcal{D}_G$	distance between each generator and the datacenter
c_{G_k,t_n}	energy price of generator G_k at time slot t_n
$\mathcal{C}_G$	unit price of each generator at each time
M_j	the j^{th} PM area
E_{M_j,t_n}	predicted energy demand of PM area M_j at time slot t_n
E_{M_j}	E_{M_j,t_n} at each time slot in the next time period
$\mathcal{E}_M$	the vector of E_{M_j} for all PM areas
$\mathcal{L}_M$	the vector of the SLO level of each PM area
$C_{j,k,t}$	monetary cost for $g_{G_k,t}$ energy from G_k at t to be used by M_j
$W_{j,k,t}$	total carbon emission for $g_{G_k,t}$ amount of energy from G_k at t
$L_{M_j,t}$	the SLO of PM area M_j
V_{A_i}	the number of SLO violations for action A_i
θ	error budget
r	a job's remaining time

increase the prediction accuracy of the produced energy amount, e.g., by using pattern match or machine learning methods, but it is hard to guarantee 100% prediction accuracy due to the instability (as verified in our experiments in Section VII). In addition, prediction at a higher frequency is needed due to the instability, which generates high computation overhead. Therefore, renewable energy sources sometimes may not generate enough energy as predicted to power the datacenters. In this case, a datacenter can use brown energy or stored energy from the energy grid. However, it degrades the performance of achieving the goals of minimizing the total energy monetary cost and total carbon emission. We assume that the cost of stored renewable energy is higher than that of directly using the renewable energy since there is an additional cost for energy storage. Thus, we define SLO as the successful running probability by only using supplied renewable energy and aim to minimize the number of SLO violations.

We denote G_k as the k^{th} renewable energy source (or generator), $g_{G_k,t}$ as the predicted energy generation amount of renewable energy source G_k at time t. Complementary cumulative distribution function or simply tail distribution represents the probability distribution that the amount of the generated renewable energy is no less than each certain amount. We use P_{G_k,t_n} to denote the tail distribution of renewable energy source G_k at t_n time.

$$P_{G_k,t_n} = \{< g^1_{G_k,t_n}, p(g^1_{G_k,t_n}) >, < g^2_{G_k,t_n}, p(g^2_{G_k,t_n}) >, \ldots\} \tag{1}$$

where $p(g^i_{G_k,t_n})$ means the probability that the amount of the generated energy is no less than $g^i_{G_k,t_n}$. Table I shows the notations used in this article.

Note that different jobs have different SLOs. To avoid a job's SLO violation due to insufficient supplied renewable energy, we can assign the job the renewable energy generator that has probability no less than the SLO to produce the energy amount no less than the job's energy demand. A problem here is how to conduct such a mapping since energy is supplied to PMs rather than jobs. To handle this problem, we propose to divide PMs to PM areas, and each PM area hosts jobs with one SLO value. Then, we supply each PM area with renewable energy sources to avoid SLO violations. Note that when we allocate jobs to the PM areas, we need to consider the constraint of PM computing resource capacity and try to consolidate jobs to as few PMs as possible to save energy. We can reply on previous methods (e.g., [40]) for these purposes, which are out of the scope of this article. Avoiding job deadline SLO violations due to insufficient computing resources has been handled in previous research. This is not the focus of this article and we directly use the previous methods for this purpose in this work. In this article, we focus on avoid violating our redefined SLO due to insufficient supplied renewable energy. Specifically, our renewable energy resource assignment **problem** is as follows:

Given a datacenter and many geo-distributed renewable energy sources (or generators) that the datacenter can use, how the renewable energy sources should be mapped to the PM areas in order to minimize the number of SLO violations (due to insufficient renewable energy), total energy monetary cost and total carbon emission?

To handle this problem, we propose an instability-resilient renewable energy allocation system that conducts such mapping periodically (e.g., every hour) (Section IV). To enhance this system, we further propose the energy-driven computing resource assignment method (Section V) and the failure prediction based energy saving method (Section VI). In this article, we assume that a PM area has enough computing resources for the jobs. We present each of the system components in the following.

IV. Instability-Resilient Renewable Energy Allocation System

The instability-resilient renewable energy allocation system conducts the mapping between renewable energy generators and PM areas to solve the above assignment problem. Therefore, it incorporates the following components:

1) It predicts the tail distribution of each renewable energy source and the energy demand in each PM area at each time slot in the next time period (Section IV-A).
2) Based on the predicted renewable energy generation and predicted energy demand, it assigns renewable energy sources to PM areas using RL-based method and linear programming method (Section IV-B).

A. Prediction for Renewable Energy Generation

We assume that our system conducts the mapping between renewable energy sources and PM areas every one hour, though it can be any time length. For each renewable energy source, it predicts the amount of generated energy every time slot t_i (e.g., 5 minutes) within the next hour, and for each PM area, it predicts the amount of energy demand every time slot within the next hour. Then, it conducts the mapping to make sure that the renewable energy sources mapped to a PM area will satisfy its energy demand at each time slot t_i within the next hour. That is, if a PM area demands ykWh at t_i and its SLO level is x%, the matched renewable energy source must produce no less than ykWh with probability no less than x% at time t_i for each time slot within the next hour. Such a mapping strategy is to avoid SLO violation due to insufficient renewable energy supply.

Therefore, for each renewable energy source, we need to predict the tail distribution of each renewable energy source, e.g., the probability of generated energy amount no less than 1 kWh is 95%, no less than 2 kWh is 92%, and so on. To do this, we use the long short term memory (LSTM) deep learning model [41], [42] since it is effective in handling time series data and can observe the correlation between different time slots. The inputs of LSTM include a set of time sequence data, which records the historical amount of generated energy of a renewable energy source, and the factors affecting the amount of the renewable energy resource (e.g., solar irradiance for solar energy, wind speed for wind energy), and the output of LSTM is the tail distribution at each t_i in the next hour. We explained the tail distribution of energy resource G_k at time t_n (P_{G_k,t_n}) in the above. We use P_{G_k} to denote the tail distribution of renewable energy generator G_k at each time in the next time period.

$$P_{G_k} = \{P_{G_k,t_1}, P_{G_k,t_2}, \ldots, P_{G_k,t_n}, \ldots, P_{G_k,t_N}\}, \quad (2)$$

where N is the number of time slots in time period T.

We need to predict the amount of energy consumption in each PM area at each t_n in the next hour. For this purpose, we also use the LSTM deep learning technique based on the historical energy consumption time-series data of a PM area.

B. Mapping Renewable Energy Sources and PM Areas

After we predict the tail distribution of the amount of generated energy in each renewable energy source, and the energy demand of each PM area, we need to map the renewable energy sources to the PM areas for energy supply to solve the problem in Section III. In this article, we use two methods to solve this problem. First, we formulate this problem as a Markov Decision Process (MDP), and use a reinforcement learning (RL) method based on Deep Q-Network (DQN) [43] to solve the MDP problem [44], [45], [46]. Second, we formulate this problem as an optimization problem and then use integer linear programming approach to solve it. We present each method in the following.

1) RL-Based Method: We first formulate this problem as an MDP, denoted by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is the state, $\mathcal{A}$ is the action, $\mathcal{P}$ is the probability between each two states and $\mathcal{R}$ is the reward. Below, we introduce these elements for our problem.

a) State space: The state space $\mathcal{S}$ is defined as the input of the RL model, and it consists of the information of renewable energy sources and PM areas in a datacenter. The information of renewable energy sources includes the tail distribution ($\mathcal{P}_G$), distance with the datacenter ($\mathcal{D}_G$), unit price ($\mathcal{C}_G$). We denote the renewable energy generators by:

$$\mathcal{G} = \{G_1, G_2, \ldots G_k, \ldots G_K\} \quad (3)$$

where G_k means the k^{th} renewable energy generator and K means the total number of renewable energy generators. The tail distribution at each time for the next time period (e.g., one hour) of all energy sources is denoted by $\mathcal{P}_G$.

$$\mathcal{P}_G = \{P_{G_1}, P_{G_2}, \ldots, P_{G_k}, \ldots, P_{G_K}\}. \quad (4)$$

The distance $\mathcal{D}_G$ vector indicates the distance between each energy source and the datacenter:

$$\mathcal{D}_G = D_{G_1}, D_{G_2}, \ldots D_{G_k}, \ldots D_{G_K} \quad (5)$$

The unit price vector indicates the unit price of each energy source at each time slot:

$$\mathcal{C}_G = C_{G_1}, C_{G_2}, \ldots, C_{G_k}, \ldots, C_{G_K}, \quad \text{where} \quad (6)$$

$$C_{G_k} = \{c_{G_k,t_1}, c_{G_k,t_2}, \ldots, c_{G_k,t_n}, \ldots c_{G_k,t_N}\}, \quad (7)$$

where c_{G_k,t_n} denotes the energy price of generator G_k at time slot t_n. As in [10], we assume that the unit price at each time slot of an energy resource is pre-known. If it is not pre-known, we can also use LSTM for the price prediction based on the factors influencing the price.

The features of the PM areas in the cloud datacenter include the predicted energy demand of each PM area at each time slot in the next time period and the SLO of each PM area. We use J to denote the total number of PM areas in the datacenter, and use M_j to denote the j^{th} PM area. We use E_{M_j} to denote the predicted energy demand of PM area M_j at each time slot in the next time period, and use $\mathcal{E}_M$ to denote the vector of E_{M_j} for all PM areas. Therefore,

$$\mathcal{E}_M = \left\{E_{M_1}, E_{M_2}, \ldots, E_{M_j}, \ldots E_{M_J}\right\}, \quad \text{where} \quad (8)$$

$$E_{M_j} = \left\{E_{M_j,t_1}, E_{M_j,t_2}, \ldots, E_{M_j,t_n}, \ldots, E_{M_j,t_N}\right\} \quad (9)$$

where E_{M_j,t_n} denotes the predicted energy demand of PM area M_j at time slot t_n. We use $\mathcal{L}_M$ to denote the vector of the SLO level of each PM area in the cloud datacenter.

$$\mathcal{L}_M = \left\{L_{M_1}, L_{M_2}, \ldots L_{M_j}, \ldots L_{M_J}\right\} \quad (10)$$

where L_{M_j} is the SLO of the j^{th} PM area. As a result, the whole state space can be defined as follow:

$$\mathcal{S} = \{S = (\mathcal{P}_G, \mathcal{D}_G, \mathcal{C}_G, \mathcal{E}_M, \mathcal{L}_M)\}. \quad (11)$$

These features are selected because they are needed to determine whether a PM area's energy demand can be satisfied, the energy monetary cost and carbon emission, which are our energy allocation goals. It is important to select correct features for decision making since more features will increase the state space, which increases the training time of the RL model.

b) Action space: The action space is defined as the assignment plans to assign renewable energy sources to the PM areas as energy supply. We use $a_{k,j}$ to denote a binary variable; $a_{k,j} = 1$ means that renewable energy source G_k is assigned to PM area M_j, and $a_{k,j} = 0$ means otherwise. Action space $\mathcal{A}$ is expressed in the following:

$$\mathcal{A} = \left\{\left(A_1, A_2, \ldots A_i, \ldots A_{K(J+1)}\right) | A_i = (a_{G_1,M_1}, a_{G_1,M_2}, \ldots a_{k,j}, \ldots a_{G_K,M_J})\right\} \quad (12)$$

$$i \in \{1, 2, \ldots K(J+1)\}, k \in \{1, 2, \ldots K\}, j \in \{1, 2, \ldots J\} \quad (13)$$

where A_i is the i^{th} action, $A_{K(J+1)}$ means that there are total $K(J+1)$ actions in the action space because each renewable energy source has the chance to be assigned to each PM area.

c) Probability: When a decision about which energy source is assigned to which PM area is made, the state is changed with certainty, so the probability between states is always 1.

d) Reward: Based on problem in Section III, we consider the following factors in reward function.

Monetary cost: The unit price of energy source G_k at time t is denoted by $c_{G_k,t}$. The monetary cost for purchasing $g_{G_k,t}$ amount of energy from energy source G_k at time t to be used by the j^{th} PM area (denoted by $C_{j,k,t}$) is calculated:

$$C_{j,k,t} = c_{G_k,t} \cdot g_{G_k,t} \tag{14}$$

Carbon emission: The amount of carbon emission per kWh of energy source G_k at time t is denoted by $w_{k,t}$, The total carbon emission for $g_{G_k,t}$ amount of energy from energy source G_k at time t is calculated by

$$W_{j,k,t} = w_{k,t} \cdot g_{G_k,t} \tag{15}$$

SLO violations: To avoid SLO violations, when we map single source G_k to the j^{th} PM area, we need to ensure that each PM area is powered by renewable energy generators that have probability no less than its SLO to produce the amount no less than its energy demand. That is, $b_{G_k,t} = g_{G_k,t} - g_{G_k,t} \cdot \epsilon \cdot D_{G_k} \geq E_{M_j,t}$ and $p(g_{G_k,t}) \geq L_{M_j,t}$ for each time slot in the next time period, in which $g_{G_k,t} \cdot \epsilon \cdot D_{G_k}$ is the energy loss in energy transmission from the source to the PM area, ϵ is the loss rate, and $L_{M_j,t}$ is the SLO of PM area M_j. This can be easily extended to the case when multiple energy sources are mapped to one PM area to meet its energy demand. If either of the above conditions is not satisfied, the running jobs in the PM area may experience SLO violations. We use V_{A_i} to denote the number of SLO violations for action A_i; that is, the number of jobs running in the PM areas, where either of the above conditions is not satisfied. Our RL-based method is distinguishing in that it can directly use these conditions to measure a variable in the reward function for an action rather than collecting the resulting variable value by taking many actions in practice, which reduces training time.

Based on the aforementioned problem, we define the reward $\mathcal{R}$ for action A_i by the following equation:

$$\mathcal{R} = \frac{1}{\sum_{t\in T}\sum_{j\in J}\sum_{k\in K}(C_{j,k,t} + W_{j,k,t}) + V_{A_i}} \tag{16}$$

Our principle of the reward function is setting a higher reward for reducing more SLO violations, total energy monetary cost and total carbon emission.

e) Reinforcement learning training: To collect the training data of DQN, the datacenter initially can use the optimization solution from our linear programming method. We can also select the action randomly and calculate the reward from the selected action offline. The data is used for the DQN training to train the DQN network. The RL agent iteratively makes the decisions and updates the network parameters, which is the training process of DQN. After the DQN is trained, we deploy it in a centralized server where the trained RL agent runs and generates the mapping plan between the energy sources and PM areas periodically.

2) Optimization Problem Based Method: Given the features in renewable energy resources and PM areas, represented by the state in Formula (11), we formulate the renewable energy resource assignment problem as follow:

$$\text{Min} \sum_{t\in T}\sum_{j\in J}\sum_{k\in K} a_{k,j} \cdot (C_{j,k,t} + W_{j,k,t}) \tag{17}$$

$$\text{Subject to:} \sum_{k\in K} a_{k,j} \geq 1, \forall j \in J \tag{18}$$

$$\sum_{k\in K} a_{k,j} \cdot b_{G_k,t} \geq E_{M_j,t} \;\&\; a_{k,j} \cdot p(g_{G_k,t}) \geq L_{M_j,t}, \quad \forall t \in T, \forall k \in K, \forall j \in J \tag{19}$$

Equation (17) aims to minimize the total energy monetary cost and total carbon emission. Equation (18) ensures that a PM area must have no less than one energy sources to supply energy. Equation (19) aims to avoid SLO violations due to insufficient renewable energy supply in each PM area. Since in our scenario, the number of renewable energy sources and the number of PM areas are integers, the elements decision variables are integers too. Thus, our problem can be transformed into an optimization problem solved by integer linear programming method [47].

V. Energy-Driven Computing Resource Assignment (ECRA)

Since many factors affect the amount of the generated renewable energy, the actual amount may vary from the predicted value. For example, the wind speed may change dramatically in a short time, which changes the amount of generated wind energy [48]. Under the renewable energy supply variance, when a PM area's received renewable energy supply is smaller than its energy demand, the SLOs of its running jobs could be violated. When its received renewable energy supply is larger than its energy demand, it could store or sell the extra energy and use up the extra energy by providing more computing resources to the jobs to expedite their execution. In this article, we propose the ECRA method for how to decrease energy use for the energy shortage case or use up the extra energy for the energy overage case by adjusting the computing resources allocated to jobs while still meeting their SLOs and deadline requirements as much as possible.

The energy consumption is determined by the consumption of computing resources including CPU, memory and bandwidth and it can be calculated based on CPU [49]. If a job is offered with less amount of CPU resource, its job completion time will be increased. On the other hand, if a job is offered with more amount of CPU resource, its job completion time will be decreased and then its deadline requirement is more likely to be met. Also, if a job has a loose SLO, it can tolerate more energy shortages, so we can reduce its computing resource, which lengthens its completion time. On the other hand, if a job has a strict SLO, we should provide it more computing resource to make it complete soon to avoid future energy shortages. By leveraging these, when an energy shortage happens, ECRA reduces the computing resources of the jobs based on their strictness of deadlines and SLOs, when an energy overage happens, ECRA assigns more computing resource to the jobs based on their strictness of deadlines and SLOs and failure probability, in order to increase the probability that a job's deadline and SLO will be met. The job remaining time can be estimated directly according to [50].

We use the time difference between a job's deadline and its estimated job remaining time to represent its urgent value (denoted by u). For example, job a has 8 minutes estimated job remaining time and its deadline is in 20 minutes. Then, its urgent value is 20-8=12 minutes. Job b has 20 minutes estimated job remaining time and its deadline is 30 minutes. Then, its urgent value is 30-20=10 minutes. The urgent value represents the time a job can pause and resume in order to complete by its deadline. A lower urgent value means higher urgency. When an energy shortage happens, ECRA should reduce the computing resource assigned to job a first to make it completes by its deadline due to two reasons. First, since job a is less urgent compared with job b so that job deadline requirements have lower probability to be violated in an energy shortage. Second, since job a has a larger urgent value, the resource amount reduced to make it completes on its deadline is larger than that of job b. Then, we can limit the number of jobs that need to cut resource amount in order to solve the energy shortage issue. When an energy overage happens, ECRA should give higher priority to job b to increase computing resources since job b is more urgent compared with job a so that job deadline requirements have lower probability to be violated and the extra energy can be utilized.

When the SLO of a job equals $x\%$ (e.g., 90%), it is allowed to experience energy shortage with probability (1-$x\%$) (e.g., 10%) during its execution. Here, we use the concept of error budget for SLOs in Google's Site Reliability Engineering (SRE) [51]. A job with $x\%$ SLO has $\theta = (1 - \mathrm{x}\%)$ error budget. Since different jobs have different remaining times, solely considering the error budget is not fair to all the jobs. Thus, we introduce the concept of *error budget per time unit*: $\bar{\theta} = \theta / r$, where r is the job remaining time. A job with a lower $\bar{\theta}$ should have a higher priority to avoid SLO violations (i.e, receiving more computing resources to complete earlier to avoid shortage) and vice versa. Therefore, when an energy shortage occurs, we should cut computing resources from a job with a higher $\bar{\theta}$ since it is allowed to experience more shortages and vice versa. When there is an energy overage, we should add computing resources to a job with a lower $\bar{\theta}$, so it will experience less shortages.

Therefore, to jointly consider the job deadline and SLO, we use metric $u \cdot \bar{\theta}$. When energy shortage occurs, ECRA reduces the computing resources of the jobs with higher $u \cdot \bar{\theta}$ first, and when energy overage occurs, ECRA increases the computing resources of the jobs with lower $u \cdot \bar{\theta}$ first. In addition, in a datacenter, a job can be failed caused by hardware failure, resource competition or software bugs [52]. When a job fails, it will restart from its previous checkpoint. To save the energy resource, only when a job's predicted failure probability is less than a threshold (γ), ECRA adds computing resources to it. How to predict a job's failure probability will be presented in Section VI.

Algorithm 1. Dynamically Energy-Driven Computing Resource Assignment (ECRA) Algorithm Executed by a PM Area Periodically.

```
1  Monitor the energy supply and the actual energy usage;
2  for each PM area do
3      if renewable energy supply < energy usage then
4          for each running job do
5              Estimate remaining time and resource usage;
6          Sort jobs in descending order of $u \cdot \bar{\theta}$;
7          while reduced energy sum < energy shortage do
8              Reduce computing resource of the top job to make it
               complete by its deadline;
9      if renewable energy supply > the energy usage then
10         for each running job do
11             Estimate remaining time and resource usage;
12         Sort jobs in ascending order of $u \cdot \bar{\theta}$;
13         while increased energy sum < energy overage do
14             Increase computing resource of the top job if its failure
               probability is less than $\gamma$;
```

Algorithm 1 shows the pseudocode of the ECRA algorithm. ECRA monitors the energy supply and the actual energy usage of each PM area (line 1). It collects the data periodically (e.g., very 10 minutes). For each PM area, when the renewable energy supply is less than the energy usage (lines 2–3), ECRA first estimates the job remaining time for each job (lines 4–5). It then sorts the running jobs in descending order based on the $u \cdot \theta$ values (line 6). Next, it picks the job on the top one by one and reduces its computing resource assigned to it so that the job can complete on the deadline. This process repeats until the sum of the picked jobs' reduced consumed energy equals the energy shortage value (lines 7–8). For example, if job a (introduced above) is selected to reduce computing resource, the amount of computing resource is set to make the remaining time as 20, which equals its deadline. As a result, the supplied energy is enough to satisfy the jobs' deadline requirements and SLOs. It is possible that after ECRA reduces the computing resources of all the jobs, the amount of supplied renewable energy still cannot satisfy the energy demand of the PM area. In this case, the PM area resorts to the stored energy or brown energy.

When the renewable energy supply is larger than the energy usage, ECRA first estimates the job remaining running time for each job (lines 9–11). It then sorts the running jobs in ascending order based on the $u \cdot \theta$ value (line 12). Next, it picks the job on the top one by one and increases the computing resource assigned to it so that the job can complete earlier by a certain percentage (e.g., 10%) of its deadline. This process repeats until the sum of the picked jobs' increased consumed energy equals the energy overage value (lines 13–14). In this way, the extra supplied energy is used to expedite the execution of each job by a certain ratio of its deadline.

VI. Failure Prediction Based Energy Saving (FPES)

As mentioned above, a job can be failed caused by hardware failure, resource competition or software bugs. In the previous methods, once a job fails, this job is restarted from the checkpoint [53]. However, we notice that such an approach could waste energy resource. As shown in Fig. 1, one job needs 6 time unites to finish including 3 units for failed job running and 3 units for the successful job running after the failure occurs at

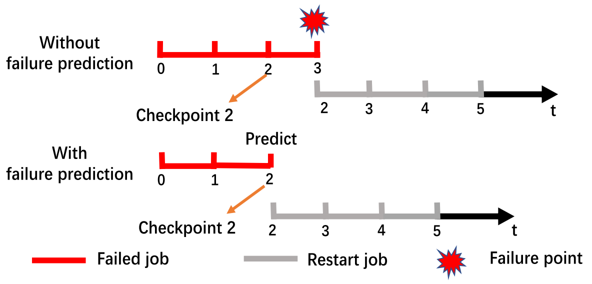


Fig. 1. Failure prediction energy saving example.

Algorithm 2. Failure Prediction Based Energy Saving (FPES) Algorithm Executed by a PM Area Periodically.

```
1 Collect job's running information for failure probability prediction;
2 for each job do
3     Predict failure probability;
4     if failure probability $> \gamma$ then
5         if failure is not right after checkpoint then
6             Restart this job from checkpoint;
7     Continue
```

time point 3 and it restarts from the checkpoint at time point 2. Thus, the energy for the original job execution from time point 2 to time point 3 is wasted. To deal with this problem, we propose the FPES method. Using FPES, after 2 time units running, a failure prediction method predicts that the job will be failed and the job restarts from checkpoint 2 in advance. Then, this job uses 5 time units to finish, which saves 1 time unit for job running. If the job failure occurs right after checkpoint 3 at time point 3, the job doesn't have to be restarted since the job running from time point 2 to time point 3 is still useful. Since longer job completion time means more energy usage, so such failure prediction can help greatly reduce energy usage. Note FPES is novel in that it saves the energy resource and job computing time of the previous checkpointing methods.

It was indicated that 41% of the total jobs suffer job failure and 35.8% [14] of the total jobs suffer the failure more than once in Google datacenter from Google cluster trace [54]. The failure prediction accuracy can be as high as 93% [14]. The failure can be predicted 5 minutes before the failure occurrence and then it can save around 10% energy for a job with 50 minutes job latency. In addition, for some jobs with longer job completion time, the failure may occur several times within this job's lifetime. For example, suppose one job's job completion time is 600 minutes and the failure happens 6 times. If we can predict the failure 10 minutes in advance, we can save around 60 minutes in total, resulting in 10% energy save. Then, for a datacenter consumed 4×10^5 kwh per year [55], failure prediction can save 1.6×10^4 kwh $\approx 4 \times 10^5$ kwh $\times 41\% \times 93\% \times 10\%$ (4% of the total energy consumption) per year. Since the failure prediction accuracy is high, this approach still can save a lot of energy consumption.

We now present the details of FPES. It also runs periodically (e.g., every 10 minutes) but FPES and ECRA run separately. FPES first predicts the failure probability of each running job using multi-layer Bidirectional LSTM [14]. FPES uses a threshold for the predicted failure probability (γ) to determine whether a job has a high failure probability. If the predicted failure probability of one job is larger than γ, FPES then checks whether the failure time point is within a small time range after a checkpoint. If not, the job will be restarted. The small time range is determined so that not restarting the job will not waste long job running time or energy consumption caused by failure. As the failure prediction model, the Bidirectional LSTM considers job features as input, such as resource utilization including CPU percentage, memory usage, bandwidth usage), the job re-submission status (new or resubmitted) and the job's waiting time. The multi-layer structure in this model can handle multiple input features for higher prediction accuracy. Algorithm 2 shows the pseudocode of the FPES algorithm. FPES collects the running job's information for prediction (line 1). For each job, FPES first predicts failure probability (lines 2–3). If the predicted failure probability is larger than the threshold γ and the failure time point is not within a small time range after a checkpoint (lines 4–5), FPES restarts this job right now (line 6). Otherwise, the job continues to run (line 7). In this way, the restarted jobs have high probabilities to complete earlier and save energy for job running.

VII. Performance Evaluation

A. Experiment Settings

In our experiment, we use the following real world datasets. All the datasets are from May 1, 2011 to May 30, 2011.

1) Datacenter and job workload: The Google cluster trace [54] records resource utilization of CPU and memory usage of each job in about 12.5 thousand PMs. For each job, we assign it an SLO value randomly chosen from the rage of [90%, 100%) [56]. Then we transfer the jobs with the same SLO value into the same PM area. The number of PM areas is determined by the SLO value. We vary the number of PM areas from 10 to 100 by controlling the division interval of the SLO range; 10 PM areas mean that the SLO range is divided by an interval of 1%, and 100 PM areas mean that the SLO range is divided by an interval of 0.1%. Unless otherwise specified, the number of PM areas is 100, and the mapping time period is one hour.

2) Renewable energy resources: In our experiments, We choose solar and wind as the renewable energy resources. We assume 500 renewable energy generators at the Virginia State (VA); half are solar energy generators and half are wind energy generators. To set the amount of energy produced by each generator at each time slot, we calculate the amount according to the methods in [57] and [58] as follows.

The amount of solar energy that can be generated in a time slot is calculated by:

$$E^{solar}(t) = \alpha \cdot A^{solar} s(t) \cdot \Delta t \tag{20}$$

where α is the ratio of how much solar energy can be transferred in to electricity, A^{solar} is the total active irradiation area of the solar panels, $s(t)$ is the solar irradiance, which means the energy per unit area (watt per square metre, W/m2), and Δt is the length of a time slot. The amount of wind energy that can be generated in a time slot is calculated by:

$$E^{wind}(t) = \beta \cdot \left(\frac{1}{2}\right) A^{wind} \rho^{air} v^3(t) \cdot \Delta t \qquad (21)$$

where β is the ratio of how much wind energy can be transferred in to electricity, A^{wind} is the total rotor area of all wind turbines, ρ^{air} is the air density, and $v(t)$ is the wind speed.

We obtained the datasets about regional solar irradiance ($s(t)$) and wind speeds ($v(t)$) at the VA from the National Renewable Energy Laboratory (NREL) [59], [60]. The data was recorded daily per hour, and we assume that the value keeps similar in different time slots in an hour. ρ^{air} is set to 1.29 Kg/m^3 and this value usually does not change in normal environment. For other parameters, we use the parameter settings in [57] and [58] for our renewable generator model. For each energy generator, the conversion efficiency ratio (α and β) is set to a value randomly chosen from [20%, 30%]. For each solar energy generator, A^{solar} is set to a value randomly chosen from [10000, 15000] m^2. For each wind energy generator, A^{wind} is set to a value randomly chosen from [20000, 25000] m^2.

3) Carbon emission rate (gCO_2e/kWh): Is to measure the amount of carbon emissions from the energy use. According to [61], coal has 968 carbon emission rate, wind has 22.5 carbon emission rate, solar has 53 carbon emission rate. Since coal is the most widely used brown energy for electricity, we use coal as our brown energy in our experiments.

4) Electricity price: The electricity price is obtained from the websites of Energy Information Administration [62] and the Switch [63], which contain the price of brown energy and renewable energy electricity price respectively at each hour. The electricity price varies from each hour as well. In general, the price for the solar energy is in the rage of [250, 350] USD/MWh, that of the wind energy is in the rage of [130, 220] USD/MWh, and that of the brown energy is in the rage of [50, 150] USD/MWh.

For increasing the computing resource assigned to jobs in ECRA, we set the percentage of the deadline to be reduced to 10%. We set $\gamma = 0.85$ and 30 seconds for the time range used in FPES. In addition, FPES runs every 5 minutes. In our experiments, first we use 80% of the renewable energy generation data as training set, and the rest 20% data as testing data to predict the tail distribution of each energy source. Second, we use 80% of the CPU utilization historical data in each PM throughout time as the training data and the rest 20% data as testing data, and then add the values of all PMs in the same PM area to get the predicted energy consumption of each PM area.

B. Compared Methods

As our work is the first to handle the problem, we cannot find comparable methods within our knowledge. We choose the following three methods for comparison, and the details of the methods are described in Section II. 1) Renewable and Cooling Aware Workload Management (RCA) [8]. 2) Green Scheduling for Cloud Datacenters (GS) [7]. 3) Renewable Energy-Aware Reinforcement Learning (REA) [10]. As the works in [7], [10] are for multiple datacenters and the energy sources already supply energy to their associated datacenters, we make changes to adapt these works to our single-datacenter scenario. Specifically, we evenly distribute the energy sources to the PM areas, that is, each energy source group supplies energy to one PM area. Also, we set the parameter values based on these articles.

In our experiments, first, we compare our renewable energy generation prediction method with the other two renewable energy generation prediction methods, which are pattern matching [7] and K-nearest neighbors (KNN) [8]. Second, we compare our energy demand prediction method with the other three energy demand prediction methods, which are pattern matching [64], Neural Network (NN) [65] and Fast Fourier Transform (FFT) [66]. Third, we compare the performance of our method with the other compared methods in [7], [8], [10] with their own prediction methods and with our LSTM prediction method. For our methods, we use RL to represent the RL-based method, RL+E to represent the RL-based method with ECRA only and RL+E+F to represent our RL-based method with the two enhancement methods.

C. Performance Metrics

- *Prediction accuracy.* To measure the prediction accuracy of renewable energy generation and energy demands, we measure the prediction accuracy as below: $A_n = 1 - \frac{|P_n - R_n|}{R_n}$ where A_n is the prediction accuracy of n^{th} prediction, P_n is the predicted value of n^{th} prediction and R_n is the real value of n^{th} prediction.
- *Uninterrupted PM area ratio.* It is defined as the percentage of PM areas that always receive renewable energy no less than their demands. It is calculated by: $B = 1 - \frac{V}{J}$ where V is the number of PM areas that the energy demands cannot be satisfied and J is the total number of PM areas.
- *SLO satisfaction ratio.* It is defined as the percentage of jobs whose SLOs are satisfied. If one PM area does not receive enough renewable energy for its energy demand in time period T, we assume that all the jobs running on this PM area cannot successfully run in that time period. We run the trace data 50 times in our experiment. In the experiment, if the number of successful running times for one job over the total number of the job running times is higher or equal to this job's SLO, we consider that this job's SLO is satisfied.
- *Monetary cost and carbon emission.* We calculate the total monetary cost based on the real price dataset [62], [63] and Equation (14). Similar to monetary cost, we calculate the total amount of carbon emission based on the real dataset [61] and Equation (15).
- *Time overhead.* We use training time latency and testing time latency to show the time overhead of the prediction methods and the source-PM area mapping methods.

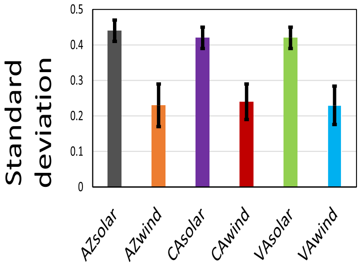


Fig. 2. Standard deviation of renewable energy in AZ, CA, and VA.

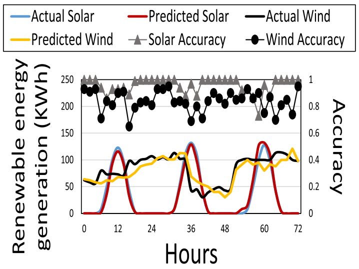


Fig. 3. Prediction of renewable energy generation.

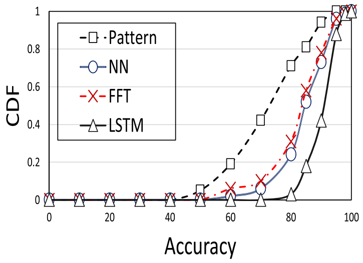


Fig. 4. Energy demand prediction accuracy.

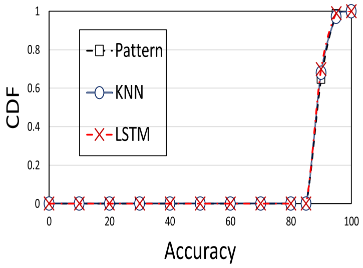


Fig. 5. Solar energy prediction accuracy.

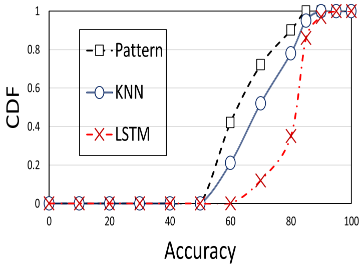


Fig. 6. Wind energy prediction accuracy.

D. *Experimental Results*

1) Renewable Energy Instability: In addition to VA, we also obtained the datasets about regional solar irradiance ($s(t)$) and wind speeds ($v(t)$) at the Arizona (AZ), California (CA) in May, 2011 from the National Renewable Energy Laboratory (NREL) [59], [60]. We choose these three locations because there are large datacenters in these states [67]. For each area, we calculate the standard deviation of $s(t)$ each day in the 30 days and then calculate the average standard deviation per day, and we also calculate the average standard deviation per day for $v(t)$.

Fig. 2 shows the average standard deviation of solar irradiance and wind speed in a month in the three different areas. The error bar means the peak and valley standard deviation value in the month. The result follows AZsolar $\approx$ CAsolar $\approx$ VAsolar $<$ AZwind $\approx$ CAwind $\approx$ VAwind. We see that the solar irradiations at different time slots in one day deviate greatly, so that the solar energy generation in one day is not stable. We found that the peak of the solar irradiation time is around 11 AM to 1PM each day, and in the rest of the day time, the solar irradiation varies. On the other hand, as the wind speed also varies in a day though it is more stable than the solar irradiation. Therefore, the energy generated by wind is also not stable.

2) Renewable Energy Generation Prediction Accuracy: Fig. 3 shows the predicted and actual amount of renewable energy and the prediction accuracy for the solar energy and wind energy on one randomly selected solar generator and one wind energy generator using LSTM in 3 days randomly selected from one month. We see that the predicted values and actual values are almost overlapped and the accuracy stays above 0.8 most of the time. The result also shows that the accuracy of solar energy prediction is higher than wind energy prediction. However, we observe that the deviation of the generated energy amount at different time slots in one day of solar energy is larger than that of wind energy, which also is reflected by the result that the standard deviation of solar irradiance is larger than that of wind energy in Fig. 2. The reason for the higher prediction accuracy of the solar energy compared to the wind energy is that since solar irradiation has certain time pattern, the amounts of the generated renewable energy at the same time in different days are similar, so it is easy to predict. But for wind energy, the relation between wind speed and time is not highly related between the same time points in different days. Therefore, for wind energy, the performance of prediction accuracy is not as high as that of the solar energy.

Figs. 5 and 6 show the CDF (Cumulative distribution function) of the solar energy prediction and wind energy prediction. The result for solar energy prediction follows: Pattern $\approx$ KNN $\approx$ LSTM. All prediction methods achieve high accuracy due to the same reason as explained in Fig. 3. The result for wind energy prediction follows: Pattern $<$ KNN $<$ LSTM. The reason is that, for pattern matching, it only observes the wind energy for each time slot in each day and uses the same value for the same time slot in different days. Since the wind speed is not highly related between the same time point in different days, the accuracy for pattern matching is worse than KNN and LSTM. For KNN, it can

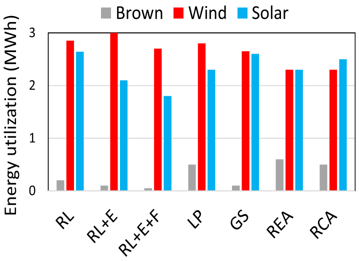
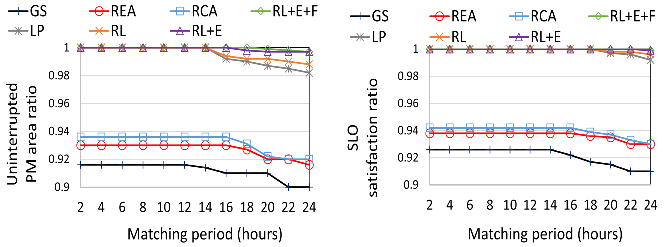


Fig. 7. Energy utilization of different energies.



(a) Uninterrupted PM area ratio (b) SLO satisfaction ratio

Fig. 8. Performance with different matching periods.

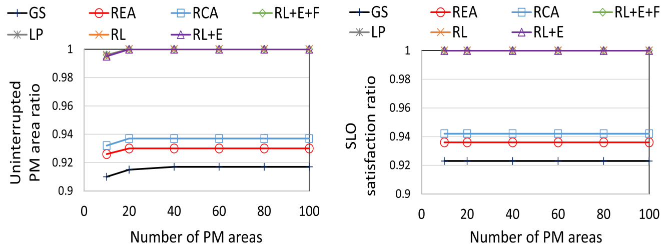
classify the time period with similar wind speed, so in certain time slots, it achieves better accuracy than pattern matching. However, it can't consider the wind speed sequence in the entire time period. LSTM can consider the entire time period when predicting the wind speed in each time slot, thus producing the highest accuracy.

3) Energy Demand Prediction Accuracy: Fig. 4 shows the CDF of different prediction methods for energy consumption of PM areas. We choose 100 PM areas in this figure. The result follows: Pattern $<$ NN $\approx$ FFT $<$ LSTM. For pattern matching, it observes the energy consumption for each PM area in each time slot each day and uses the same value for the same time slot in different days. FFT can process the time-series data and then find the most prominent pattern so that it achieves better performance than pattern matching. NN can consider the relationships between time points in the time sequence data, and LSTM can consider the relationship between time points in a longer time length, so LSTM can more accurately predict the energy consumption in each time slot than NN.

4) Performance Comparison With Compared Methods: LP is used to represent out linear programming method. Fig. 7 shows the energy utilization of different types of energies of each method. We observe that for brown energy: REA $>$ RCA $\approx$ LP $>$ RL $>$ RL+E $\approx$ GS $>$ RL+E+F. RCA and LP use around 17% less brown energy than REA, RL uses around 60% less brown energy than RCA, RL+E and GS use around 50% less brown energy than RL, and RL+E+F consumes around 50% less brown energy than RL+E and GS. For REA, it aims to minimize the total energy monetary cost. Since the brown energy is cheaper than solar energy, it uses more brown energy to minimize the total price. For RCA, although it aims to minimize the total monetary cost, it tries to use more solar energy. LP consumes slightly more brown energy than RL. Since GS aims to minimize the total carbon emission, it uses renewable energy resource as much as possible, which leads to low total brown energy usage. ECRA can dynamically adjust the amount of computing resource assigned to each job to handle renewable energy supply shortage and overage under varying renewable energy supply. Thus, RL+E uses more renewable energy compared with RL, resulting in less brown energy utilization. Specifically, RL+E uses 5% more wind energy, 20% less solar energy and 50% less brown energy than RL. FPES restarts some potentially failed jobs earlier to avoid energy waste for useless job running due to job failures so that RL+E+F consumes less brown energy than RL and RL+E. Specifically, RL+E+F consumes 10% less wind energy, 14% less solar energy and 50% less brown energy than RL+E.

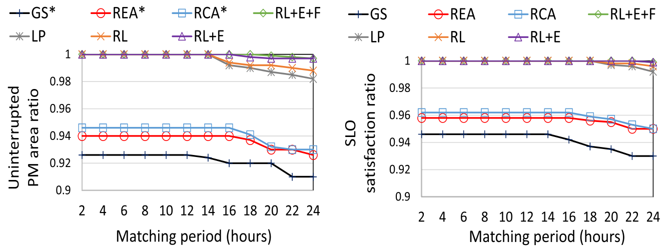
We observe that the wind energy utilization follows RL+E $>$ RL $>$ LP $\approx$ RL+E+F $\approx$ GS $>$ RCA $\approx$ REA, RL+E uses around 5% more wind energy than RL, RL uses around 2% more wind energy than LP, RL+E+F, and GS, which use around 17% more wind energy than RCA and REA. We also observe that the solar energy utilization follows RL $\approx$ GS $>$ RCA $>$ REA $\approx$ LP $>$ RL+E $>$RL+E+F. RL and GS use around 6% more solar energy than RCA, RCA uses around 9% more solar energy than REA and LP, which use around 10% more solar energy than RL+E. RL+E uses around 17% more solar energy than RL+E+F. The wind energy is cheaper than the solar energy and also its carbon emission is lower than solar and brown energy. Therefore, since RL and LP aim to both reduce the total carbon emission and total energy monetary cost, wind is a better choice, so they use more wind energy than others. Because LP uses more brown energy, it uses less wind and solar energy than RL. In addition, RL+E uses more wind energy and less solar energy compared with RL. It is because wind energy is more unstable and cheaper compared with solar energy, ECRA can adjust the amount of computing resource assigned to the jobs upon supply shortage and overage and they consume the same amount of total energy. FPES can restart the potentially failed jobs earlier to avoid energy waste for useless job running so RL+E+F consumes less wind energy and also solar energy compared with RL+E. GS aims to minimize the total carbon emission, so it uses more wind energy and solar energy than others. REA and RCA aim to minimize the total energy monetary cost and RCA tries to use more solar energy. As a result, REA uses more brown energy (the cheapest energy) and RCA uses more solar energy than others.

Fig. 8 shows the uninterrupted PM area ratio and SLO satisfaction ratio versus different matching time periods. In Fig. 8(a), the result follows RL $\approx$ RL+E $\approx$ RL+E+F $\approx$ LP $>$ RCA $\approx$ REA $>$ GS. On average, our approaches and LP generate around 8% higher uninterrupted PM area ratio than RCA and REA, which generate around 2% higher uninterrupted PM area ratio than GS. Our RL and LP use LSTM to more accurately predict the amount of generated renewable energy of each energy generator and energy demand of each PM area, so PM areas' energy demands are satisfied most of the time, thus producing the highest uninterrupted PM area ratio. As shown previously in Figs. 5, 6 and 4, other prediction methods used in RCA, REA and GS have lower accuracy than LSTM, so more PM



(a) Uninterrupted PM area ratio (b) SLO satisfaction ratio

Fig. 9. Performance with different number of PM areas.



(a) Uninterrupted PM area ratio (b) SLO satisfaction ratio

Fig. 10. Performance with different matching time periods using our prediction method.

areas experience interruptions. Therefore, RCA, REA and GS generate lower uninterrupted PM area ratios. As shown in Fig. 7, GS uses more renewable energies than REA and RCA, so GS has lower uninterrupted PM area ratio than those of REA and RCA because of the instability of the renewable energies. We see that when the matching period is high, RL+E+F and RL+E generates around 0.7% higher uninterrupted PM area ratio than RL. This is because ECRA reduces or increases the computing resources of jobs considering their deadlines and SLOs upon an energy shortage or overage happens.

In Fig. 8(b), we can observe that the SLO satisfaction ratio result follows RL ≈ RL+E ≈ RL+E+F ≈ LP > RCA ≈ REA > GS. On average, our approaches and LP generate around 7% higher SLO satisfaction ratio than RCA and REA, which generate around 2% higher SLO satisfaction ratio than GS. RL and LP try to power each PM area with renewable energy generators that have probability no less than the PM area's SLO to produce the amount no less than its energy demand to ensure that insufficient renewable energy supply will not lead to SLO violations. As a result, they produce the highest SLO satisfaction ratio. REA, RCA and GS do not consider the possible SLO violations due to insufficient renewable energy supply and the subsequent energy supply switch. Therefore, they produce lower SLO satisfaction ratios. Since GS has the lowest uninterrupted PM area ratio, more jobs tend to experience SLO violations, so GS produces the lowest SLO satisfaction ratio. We see that when the matching period is high, RL+E+F and RL+E generates 0.2% higher SLO satisfaction ratio than RL due to the same reason as in Fig. 8(a).

In Fig. 8, we also observe that as the matching period increases, the values of the two metrics of all methods decrease. This is because when the matching period is longer, the predicted values tend to be less inaccurate. Therefore, less frequent scheduling and longer scheduling time period make it less likely to guarantee that the supplied renewable energy is no less than the energy demand of a PM area. However, the slower decreasing rates of RL and LP mean that their scheduling time period can be longer than other methods, which saves computation resources.

Fig. 9(a) and (b) show the uninterrupted PM area ratio and the SLO satisfaction ratio versus different number of PM areas. The results show RL ≈ RL+E ≈ RL+E+F ≈ LP > RCA > REA > GS. On average, for both ratios, our approaches and LP produce around 6–7% higher ratios than RCA, RCA produces around 0.6–0.7% higher ratios than REA, and REA produces around 1-2% higher ratios than GS. The relative performance results are consistent with those in Fig. 8 due to the same reasons as explained. For all the methods, as the number of PM areas increases, the uninterrupted PM area ratio and the SLO satisfaction ratio increase slightly. The reason is that, more PM areas mean fewer PMs in a PM area. Then, when a PM area does not receive sufficient renewable energy, fewer PMs and hence fewer jobs are interrupted. Therefore, more PM areas lead to higher uninterrupted PM area ratio and SLO satisfaction. RL, RL+E and RL+E+F have similar performance since the matching period is not long (one hour).

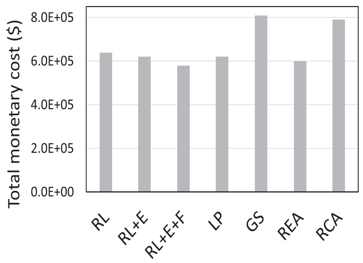


Fig. 11. Total monetary cost.

Fig. 10 shows the uninterrupted PM area ratio and the SLO satisfaction ratio versus different matching time period with our LSTM prediction method for energy demand. In the figure, GS*, REA* and RCA* means that we used LSTM and their scheduling methods as new methods. The results follow RL ≈ RL+E ≈ RL+E+F ≈ LP > RCA* > REA* > GS*. On average, for both ratios, our approaches and LP produce 4-6% higher ratios than RCA*, RCA* produces 0.3–0.5% higher ratios than REA*, and REA* produces 2% higher ratios than GS*. The relative performance results are the same as in Fig. 8 due to the same reasons. Comparing this figure and Fig. 8, we observe that the uninterrupted PM area ratio and the SLO satisfaction ratio of GS*, REA* and RCA* are higher than those of GS, REA and RCA, respectively. This result indicates that our prediction method is more accurate and it can help achieve higher uninterrupted PM area ratio and SLO satisfaction ratio. We see that when the matching period is high, RL+E+F and RL+E generates 0.7% higher uninterrupted PM area ratio than RL and 0.2% higher SLO satisfaction ratio than RL.

Fig. 11 shows the total monetary cost of each method and the results follow GS > RCA > RL > LP ≈ RL+E ≈ REA > RL+E+F. RCA costs $20000 (2%) less than GS, RL costs

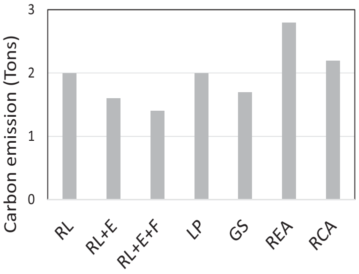


Fig. 12. Carbon emission.

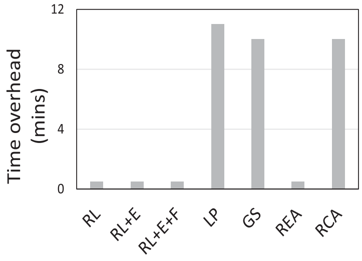


Fig. 13. Time overhead.

$150000 (19%) less than RCA, LP, RL+E and REA costs $20000 (3%) less than RL, and RL+E+F costs $40000 (6%) less than LP, RL+E and REA. GS aims to minimize the total carbon emission, so it uses renewable energy resource as much as possible though the price of the solar energy is much more higher than the wind and brown energy, thus generating the highest energy monetary cost. RCA aims to minimize the total monetary cost while trying to use solar energy as much as possible. Our LP and RL tend to choose wind energy due to its cheaper price and lower carbon emission as explained above, their total energy monetary cost is lower than GS and RCA. REA tends to use more brown energy since the goal for REA is to only minimize the total energy monetary cost. Since the price of each energy resource follows solar $>$ brown $\approx$ wind, REA and LP generate lower monetary cost compared with RL. Due to the same reason explained in Fig. 7, RL+E generates $20000 (3%) lower monetary cost than RL and RL+E+F generates $40000 (6%) lower monetary cost than RL+E. So RL+E+F generates the lowest monetary cost.

Recall that the carbon emission rate for each energy resource follows brown $>$ solar$>$wind. Fig. 12 shows the carbon emission of each method. The results show REA $>$ RCA $>$ LP $\approx$ RL $>$ GS $\approx$ RL+E $>$ RL+E+F. RCA reduces 0.6Tons (12%) of carbon emission of REA, LP and RL reduce 0.2Tons (9%) carbon emission of RCA, GS and RL+E reduce 0.4Tons (20%) carbon emission of LP and RL, and RL+E+F reduce 0.2Tons (13%) carbon emission of GS and RL+E. REA tends to use more brown energy since it aims to minimize the total energy monetary cost. Since the carbon emission amount of the brown energy is much higher than the solar energy and wind energy, REA produces the most carbon emission. RCA uses more renewable energy than REA since RCA tries to use solar energy as much as possible, so that RCA generates less carbon emission than REA. Since both LP and RL based methods aim to reduce carbon emission and also total energy monetary cost, they use more wind energy, thus producing less carbon emission. GS only aims to minimize the total carbon emission, so it uses renewable energy resource as much as possible, which produces the lower carbon emission. Due to the same reason explained in Fig. 7, RL+E produces 0.4Tons (20%) lower carbon emission than RL and RL+E+F produces 0.2Tons (13%) lower carbon emission than RL+E, So RL+E+F produces the lowest carbon emission. Combining the results in Figs. 11 and 12, we can conclude that our LP and RL based methods perform well in both reducing total energy monetary cost and carbon emission, while other methods cannot achieve both goals simultaneously.

Fig. 13 shows the time overhead for one decision of each method. The time overhead contains the testing time overhead for RL based methods and REA. Since the training process is offline, we only show the testing time. The time overhead of ECRA includes the operations in Algorithm 1. Since the failure prediction model training can be offline, the time overhead of only FPES includes the prediction model inference time and the operations in Algorithm 2. The results show LP $\approx$ GS $\approx$ RCA $>$ REA $\approx$ RL+E $\approx$ RL $\approx$ RL+E+F. REA and our approaches generates 95% less time overhead than other compared methods, and our approaches generate similar time overhead. REA and RL based methods use the reinforcement learning model, which needs a long time for training but much less time to make the decision. After the model is trained, the decision making process takes a short time, which is much shorter than other methods. Other methods need to solve an optimization problem without the need for training. LP considers more input features (including carbon emission, energy price and SLO violation) than GS and RCA, so its time overhead for making a decision is much more than GS and RCA. Since GS and RCA consider less input features, their optimization problems are easier to solve, so their time overheads of making a decision are lower than LP. Since the time overhead of the two enhancement methods is low, RL, RL+E, and RL+E+F have similar overhead.

VIII. Conclusion

Renewable energy supply is a promising solution to current datacenter energy supply, which is much more environment-friendly. However, the instability of renewable energy may lead to insufficient energy supply to the datacenter, resulting in job running interruption or even failures. Previous work attempting to achieve higher energy generation prediction cannot completely handle this problem since sufficient renewable energy supply cannot be guaranteed due to energy instability.

In this article, we propose a renewable energy resource allocation system for a cloud datacenter. Our objective is to avoid SLO violations due to interruption from insufficient renewable energy supply while minimizing the total energy monetary cost and total carbon emission. First, using deep learning technique, the system predicts the tail distribution of each renewable energy source at each time slot in the next time period. Second, it predicts the energy demand in each PM area by predicting the CPU utilization for each PM in a PM area. Third, based on the predicted results, the system assigns renewable energy sources to PM areas to solve the above problem using RL-based method and linear programming method. In addition, we propose two enhancement methods: energy-driven computing resource assignment method (ECRA) and failure prediction based energy saving (FPES). ECRA dynamically adjusts the amount of computing resources assigned to each job based on its urgent value to handle renewable energy supply overage and outrage while avoiding SLO violations and reducing job completion time. FPES restarts some jobs with high predicted failure probabilities to avoid energy waste for useless job running of failed jobs. Our extensive real trace driven experiments show that our system achieves superior performance than other methods in terms of the aforementioned goals, our enhancement methods are effective in improving system performance. Since job migration between PMs in load balancing in a cloud datacenter can be very resource intensive, in our future work, we will further study how to minimize the energy cost of the job migration process.

References

[1] How much energy do data centers really use?, 2020. [Online]. Available: https://energyinnovation.org/2020/03/17/how-much-energy-do-data-centers-really-use/

[2] The real amount of energy a data center uses, 2022. [Online]. Available: https://www.akcp.com/blog/the-real-amount-of-energy-a-data-center-use/

[3] For the balance sheet and the sake of the planet, its time data centers reduce power consumption by improving utilization, 2021. [Online]. Available: https://datacenterfrontier.com/data-center-management-solutions-reduce-costs-and-carbon-emissions/

[4] Emission regulation. Accessed: Jan. 2023. [Online]. Available: http://sapientservicesllc.com/regulations-govern-data-center-operations/

[5] How Microsoft's new datacenter region in Sweden incorporates the company's sustainability commitments, 2021. [Online]. Available: https://news.microsoft.com/europe/features/how-microsofts-new-datacenter-region-in-sweden-incorporates-the-companys-sustainability-commitments/

[6] A. Malekpour and A. Pahwa, "Stochastic networked microgrid energy management with correlated wind generators," *IEEE Trans. Power Syst.*, vol. 32, no. 5, pp. 3681–3693, Sep. 2017.

[7] C. Gu, C. Liu, J. Zhang, H. Huang, and X. Jia, "Green scheduling for cloud data centers using renewable resources," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2015, pp. 354–359.

[8] Z. Liu et al., "Renewable and cooling aware workload management for sustainable data centers," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Model. Comput. Syst.*, 2012, pp. 175–186.

[9] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, and W. Dai, "Energy efficient task allocation and energy scheduling in green energy powered edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 89–99, 2019.

[10] C. Xu, K. Wang, P. Li, R. Xia, S. Guo, and M. Guo, "Renewable energy-aware big data analytics in geo-distributed data centers with reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 205–215, First Quarter 2020.

[11] I. De Courchelle, T. Guérout, G. Da Costa, T. Monteil, and Y. Labit, "Green energy efficient scheduling management," *Simul. Modelling Pract. Theory*, vol. 93, pp. 208–232, 2019.

[12] H. Wang, J. Gong, Y. Zhuang, H. Shen, and J. Lach, "Healthedge: Task scheduling for edge computing with health emergency and human behavior consideration in smart homes," in *Proc. IEEE Int. Conf. BigData*, 2017, pp. 1213–1222.

[13] J. Gao, H. Wang, and H. Shen, "Smartly handling renewable energy instability in supporting a cloud datacenter," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2020, pp. 769–778.

[14] J. Gao, H. Wang, and H. Shen, "Task failure prediction in cloud data centers using deep learning," *IEEE Trans. Serv. Comput.*, vol. 15, no. 3, pp. 1411–1422, May/Jun. 2022.

[15] Source code. Accessed: Jan. 2023. [Online]. Available: https://github.com/pcl-projects/Renewable-Energy-Allocation-System

[16] G. Chen et al., "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proc. USENIX Symp. Netw. Syst. Des. Implementation*, 2008, pp. 337–350.

[17] B. Heller et al., "ElasticTree: Saving energy in data center networks," in *Proc. USENIX Symp. Netw. Syst. Des. Implementation*, 2010, pp. 249–264.

[18] Y. Lin and H. Shen, "EAFR: An energy-efficient adaptive file replication system in data-intensive clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1017–1030, Apr. 2017.

[19] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Energy-efficient cloud resource management," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2014, pp. 386–391.

[20] M. Xu and R. Buyya, "Managing renewable energy and carbon footprint in multi-cloud computing environments," *J. Parallel Distrib. Comput.*, vol. 135, pp. 191–202, 2020.

[21] Z. Liu, H. Yu, R. Liu, M. Wang, and C. Li, "Configuration optimization model for data-center-park-integrated energy systems under economic, reliability, and environmental considerations," *Energies*, vol. 13, 2020, Art. no. 448.

[22] S. Nayak, S. Panda, S. Das, and S. Pande, "An efficient renewable energy-based scheduling algorithm for cloud computing," in *Proc. Int. Conf. Distrib. Comput. Internet Technol.*, 2021, pp. 81–97.

[23] Z. Wen, Y. Wang, and F. Liu, "StepConf: Slo-aware dynamic resource configuration for serverless function workflows," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1868–1877.

[24] G. Safaryan, A. Jindal, M. Chadha, and M. Gerndt, "SLAM: SLO-aware memory optimization for serverless applications," in *Proc. IEEE 15th Int. Conf. Cloud Comput.*, 2022, pp. 30–39.

[25] S. K. Shukla, D. Ghosal, and M. K. Farrens, "Understanding and leveraging cluster heterogeneity for efficient execution of cloud services," in *Proc. IEEE 10th Int. Conf. Cloud Netw.*, 2021, pp. 56–64.

[26] C. Zhang, M. Yu, F. Yan, and W. Wang, "Enabling cost-effective, SLO-aware machine learning inference serving on public cloud," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1765–1779, Third Quarter 2022.

[27] S. K. Shukla and M. K. Farrens, "Leveraging network delay variability to improve QoE of latency critical services," in *Proc. IEEE Int. Conf. Netw., Archit. Storage*, 2021, pp. 1–8.

[28] D. Alsadie, Z. Tari, E. J. Alzahrani, and A. Y. Zomaya, "Life: A predictive approach for VM placement in cloud environments," in *Proc. Int. Symp. Netw. Comput. Appl.*, 2017, pp. 1–8.

[29] R. K. Ramesh, H. Wang, H. Shen, and Z. Fan, "Machine learning for load balancing in cloud datacenters," in *Proc. IEEE/ACM 21st Int. Symp. Cluster, Cloud Internet Comput.*, 2021, pp. 186–195.

[30] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proc. Symp. Operating Syst. Princ.*, 2017, pp. 153–167.

[31] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Commun. Mag.*, vol. 57, no. 6, pp. 114–119, Jun. 2019.

[32] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi, "Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 923–934, Apr. 2020.

[33] J. Kumar, R. Goomer, and A. Singh, "Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters," *Procedia Comput. Sci.*, vol. 125, pp. 676–682, 2018.

[34] W. Ding, F. Luo, C. Gu, H. Lu, and Q. Zhou, "Performance-to-power ratio aware resource consolidation framework based on reinforcement learning in cloud data centers," *IEEE Access*, vol. 8, pp. 15472–15483, 2020.

[35] X. Zhou, K. Wang, W. Jia, and M. Guo, "Reinforcement learning-based adaptive resource management of differentiated services in geo-distributed data centers," in *Proc. Int. Symp. Qual. Serv.*, 2017, pp. 1–6.

[36] S. Telenyk, E. Zharikov, and O. Rolik, "Modeling of the data center resource management using reinforcement learning," in *Proc. Int. Sci.-Practical Conf. Problems Infocommunications Sci. Technol.*, 2018, pp. 289–296.

[37] H. Shen and L. Chen, "Distributed autonomous virtual resource management in datacenters using finite-Markov decision process," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3836–3849, Dec. 2017.

[38] R. Komp, *Practical Photovoltaics: Electricity From Solar Cell*, 3rd ed. Ann Arbor, MI, USA: Aatec publications, 2003.

[39] P. Gipe, *Wind Power, Revised Edition: Renewable Energy for Home, Farm, and Business*. Chelsea, VT, USA: Chelsea Green, 2004.

[40] H. Shen and L. Chen, "CompVM: A complementary VM allocation mechanism for cloud systems," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1348–1361, Jun. 2018.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, 1997.

[42] J. Gao, H. Wang, and H. Shen, “Task failure prediction in cloud data centers using deep learning,” in *Proc. IEEE Int. Conf. BigData*, 2019, pp. 1111–1116.

[43] V. Mnih, K. Kavukcuoglu, and D. Silver, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529–533, 2015.

[44] L. Tan, S. Song, P. Wu, Z. Chen, R. Ge, and D. Kerbyson, “Investigating the interplay between energy efficiency and resilience in high performance computing,” in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2015, pp. 786–796.

[45] A. Legrand, D. Trustram, and S. Zrigui, “Adapting batch scheduling to workload characteristics: What can we expect from online learning?,” in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2019, pp. 686–695.

[46] P. Kochovski, R. Sakellariou, and M. Bajec, “An architecture and stochastic method for database container placement in the edge-fog-cloud continuum,” in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2019, pp. 396–405.

[47] J. Abara, “Applying integer linear programming to the fleet assignment problem,” *Interfaces*, vol. 19, pp. 20–28, 1989.

[48] M. Canale, L. Fagiano, and M. Milanese, “KiteGen: A revolution in wind energy generation,” *Energy*, vol. 34, pp. 355–361, 2009.

[49] X. Fan, C. Ellis, and A. Lebeck, “The synergy between power-aware memory systems and processor voltage scaling,” in *Proc. Int. Workshop Power-Aware Comput. Syst.*, 2003, pp. 164–179.

[50] W. Fang, Y. Guo, W. Liao, K. Ramani, and S. Huang, “Big data driven jobs remaining time prediction in discrete manufacturing system: A deep learning-based approach,” *Int. J. Prod. Res.*, vol. 58, pp. 2751–2766, 2020.

[51] Implementing SLOs, 2020. [Online]. Available: https://sre.google/workbook/implementing-slos//

[52] Y. Sharma, B. Javadi, W. Si, and D. Sun, “Reliability and energy efficiency in cloud computing systems: Survey and taxonomy,” *J. Netw. Comput. Appl.*, vol. 74, pp. 66–85, 2016.

[53] M. Kweun, W. Lee, G. Kim, J. Hwang, and Y. Lee, “Lineage checkpoint approach for long-lineage problem in apache spark,” in *Proc. IEEE Int. Conf. Big Data*, 2020, pp. 5733–5735.

[54] C. Reiss, J. Wilkes, and J. Hellerstein, “Google cluster-usage traces: Format+ schema,” Google Inc., White Paper, vol. 1, 2011.

[55] R. Bashroush and A. Lawrence, “Beyond pue: Tackling it’s wasted terawatts,” 2020. [Online]. Available: https://uptimeinstitute.com/beyond-puetackling-it’s-wasted-terawatts

[56] Oracle SLO requirements. Accessed: Jan. 2023. [Online]. Available: https://docs.oracle.com/en/enterprise-manager/index.html

[57] C. Ren, D. Wang, B. Urgaonkar, and A. Sivasubramaniam, “Carbon-aware energy capacity planning for datacenters,” in *Proc. Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst.*, 2012, pp. 391–400.

[58] C. Stewart and K. Shen, “Some joules are more precious than others: Managing renewable energy in the datacenter,” in *Proc. Workshop Power Aware Comput. Syst.*, 2009, pp. 15–19.

[59] NREL solar radiation research laboratory-solar dataset. Accessed: Jan. 2023. [Online]. Available: https://midcdmz.nrel.gov/apps/sitehome.pl?site=BMS

[60] NREL wind technology center-wind dataset. [Online]. Available: https://midcdmz.nrel.gov/apps/sitehome.pl?site=NWTC

[61] Measurement and instrumentation data center. Accessed: Jan. 2023. [Online]. Available: https://midcdmz.nrel.gov/l

[62] Wholesale electricity and natural gas market data. Accessed: Jan. 2023. [Online]. Available: https://www.eia.gov/electricity/wholesale/

[63] Which is the cheapest renewable energy supplier in 2019. Accessed: Jan. 2023. [Online]. Available: https://theswitch.co.uk/blog/energy/cheapest-green-supplier

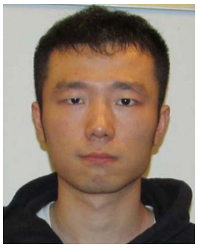
[64] R. Karp and M. Rabin, “Efficient randomized pattern-matching algorithms,” *IBM J. Res. Develop.*, vol. 31, pp. 249–260, 1987.

[65] D. Specht, “A general regression neural network,” *IEEE Trans. Neural Netw.*, vol. 2, no. 6, pp. 568–576, Nov. 1991.

[66] H. Sorensen, “Real-valued fast fourier transform algorithms,” *IEEE Trans. Acoust. Speech. Signal Process.*, vol. 35, no. 6, pp. 849–863, Jun. 1987.

[67] M. Xua and R. Buyyab, “Managing renewable energy and carbon footprint in multi-cloud computing environments,” *J. Parallel Distrib. Comput.*, vol. 135, pp. 191–202, 2019.

Haiying Shen (Senior Member, IEEE) received the BS degree in computer science and engineering from Tongji University, China in 2000, and the MS and PhD degrees in computer engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Associate Professor with the Department of Computer Science, the University of Virginia. Her research interests include distributed computer systems, cloud computing, Big Data, distributed machine learning and cyber-physical systems. She is a Microsoft Faculty fellow of 2010, and a senior member of the ACM.



Haoyu Wang received the BS degree from the University of Science & Technology of China, and the MS degree from the Columbia University in the city of New York. He is currently working toward the PhD degree with the Department of Computer Science of University of Virginia. His research interests include data center, cloud and distributed networks.

Jiechao Gao received the BS degree from Jilin University 2016, and the MS degree from Columbia University in the city of New York 2018. He is currently working toward the PhD degree with the Department of Computer Science of University of Virginia. His research interests include distributed networks, cloud computing, machine learning algorithms and applications.

Rajkumar Buyya (Fellow, IEEE) is a Redmond Barry distinguished professor and director of the Cloud Computing and distributed systems (CLOUDS) Laboratory with the University of Melbourne, Australia. He has authored more than 850 publications and seven text books including ”Mastering Cloud Computing” published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. Software technologies developed under Dr. Buyya’s leadership have gained rapid acceptance and are in use with several academic institutions and commercial enterprises in 50+ countries around the world.