GreenFog: A Framework for Sustainable Fog Computing

Adel N. Toosi
Department of Software Systems and Cybersecurity, Monash University
Clayton, Victoria, Australia
adel.n.toosi@monash.edu

Chayan Agarwal
Department of Software Systems and Cybersecurity, Monash University
Clayton, Victoria, Australia

Lena Mashayekhy
Department of Computer and Information Sciences, University of Delaware
Newark, Delaware, USA
mlena@udel.edu

Sara Kardani Moghaddam
Department of Software Systems and Cybersecurity, Monash University
Clayton, Victoria, Australia

Redowan Mahmud
School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University
Perth, Western Australia, Australia
mdredowan.mahmud@curtin.edu.au

Zahir Tari
School of Computing Technologies, STEM College, RMIT University
Melbourne, Victoria, Australia
zahir.tari@rmit.edu.au

ABSTRACT

While the Fog computing model is the platform of choice for many IoT applications, the alarming rate of increase in energy demand and carbon footprint of Fog environments becomes a critical issue. It is, therefore, necessary to reduce the percentage of brown energy consumption in these systems and integrate renewable energy use into Fog computing environments. Renewable energy sources, however, are prone to availability fluctuations due to their variable and intermittent nature. In this paper, we propose a new Fog computing framework using renewables and design various optimization techniques, including linear programming optimization, linear regression estimation, and Multi-Armed Bandits (MAB) learning to optimize renewable energy use in Fog environments based on the novel idea of load shaping with adaptive Quality of Service (QoS). The goal of the proposed framework is to favor green energy utilization depending on the available information in renewable energy-powered Fog environments. The proposed optimization techniques are used to achieve this goal by dynamically adjusting QoS and autoscaling of resources. The proposed framework, along with the optimization techniques, are tested on a real-world micro data center (Fog environment) powered by solar energy sources connected to multiple IoT devices. The results show that our proposed framework can offer up to 15% brown energy usage reduction while efficiently adjusting the QoS of applications.

1 INTRODUCTION

The Internet-of-Things (IoT) is growing rapidly, and the number of IoT devices is expected to increase to 75.4 billion in 2025, according to Statista [2]. The Fog computing paradigm seeks to minimize the cost and latency of delay-sensitive IoT applications in many domains such as self-driving cars, telemedicine, and Industry 4.0. This novel paradigm processes the enormous amount of data continuously being generated by different applications close to the source, at the boundary of the network, instead of being sent to the cloud or large remote data centers. In practice, the distribution of such computing services throughout the edge is accomplished by having many small-sized devices or clusters of servers referred to as "Fog environments" or "micro data centers."

The rise of IoT and Fog computing elicits an increase in global energy consumption and has a massive impact on the carbon footprint of the ICT (Information Communication Technology) industry. A report by Xailient [28] indicated that AI-powered IoT cameras are predicted to add over 4 trillion kilograms in annual carbon dioxide emissions (Kg CO2e) by 2030, which is the equivalent of adding 860 million cars to the road in a decade. Therefore, Fog computing requires innovations in energy supply, management, and use [32]. To further lower the carbon footprint of the IoT ecosystem, it is widely accredited that renewable or green energy sources must be used as the primary power supply of Fog environments.

One of the main challenges of powering Fog environments with renewable energy sources such as solar is *intermittency* and *variability* of power input. A promising solution to this challenge is matching demand to supply of green and renewable energy. In this paper, in order to effectively utilize renewable energy sources, we propose a framework called *GreenFog* that conducts load shaping of Fog environments powered by on-site renewable energy. This framework operates without the usage of a battery as we believe installation of battery storage for Fog environments is often prohibitive. The main reasons are: 1) the cost of purchasing and maintaining lithium-ion batteries, the most practical solution to storing energy, can dominate the total cost of the system; 2) lithium-ion batteries cannot be totally recycled due to technical constraints, economic barriers, and regulatory gaps; and finally 3) the lifespan lithium-ion batteries are comparatively shorter than other parts of the system. In addition, batteries are most suited for off-grid applications. Thus, in this work, we assume that the Fog environment is not connected to batteries and relies on a grid-connected on-site power system.

GreenFog focuses on dynamic adjustment of Quality of Service (QoS) for IoT applications to match the energy consumption of the Fog environment with its renewable energy supply. Dynamic QoS management is acceptable for many IoT or smart applications in practice. For example, it might be admissible to reduce the accuracy of an approximation task that estimates the number of cars passing traffic intersections to control the traffic lights during the night. Or in a real-time video analytics application that detects objects

in the stream of video frames, resolution or frame rate can be adjusted in off-peak hours to save resource usage. Owing to these opportunities, we propose GreenFog to match the application's QoS with the renewable energy availability in a multi-server Fog environment. GreenFog shapes the energy consumption through dynamic scaling of the application containers which are becoming the norm for building IoT applications. At the same time, GreenFog controls the QoS of tasks submitted to the gateway, e.g., the frame rate or image quality at which the video streaming data must be processed. This is done through a designing a feedback loop mechanism in which the cluster orchestrator asks IoT applications to adjust their QoS requirements. The Fog environment resources are then sized according to the number of active containers. Thus, the **key contributions** of the paper are as follows:

- A framework called *GreenFog* for dynamic QoS management of the IoT application based on load shaping according to the renewable energy availability at the renewable energy-powered Fog environment.
- The optimal offline model for load shaping with prior knowledge of renewable energy availability to minimize brown energy usage, while providing required QoS for the hosted IoT application. The optimal model is used as a baseline for comparison to other proposed load shaping algorithms.
- A fast and lightweight reactive heuristic approach based on a linear regression model and profiling to overcome the complexity of the optimal model and lack of in advance knowledge of energy availability.
- A machine learning technique using Multi-Armed Bandit (MAB) model which automatically learns how QoS adjustment affects energy consumption and dynamically adapts QoS to maximize the utilization of renewable energy and remove the need for profiling.
- Validation of GreenFog and performance evaluation of proposed load shaping algorithms in a real implementation using real-world traces of renewable energy and application demonstration using object detection tasks in live stream of videos.

The remainder of the paper is organized as follows: the background is discussed in Section 2. A system overview is discussed in Section 3. Following this, the optimal model is discussed in Section 4. Since it is impractical to collect all required information for the optimal model, a proposed fast and lightweight reactive heuristic using a linear regression model is discussed in Section 5. A proposed threshold-based autoscaler for the right sizing of active resources in the Fog environment is discussed in Section 6. Since both the previous approaches require either a priori knowledge or profiling, an online learning solution based on Multi-Arm Bandit (MAB) model is proposed in Section 7. Section 8 shows the evaluation results of comparing the proposed techniques under a real-world setup and application demonstrator. Finally, Section 11 presents the conclusions and future directions.

2 BACKGROUND AND MOTIVATIONAL SCENARIO

2.1 Challenges of using renewable energy in Fog Computing

The utilization of green energy in Fog computing is a relatively new area of research and is getting significant attention than ever before [5] [31] [18]. Recent studies considered using renewable energy to power Fog environments, and efforts have been made in the research community to facilitate such integration [6, 21]. However, using renewable energy to power Fog environments is challenging due to the stochastic behavior of dynamic workloads served and the fluctuation and variability of renewable energy generation.

Despite the challenges of using renewable energy, micro data centers at the edge locations can rely on these volatile renewable energy sources to support reliable operation [9]. At the time of no renewable energy, these systems heavily depend on the conventional utility power grid where the energy sources are predominantly brown energy like coal or nuclear. Using batteries in many cases is also unfavorable as battery-related costs are high and can dominate the cost of solar- or wind-powered systems. In addition, batteries use chemicals that are harmful to the environment [11]. Current technologies such as lead-acid and lithium-ion batteries are also not sustainable [19] [27]. Therefore, it is crucial to optimize renewable energy use for Fog environments to make it sustainable.

Many researchers have explored this problem with different solutions, such as placement of application components in a way that the utilization of renewable energy sources is maximized while an acceptable latency and QoS are provided [22]. They have also explored offloading tasks to other Fog environments with excess or enough renewable energy [1] [18] or hardware level solutions such as frequency and modulation level scaling [17]. Different from these approaches, we aim to shape the load of the Fog environment to match the renewable energy supply through QoS adaptation of IoT applications. Such a system can offer incentives like discounts or lower charging rates to users willing to relax their QoS requirements in favor of more efficient renewable energy use (that is, end users might accept a lower QoS in favor of discounted service or opt for low QoS to be more green). The design and implementation of such market and pricing mechanisms fall out of the scope of this work.

2.2 A Motivational Scenario

Cameras are pervasive in numerous smart applications. Cameras can be installed in buildings for surveillance and business intelligence, or deployed on streets for traffic control and crime prevention. Video analytics is an integral part of IoT applications. At the same time, real-time video analytics is one of the killer applications for Edge and Fog Computing [3]. The real-time video analytics applications are most suited for edge deployment due to high volume of data generated by cameras and their low latency requirements.

Video analytics can have very high resource demands [33]. Some of the most accurate Deep Neural Networks for object recognition require 30GFlops to process a single frame [33]. Nevertheless, for most of these applications, the data quality directly drives the QoS. Illustratively, the features from a high-resolution image can be

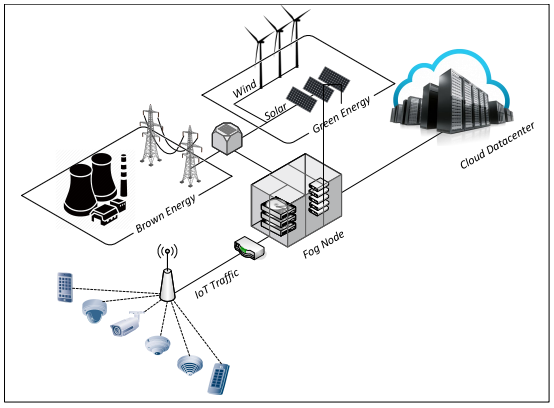

Figure 1: Schematic view of a renewable-powered Fog.

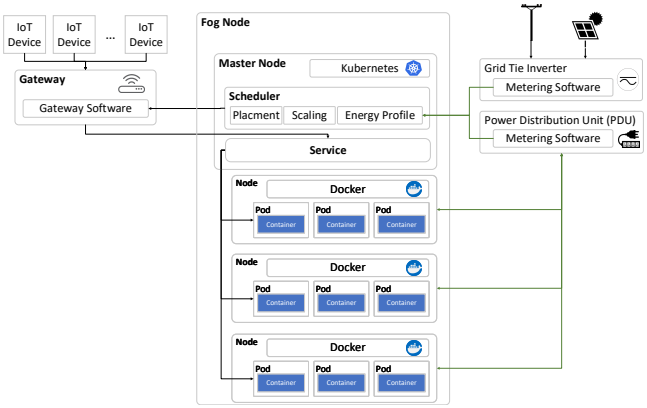

Figure 2: GreenFog Software System Overview

extracted more rigorously than from a low-resolution image; consequently enhancing the accuracy of detecting any event of interest or number of frames analyzed by the system direct impacts the accuracy of the object tracking algorithm. Therefore, application resource demand can be reduced by lowering the QoS requirements. For example, the rate or resolution at which the video frames are sent for image processing can be directly translated to QoS in a video analytics application.

Concomitantly, dynamic QoS control is acceptable for many of these applications. For example, it might be acceptable to increase the error bounds of counting the number of cars crossing an intersection of a road to set the traffic light to minimize resource or energy demand. Or accuracy of an object detection algorithm in pest bird repellent application can be lowered to an acceptable range to reduce resource usage of energy constraint devices [24]. In this context, during resource or energy scarcity or uneven surge in the number of requested tasks, we can determine the most suitable accuracy level or QoS for the application to meet resource/energy constraints. This has motivated us to build a framework to dynamically adapt QoS to shape the load and match energy consumption with the renewable energy supply of the Fog environment. Please note that the acceptable range at which QoS can be moderated is given as an input to such a system. Therefore, the flexibility provided by the application is the key driver of performance.

3 SYSTEM OVERVIEW AND GREENFOG FRAMEWORK

Figure 1 shows a schematic view of a Fog environment powered with an on-site renewable energy generation system with a grid-tied inverter that can work without batteries. The Fog environment is a multi-server cluster that monitors its own power consumption and can trace the amount of generated renewable power available for usage. As we can see in Figure 1, the Fog environment is connected to both the grid (brown energy) for reliability and an onsite renewable energy system (green energy) to reduce its carbon footprint. At the same time, the Fog environment is connected to a gateway through which IoT devices send their processing tasks (requests) to the Fog environment for execution. On the back-end, the Fog environment is connected to the cloud.

Figure 2 illustrates the architecture of our proposed software framework and its various software components. IoT devices are connected to a gateway device running a *gateway software* that would allow the system to set the QoS for the IoT applications. In our proposed framework, IoT devices have APIs that allow gateway software to dynamically adapt QoS requirement remotely by setting configurations at the IoT devices. The gateway software also monitors the request and service rate for these devices. We assume that the Fog environment hosts the IoT applications in the form of containerized micro-services, e.g., Docker Containers. The Fog environment is equipped with a cluster manager and container orchestrator software (e.g., K3s[1]), for automating deployment, scaling, and management of containerized applications. In the Fog environment, a *master node* runs a *scheduler* program responsible for interacting with the gateway software, thereby informing expected QoS and rate of requests. The scheduler also interacts with the energy metering software (*energy profile* component), which monitors available green energy and the current energy consumption of the Fog environment. Through its placement algorithm, the scheduler makes the decision regarding the *placement* of the container instances (Pods in Kubernetes terminology) on different worker nodes. Another major component in the scheduler is the *scaling* component which is responsible for autoscaling of Fog environment resources to meet the demand.

As noted earlier, the scheduler communicates QoS adjustment to the gateway software. The QoS adjustment is basically performed via a comparison between the available renewable power and the current power consumption of the Fog environment. The gateway software accordingly adjusts the QoS of tasks within acceptable Service Level Agreement (SLA) range at which the IoT devices send requests to the Fog environment. In the following sections, we propose different algorithms and optimization techniques to scale resources and adjust QoS. Note that grid energy can be used whenever renewable energy is insufficient to support the energy requirement of the current load in the system.

[1]K3s is a lightweight Kubernetes distribution which is suitable for resource-constrained environments such as Fog or Edge.

4 OPTIMIZATION MODEL

In this section, we propose an optimal model for the orchestration of resources and QoS adjustment in the GreenFog framework. The optimal model will be used as a baseline for evaluation. Assuming time is divided into time slots, we have $t = \{0, \ldots, T\}$, where T is the decision horizon. The green energy system $\mathcal{G}$ at any time t generates e^t units of electricity. Given the fluctuations inherent in the renewable energy, e^t must be predicted with forecasting methods based on previous states of energy generation and the field weather forecast. For example, solar panels electricity generation can be predicted considering several factors including the sky condition, temperature, and the amount of electricity produced in previous hours. However, in this work, we assume that the value of e^t for any time slot t is known *a priori* since optimal model is used as the baseline for evaluation.

The Fog environment $\mathcal{F}$ is a Kubernetes cluster (K3s cluster) that consists of N Kubernetes nodes (or worker machines). Each Kubernetes node $n_i \in N$ hosts at most P pods, where each pod holds a single container. Therefore, the Fog environment has the maximum of $C = P \times N$ containers in full capacity, and at each time t only some containers ($\leq C$) become active to match the demand based on available energy e^t for that time slot. When the demand exceeds the renewable power availability, the gateway adjusts QoS to match demand with the available power. Each pod requires e_p units of electricity. Each Kubernetes node runs on a different physical machine. If a node is not needed, we simply put the machine to a shutdown state. Each machine requires e_b units of energy in a shutdown state, while a running machine requires a further e_n units of energy. Each task requires e_j units of energy added up to other energy terms. In addition, there is an upper limit on the total energy consumption when running all the containers and a lower limit when running one container. In our case, these energy limits determine the upper-bound and lower-bound on the rate of the incoming tasks (QoS), represented by r_{max} and r_{min}, respectively. In other words, these limits (r_{max}, r_{min}) can be set as the lowest and the highest QoS required. Note that in our formulations, for the sake of simplicity, we consider request rate as the QoS factor, e.g., frame rate in our video analytics example. Other QoS factors can be applied to our model similarly. We also assume that each request is mapped to an IoT task in the system, e.g., an image (frame) to an image processing task.

We consider a set of homogeneous tasks generated by the IoT application. The required processing time of a task is denoted by c. The goal of GreenFog is to choose the best rate for the IoT tasks and adjust them accordingly based on the available renewable power input. We define sets of decision variables: x_i^t to determine the number of active pods at Kubernetes node n_i; y^t to determine the arrival rate of requests from IoT devices; and z_i^t to determine if Kubernetes node n_i should be on or off.

Our objective is to minimize the excess energy usage (brown energy usage) and determine the arrival rates of IoT tasks. We formulate this problem as an Integer Program (IP) as follows:

$$\text{Minimize} \sum_{t=1}^{T} E^t \tag{1}$$

where

$$E^t = \sum_{n_i \in N} (e_p \cdot x_i^t + e_n \cdot z_i^t + e_b) + (e_j \cdot y^t) \tag{2}$$

Subject to:

$$E^t \geq e^t \qquad \forall t \in T \tag{3}$$

$$x_i^t \leq P \qquad \forall n_i \in N, t \in T \tag{4}$$

$$x_i^t \leq \mathbf{M} \cdot z_i^t \qquad \forall n_i \in N, t \in T \tag{5}$$

$$\sum_{n_i \in N} z_i^t \leq N \qquad \forall t \in T \tag{6}$$

$$r_{min} \leq y^t \leq r_{max} \qquad t \in T \tag{7}$$

$$c \cdot y^t = \sum_{n_i \in N} x_i^t \qquad \forall t \in T \tag{8}$$

$$x_i^t \in \mathbb{Z}_{\geq 0} \qquad \forall n_i \in N, t \in T \tag{9}$$

$$y^t \in \mathbb{Z}_{\geq 0} \qquad \forall t \in T \tag{10}$$

$$z_i^t \in \{0, 1\} \qquad \forall n_i \in N, t \in T \tag{11}$$

The objective function minimizes the total brown energy consumption subject to constraints. Constraint (3) guarantees that all generated renewable energy must be used. Constraint (4) guarantees that each Kubernetes node has at most P pods. Constraint (5) ensures that the energy cost of turning on a Kubernetes node should be calculated only once. This means that if any pod has been already deployed on a Kubernetes node, there is no additional cost for the node, but for new pods. We model this using the *Big M* method [7].[2] Constraint (6) ensures that the Fog environment does not exceed its capacity of N Kubernetes nodes. Constraint (7) ensures that the selected rate for the IoT tasks is between its minimum and maximum rates. Constraint (8) ensures that enough pods are running to service all the incoming tasks. To ensure this, we multiply the processing time for each job c with the rate of incoming tasks at time t which is denoted by y^t and match this product with the total number of pods running at time t which is denoted as $\sum_{n_i \in N} x_i^t$. Constraints (9) and (10) guarantee that the decision variables x_i^t and y^t are non-negative integers, respectively. Constraints (11) guarantee that decision variable z_i^t is binary.

The proposed IP model requires parameters like the energy per task, energy per pod, energy per node, and also the processing time for each task; gathering such information might not be feasible if not impossible in practice. Furthermore, using the IP model is not viable in practice as it is computationally expensive. Finally, an accurate forecasting of renewable energy power input is a difficult task. Therefore, we only rely on this IP model as a great benchmark to show the best case solutions. Next, we present our proposed online approach based on liner regression and profiling to address these challenges.

5 LINEAR REGRESSION ALGORITHM

To address the problems faced by the IP model, we propose a heuristic based on a linear regression model using profiling to set QoS and scale resources. The proposed linear regression model sets the rate of tasks (QoS) for the IoT devices. Since such a linear regression

[2] Big M method is a method of solving linear programming problems using the simplex algorithm.

model only gives the rate of tasks as output, we propose a threshold-based autoscaling algorithm in Section 6 to scale the Kubernetes cluster as per the computing demand.

Profiling is necessary to gather the data needed to design a linear regression model between the power consumption and the rate of tasks. We expect that there is a simple linear correlation between the power consumption of the Fog environment and its load shaped, based on the setting of the rate of tasks. For profiling, we identify the power consumption of the Fog environment at different task rates. Accordingly we set α (y-intercept) and β (the slope) of the linear regression equation. We also find out the maximum and minimum rate that the Fog environment is able to service such that the average pending time of tasks on each pod does not go lower or higher than certain limits. This is aligned with the IoT application SLA and will be further discussed in Section 8.2.

Algorithm 1 presents the pseudo code of the linear regression method which collects the available green power and computes the rate at which tasks can be processed. The pseudo code shows that how after calculating α (y-intercept) and β (slope) for the linear regression, we can easily set the rate of tasks given the available renewable power. The algorithm is executed recurrently at the specified time intervals. It first reads the current availability of the green power generation and provides it to the regression model. Then, it sets the acceptable rate of the tasks at the gateway software in a way that the Fog environment can fully utilize the available green energy. r_min and r_max are the minimum and maximum acceptable rate according to the SLA, respectively. The algorithm has a time complexity of $O(1)$ providing the highest scalability.

6 THRESHOLD-BASED LINEAR AUTOSCALER

As we discussed earlier, a linear regression model only sets the QoS. Thus, we propose an autoscaling algorithm to scale cluster as per the computing demand. Algorithm 2 shows the pseudo code of the proposed autoscaler with two thresholds to keep the average pending number of tasks among all of the running pods to be no more than the maximum pod load threshold and no less than the minimum pod load threshold. Algorithm 2 is designed based on the state of the art threshold-based autoscaling techniques commonly used in practice. The proposed threshold-based Linear Autoscaler can be extended or replaced with any other autoscaler.

The placement/removal of a pod is performed linearly in a way that all the worker nodes with any pod should be fully packed/empty

Algorithm 1: LinearRegression

```
function LinearRegression(interval)
renPower ← Read the Available Renewable Power
rate ← α + β ∗ renPower
if rate < r_min then
    rate ← r_min
else
    if rate > r_max then
        rate ← r_max
setQoS(rate)
sleep for interval
```

Algorithm 2: Linear Autoscaler

```
function autoScaler(minPodLoad, maxPodLoad)
for p ∈ currentPods do
    pendTasks ← pendTasks + Num of pending tasks in p
end
avgTask ← pendTasks ÷ size(currentPods)
if avgTask < minPodLoad then
    n ← Node at the top of the stack
    Decrease the deployment scale by 1 (remove a Node n's pod)
    if Node n has no active Pod then
        Shutdown Node n
        Remove n from the stack
    end
else
end
if avgTask > maxPodLoad then
    n ← Node at the top of the stack
    if Node n is full then
        Start a new node m
        Add m to the stack
    end
    Increase deployment scale by 1 pod
end
```

before adding/removing a new node. This is due to the fact that having multiple nodes with a fewer pods than their capacity will increase energy waste. To do this, our proposed autoscaler maintains nodes in a stack such that when a node is added to the stack, all other nodes below are allocated the maximum number of pods and pods are removed from the node at the top of the stack. If all of the currently switched on nodes are fully utilized and we require to add pods, we switch on a new server (a worker node) and start allocating new pods to it. Thereby we increase the number of pods allocated to a node until it cannot host anymore pods; then we add a node to the list. At any given time there may be a worker node with no pods allocated to them. To save energy, we remove these nodes, and we switch off servers with no active nodes. This way we expand/shrink the size of the cluster and make sure nodes are always fully utilized before adding/removing a node to the system.

As we need to iterate through all P pods running on maximum N servers, the time complexity of AutoScaler algorithm is $O(NP)$ which allows for a linear scalability in terms of number of servers and number of pods.

7 MULTI-ARMED BANDIT APPROACH

Our proposed IP and linear regression algorithms require a certain knowledge preliminary to the implementation. In the case of the IP optimization model, we need values like energy consumption and generation at each level and also how many pods are present in each node. Similarly, for the linear regression model, we need to perform system profiling to gather data for the model which is difficult in practice. Due to the mentioned limitations, these solutions cannot be easily adapted to all scales and structures. To solve these problems, we propose an online machine learning model based on the Multi-Armed Bandit (MAB) approach. The model continuously learns and improves its results without relying on *a*

priori knowledge. We model our problem as MAB since it requires no initial knowledge about the system and it is easy to implement.

MAB is a special case of Reinforcement Learning. The original MAB problem [20] was in relation to a problem in a casino with multiple slot machines where a user needed to select the slot machine with the highest chance of a jackpot. The model has multiple slot machine arms to choose from thereby obtaining the name Multi-Armed Bandit. We present our problem as a MAB problem with the agent having to choose among different actions (arms) which affect the QoS of the IoT application and to find the action with the best outcome by looking at the reward after making a decision. We solve the formulated MAB problem using the Upper Confidence Bound (UCB) method, where a reward function returns a value between 0 and 1 [20]. The UCB algorithm updates its exploration-exploitation balance as it gathers more information. Algorithm 3 shows the implementation of our online MAB-based learning approach. Our algorithm makes a decision every time the system becomes stable. The stable condition is achieved when there is no change in the number of pods by the Autoscaler for the last consecutive time slots (4 time slots in our setup). When the system is stable, our MAB-based algorithm computes the reward for the last choice made using a reward function which is discussed later. After receiving the reward, it registers the reward with the choice in the policy and makes the next choice using the UCB method and set the rate of tasks accordingly.

The actions selected by our approach are related to the adjustment of the QoS, meaning the rate at which the fog environment accepts incoming tasks. There are 5 actions in our solution, referring to varying the rate of tasks (QoS): 1) increase rate high, 2) increase rate low, 3) do nothing, 4) decrease rate low, and 5) decrease rate high. All these actions have rewards set between 0 to 1, initially valued at 0.5. The reward function is designed in a way that it forms different rewards for the actions. It depends on the difference between the available renewable energy and the energy consumed and tries to tell which action is best suited. If there is a sufficiently small difference (e_p) between them (available renewable energy and energy consumed), then *do nothing* is given as the highest reward (i.e., 1), and other actions are given a lower reward (0.25), presenting that the *do nothing* as the best choice. Similarly, if the available renewable energy is much greater (4^*e_p) than the energy consumed, then both *increase rate* actions are given higher rewards (1), whereas *do nothing* is given a lower reward (0.25) and both *decrease rate* actions are given a zero reward. Similarly, if the available renewable energy is much lower (4^*e_p) than the energy consumed, then both *decrease rate* actions are given higher rewards (1), whereas *do nothing* is given a lower reward (0.25) and both *increase rate* actions are given a zero reward.

MAB algorithms generally have polynomial time-complexity which provides the highest among all proposed approaches. However, since GreenFog is designed to work in a multi-server small-scale Fog environment, it is safe to assume that the number of nodes in the cluster does not become very large.

Algorithm 3: MAB

Input *minPodLoad, maxPodLoad* **if** *Stable* **then**
  $reward \leftarrow getReward(arm)$
  $policy.setReward(arm, reward)$
  $arm \leftarrow policy.choice()$
  $setRate(arm)$
end
autoscaler(minPodLoad,maxPodLoad)

Table 1: Hardware Information and Quantity

Hardware	Quantity
Dell R710 2RU Server Dual X5570 CPU 4 core 2.93 GHz, 48GB DDR3 ECC	3
RAM 900GB 10K SAS	2
2.5" Bay PERC H700 Raid card with Battery Back Cache	8
870W PSU	2
EMAB22 Eaton Vertical Managed G3 ePDU - 0U (C20 16A 1P), 20XC13, 4XC19, 52 x 53 x 1604, 2.46kg	2

Table 2: Virtual Machine Information

Quantity	5
Flavour	m1.xlarge
Memory	16GB
VCPUs	8
Disk	160GB

8 PERFORMANCE EVALUATION

We have conducted a series of experiments to evaluate the performance of the GreenFog Framework using each of the proposed algorithms in experimental settings on a real-world testbed and with a practical application demonstration. We first present the environmental setup followed by different application settings and finally present some results and discussions.

8.1 Environmental Setup

We utilize a cluster of 3 Dell servers managed by OpenStack in our laboratory at Monash University as the Fog environment. This is a reasonable scale for Fog node.

8.1.1 Hardware. As shown in Table 1, the setup contains 3 physical servers connected to Eaton ePDUs (Electronic Power Distribution Unit) which can be remotely accessed to monitor power consumption of the servers. In our setup, a total of 5 Virtual Machines (VMs) can be hosted out of which one acts as a kubernetes master and others as kubernetes nodes (workers). The master VM runs on the main server alone and the rest are running on the other two physical servers (2 VMs on each physical server). The types of VM are presented in Table 2. We limit the maximum resources (CPU, Memory, etc.) that can be used by each pod. Every VM can accommodate up to 5 pods including the master VM, with a total of 25 pods for all VMs. Apart from the servers, we use a group of three Raspberry Pi Model 3B+ to act as the IoT devices. The Raspberry Pis use the Raspberry Pi Camera Module v2 to take the images and send it to the service end point.

8.1.2 Software. The Raspberry Pis run a client code taking images and sending them to the kubernetes service running on the master node. They also run a flask server with RESTful APIs through which we can set the image rate. The flask server accepts the interval in seconds that images are sent. This way the rate of tasks submitted to the service can be set for the Raspberry Pis. Since there is an application overhead of around 0.3 seconds, the minimum interval of 0.3 seconds can be set. The maximum power consumption of our system is around 1 KW whereas the maximum renewable energy

available in the traces is also roughly 1kWh as anything above will not be utilized by the servers. We have not taken into account the energy needed for cooling or other power consumption sources. We presume that the renewable energy available is only to be utilized for the server power consumption.

We deploy YOLO V3 (You Only Look Once) [25], an object detection technology, as a service in the pods to process the image data being sent by the Raspberry Pis to the end point of Kubernetes service. YOLO is a real time object detection system available as a library in Python. The servers are set up in a way that the master node assigns a new incoming task to a pod with the lowest load. After object detection is performed, the pods send a post request to another server with the detected objects and information about the image in a JSON object. In addition, there is a Gateway software which interacts with the master node and the Raspberry Pis, that evenly sets the rate provided by the master node across all the 3 Raspberry Pis. All of the images being sent by the Raspberry Pis are of the same resolution 1200x600 pixels thereby the only aspect of the QoS that concerned the Fog environment is the rate at which the tasks are sent to the master node.

8.1.3 Renewable Energy Trace. The renewable energy traces used for all the experiments is based on the availability of solar energy for a location of a data center in Lyon, France. We used the data traces by SoDa service with one-hour granularity between the 20th and 21st of September, 2007. The Global Horizontal Irradiance (GHI) for the location is used to calculate the output for the solar photovoltaics (PV) power. We assume that the data center consumes power generated by the PV panels of a total area that generates 1 KW of electricity at peak with a tilt angle of 45^o and PV cell efficiency of 30%. For more details check our previous paper in [Authors].[3]

8.2 Linear Regression

The linear regression model required a profiling for different rates of images while the Autoscaler changes the number of pods according to the average number of pending tasks. The lower threshold is set at 10 meaning that if the average number of pending tasks in the pods goes below 10, the scaler reduces one pod from the deployment. Similarly, the upper threshold is set at 15, where the scaler increases the deployment by one pod if the average number of pending jobs becomes greater than the upper threshold.

The profiling results are shown in Table 3. Our aim is to identify the power consumption for different rates we set. For each of the rates, we wait until the Fog environment becomes stable thereby giving stable and appropriate power consumption values. After collecting the data shown in Table 3, we are able to form a linear

[3]The citation is hidden for the sake of double blind review.

Table 3: Profiling Data

Total Power (W)	Power Per Task (W)	Images/Second
258.8	595.24	0.4348
261.4	339.82	0.76923
472	377.6	1.25
586.9	322.795	1.82
805.4	342.295	2.353
974.9	353.40125	2.76

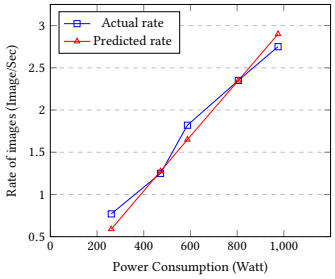

Figure 3: Regression model predictions vs actual values

regression model where the power consumed by the Fog environment is the explanatory variable and the rate (Images per Second) is the dependent variable to get α (y-intercept) as -0.2535 and β (slope) as 0.00324 for the equation.

Fig. 3 shows how well the regression model predicts the power consumption compared to the actual data. Note that the power consumption (not energy) per task listed in Table 3. The image processing task executed per image is a heavy process and requires significant amount of time and resources leading to power consumption values reported in Fig. 3. Therefore, to keep the queue length and response time for tasks the number of pods needs to adjusted in the system.

We ran the Linear Regression model with the Autoscaler utilizing the renewable energy traces in which we translated every hour in the actual trace into a 15-minute interval in the experiment. This is to perform shorter experiments with more data points. The code reads the available green energy generation for the appropriate time and sets the rate for IoT devices based on the above equation. With limited capacity (the maximum number of pods is limited), we profiled the maximum rate that can be set. Similarly, there is a minimum rate when only the master node is ON with only one pod.

Therefore, the framework shuts down any VM not currently used and also shutdowns any server with no VM running. As the master VM always needs to be running, a minimum power consumption of around 250-300 Watts is reported. When a server is shut down, it still consumes a base power which is around 20-30 Watts.

As we can see in Fig. 4, the actual power consumption increases and decreases according to the renewable energy availability. The total absolute energy difference between the available renewable energy and the energy consumed in the 24-hour test is 4.886 kWh (this is shown by the shaded area in the figure). As stated earlier, we scale down the experiments to 6 hours by assuming every hour in the renewable energy traces to be about 15 minutes in real life. However, we report results based on the 24-hour scale. The duration of 15 minutes was chosen, as booting up/down a server takes at least 10-12 minutes, thereby we need at least 15 minutes to ensure the system becomes stable at around 15 minutes before the next decision is made. The sudden spikes and dips seen in Fig. 4 deviating from the renewable energy are due to how AutoScaler performs. As it increases the number of pods linearly (1 after another), it might take some time to get to the required number of pods needed to handle the incoming tasks. Due to this, the average pending tasks in each pod could fluctuate to extreme limits. Therefore, the Autoscaler could overshoot or undershoot the number of pods needed causing

such downward or upward spikes. In addition, a node shutdown might take 10-12 minutes, adding a huge latency to reach stability.

8.3 IP Model with Linear Autoscaling

We use the IP model from Section 4 with the default Linear Autoscaler from Section 6. The IP model requires certain information about the system such as energy consumed per pod, energy consumed by the servers when in shutdown or running, energy needed to service a task, and the processing time it takes for a task. To obtain the energy consumed per pod, we ran the system with one VM (master) with 2 pods running and then, we scaled the deployment by adding another pod to the same VM to record the rise in energy consumption. For energy consumed by the servers, we first switched off the server and recorded the power consumption then we switched on the system and noted the difference in the power consumption. To figure out the energy needed to service a task, we ran the Fog environment with Autoscaler from Section 6 and then we sent one image every second and waited until the cluster becomes stable and noted down the power consumption; from this value we subtracted the values obtained previously like the energy per pod and the energy per node in On/Off state appropriately to get the energy per task. Similarly, to calculate the processing time a task can take, we ran our Fog environment setup with Autoscaler and the rate of one task per second and noted down the number of pods needed to cater to these requirements.

If such information is known, this model can provide the number of servers that should be running, how many pods in each server are required, and also the acceptable rate of jobs which helps in setting QoS. In this experiment, we only use the rate of tasks and ignore scaling information that the IP model produces. For scaling, we use the Linear Autoscaler to automatically allocate resources. We tested the IP model in this environment with the same solar traces we used in the previous experiment. The results are shown in Fig. 5 illustrating that IP model with Linear Autoscaler has 16% less absolute energy consumption difference compared to the Linear Regression model for the same data by being 4.095 kWh compared to the 4.886 kWh, respectively. Results shows the impact of possessing future knowledge and detailed information regarding task and pod energy consumption by the IP model while a linear autoscaler is used. In the next subsection, we investigate the performance of IP model if scaling information that the IP model produces is used instead of linear autoscaling.

8.4 IP with Direct Scaler

Fig. 6 shows the results of the experiments for the IP model from Section 4 with a new scaler, called *Direct Scaler*, which utilizes all the decision variables from the IP model. The Direct Scaler receives the information owing to which nodes are to be switched On/Off, and the number of pods that should be running in each node from the IP model. The IP model also sets the rate of images (QoS) that should be sent to the gateway software. Fig. 6 shows a considerably smooth power consumption pattern with a lower number of spikes compared to the results in Subsections 8.3 and 8.2. The total absolute difference between the green energy and the actual consumed energy is around 3.677 kWh. This approach performs significantly better than the Linear Regression and the IP model with Linear Autoscaler. Results illustrate that IP model with Direct Scaler has 25% less absolute energy consumption difference compared to the Linear Regression model for the same data by being 3.677 kWh compared to the 4.886 kWh, respectively. Similarly, it is 10% less compared to the 4.095 kWh for the IP model with linear autoscaler. Note that IP with direct autoscaler can be used as the baseline for the comparison with other approaches as it has future knowledge and detailed information of system.

8.5 MAB model with Linear Autoscaler

Fig. 7 depicts power consumption for the GreenFog Framework, where the MAB model from Section 7 with the Linear Autoscaler mentioned in Section 6 is used. We conducted two days experiments, where the first day would provide MAP model the opportunity to learn the system behaviour. The renewable energy data used is exactly the same across 2 days and matches with the data used with other approaches to help make comparisons. In Fig. 7, we can see that the MAB model tries to match the energy consumed with the renewable energy on the first day and improves in the second day as time passes. The total absolute energy difference between the available renewable energy and the energy consumed in the second day is 4.164 kWh (shown by the shaded area in the figure). This is 14% better than the linear regression model about the same as the IP model with Linear Autoscaler. However, as we expect, the baseline IP model with Direct Scaler outperform this model by 11% less total absolute energy difference. The results show how the MAB model performs very well compared to other approaches if sufficient and proper data for training is provided.

8.6 Comparison of Response Times

Since response time is of critical concern in any Fog environment, we evaluated the response time for different optimization models in the GreenFog framework. Fig. 8 shows the CDF of response time of IoT requests for different optimization models. In Fig. 8, we can see that 90% of requests are served in less than 29 seconds when the IP model with Direct Scaler is used. However, the IP model with Linear Autoscaler reaches the same stats at 404 seconds. The linear regression and MAB models with Linear Autoscaler serve 90% of requests at 348 and 241 seconds, respectively. The significant difference between the response time of requests of different models is due to the type of autoscaler they use. The Linear Autoscaler maintains a lower threshold for the number of pending requests in the queue of each pod, whereas the direct scaler does not. Among the models that used the Linear Autoscaler, MAB provides the best response time; whereas, the linear regression performs slightly better than the IP model.

9 RELATED WORK

The energy consumption of data centers is set to account for 3.2 percent of the total worldwide carbon emissions by 2025 [4]. If we add the energy use by all the intermediary nodes and routers involved to transfer data from edge devices to the cloud coupled with the energy of the edge devices themselves, the figure is extremely intimidating. Hittinger and Jaramillo [14] discussed how scientists and regulators need to address this energy issue to ensure that the benefits of IoT do not come at the expense of rising energy.

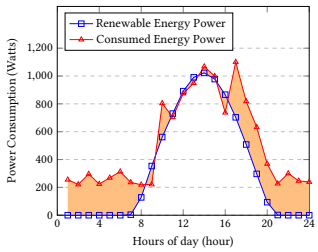

Figure 4: Linear Regression with Linear AutoScaler

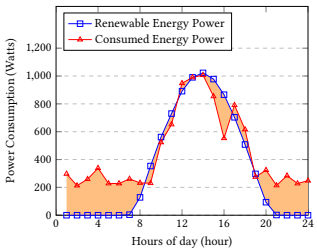

Figure 5: IP with Linear Autoscaler

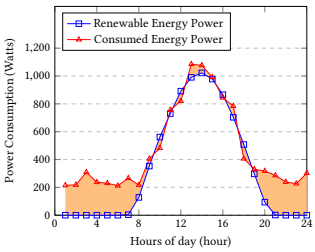

Figure 6: IP with Direct Scaler

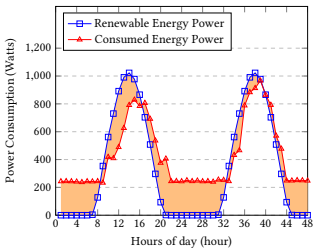

Figure 7: MAB with Linear Autoscaler

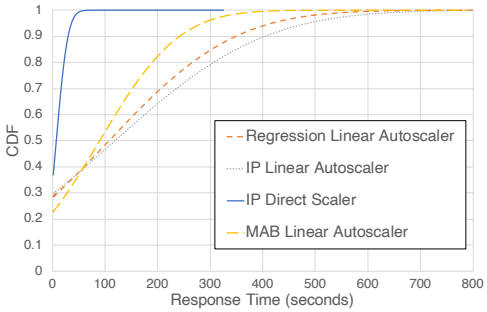

Figure 8: CDF of Response Time for Different Optimization Techniques

To help utilize both cloud technologies and edge computing devices, a good bridge is needed which is provided by the introduction of Fog computing as stated in [30]. Fog computing provides micro centers closer to the edge devices that helps reduce the usage of the network bandwidth by the partial offloading of workloads from the cloud to the edge which in turn leads to saving energy.

A study conducted in [15] shows that the Fog environment can reduce energy consumption compared to the centralized system like the cloud. Similarly, Farooqi *et al.* [8] explained how Fog computing is essential to reduce the overall energy usage. However, to make Fog computing sustainable, many researchers have proposed the usage of green energy to power Fog micro data centers [31] [18] [17] [8] [26] [16].

To utilize the green energy most efficiently and to further help the usage of renewable energy for the Fog computing environments, Li *et al.* [21] presented approaches to help improve the usage of green energy. However, contrary to our framework, they do not consider QoS adjustment in their proposed approaches. Goudarzi *et al.* [12] and Mahmud *et al.* [22] suggested finding the optimal placement of tasks for Fog computing environments, so as to maximize utilization of green energy sources like Solar, Wind, and Hydro. They focused on meeting QoS requirements of the IoT applications, while they did not consider the adjustment of QoS for IoT applications similar to our work.

Gu *et al.* [13] presented an approach to maximize green energy utilization using an optimal task allocation system which satisfies the required QoS according to its availability in these Fog nodes. They proved the NP-hardness of the problem of the joint consideration of VM migration, task allocation, and green energy scheduling. While we proposed an optimal solution along with regression-based heuristic and machine learning approaches based on QoS adjustment, they tackled the computation complexity using a heuristic algorithm approximating the optimal solution. In addition, they do not consider QoS adjustment as we do in this work. Toor *et al.* [26] and Karimiafshar *et al.* [17] proposed techniques based on dynamic voltage and frequency scaling to match energy consumption of cloud data centers with renewable energy availability. Their approach differs from ours as we focus on workload shaping based on QoS adjustment while they focus on hardware techniques.

In a study performed by Zhang *et al.* [32], the energy harvesting in the edge devices is optimized subject to QoS constraints. Adding to this, Xu *et al.* [29] reduced the energy usage from the grids using brownout techniques and switching off the servers with no load. Their approach is similar to our methods but it is designed for cloud environments. They also focused on temporal load shifting which is not possible for Fog environments and the IoT ecosystem.

Mahmud et al. [23] and Karimiafshar *et al.* [16], in their research tells us about a docker containerization fog environment with a custom scaler, scheduler and dispatcher which can be used to maximise Green energy usage or to maximise Response time such that we can always find a balance depending on the task being performed by the Fog environment. Even though they tell about the opportunity to be able to customise the scaler and dispatcher to meet the needs we use these techniques to further this by showing how QoS can be chosen to counter the usage of green energy and all the different ways this can be done and their results.

Goiri *et al.* [10], tells on ways the fog environment can predict the solar or green energy capacity and make decisions early, We use this approach to add to our Optimization model such that we the fog environments can act early and utilize the green energy more efficiently. In summary, our proposed methods are different from the existing approaches, as focusing on reactive QoS adjustment of IoT application to optimize renewable energy use for Fog environments.

10 DISCUSSIONS

This paper presents GreenFog, a framework for building a sustainable fog computing platform based on QoS adjustment. GreenFog targets the deployment of edge/fog platforms that have a source of renewable energy such as solar and focuses on applications where the QoS can be flexibly adjusted based on data quality (e.g., image processing). Other examples of applications that fit the proposed framework include, but are not limited to, surveillance video analytics or audio signal processing.

The main aim of this work is to introduce a novel framework for building sustainable fog computing environments. The main

strength of our work is that we developed the entire software system and tested it using a real-world testbed (Fog node, Raspberry Pis, cameras), applications (Yolo), and traces (Solar data). However, working with real systems and relying on actual measurements limit the scope for conducting extensive and various experiments possible in simulations. For example, the evaluation is limited to two days as experiments are happening in real-time. We are also bounded by the scale limitations of equipment in an academic research laboratory. Considering these limitations, we showed that GreenFog provides a promising approach to build sustainable Fog environments, and the proposed optimization models can efficiently match power consumption and renewable power generation.

The main goal of our paper is to represent the possibility of optimizing green energy use through adaptive QoS adjustments. Therefore, we focused on methods to set optimized QoS levels and relied on a simple widely used threshold-based auto-scaling for the right-sizing of resources. Experimental results show that the autoscaling approach significantly impacts the overall system performance. Therefore, the design and evaluation of more advanced autoscaling techniques for the GreenFog framework represent a promising future direction.

11 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed GreenFog, a framework to optimize green energy use for Fog environments with onsite renewable electricity generation. We designed and implemented an optimization model to minimize the usage of brown energy and maximize the utilization of the available green energy by adjusting QoS for the IoT application. This model requires exact calculations of the energy requirements of different components of the Fog environment that would be hard to collect in practice. We proposed a heuristic based on a Linear Regression approach to overcome such a challenge while attaining a comparatively similar result. To further help the cause of the Linear Regression approach, a threshold-based Linear Autoscaler was also proposed. Since the performance of the Linear Regression model relies on accurate profiling, we proposed a machine learning model based on the Multi-Armed Bandits problem that removes the need for in advance knowledge of the system workload and energy consumption model. We evaluated and validated the GreenFog framework equipped with different optimization techniques and approaches on a real testbed. We measured the actual power usage and response times of requests with a practical IoT application and realistic traces of renewable energy. In the future, we are interested in developing a model to work based on forecasting future renewable energy availability. We further look into solving this problem with other machine learning methods such as deep reinforcement learning to improve the efficiency of the proposed framework.

REFERENCES

[1] Mohammad Aazam, Sherali Zeadally, and Khaled A. Harras. 2018. Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities. *Future Generation Computer Systems* 87 (2018), 278–289.

[2] Tanweer Alam. 2018. A reliable communication framework and its use in internet of things (IoT). *CSEIT1835111| Received* 10 (2018), 450–456.

[3] Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodík, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath, and Sudipta Sinha. 2017. Real-time video analytics: The killer app for edge computing. *computer* 50, 10 (2017), 58–67.

[4] Anders Andrae. 2017. Total consumer power consumption forecast. *Nordic Digital Business Summit* 10 (2017).

[5] Piotr Borylo, Artur Lason, Jacek Rzasa, Andrzej Szymanski, and Andrzej Jajszczyk. 2016. Energy-aware fog and cloud interplay supported by wide area software defined networking. In *2016 IEEE International Conference on Communications (ICC)*. 1–7. https://doi.org/10.1109/ICC.2016.7511451

[6] X. Chang, W. Li, C. Xia, J. Ma, J. Cao, S. U. Khan, and A. Y. Zomaya. 2018. From Insight to Impact: Building a Sustainable Edge Computing Platform for Smart Homes. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. 928–936. https://doi.org/10.1109/PADSW.2018.8644647

[7] Marco Cococcioni and Lorenzo Fiaschi. 2021. The Big-M method with the numerical infinite M. *Optimization Letters* 15, 7 (2021), 2455–2468.

[8] A. M. Farooqi, S. I. Hassan, and M. A. Alam. 2019. Sustainability and Fog Computing: Applications, Advantages and Challenges. In *2019 3rd International Conference on Computing and Communications Technologies (ICCCT)*. 18–23. https://doi.org/10.1109/ICCCT2.2019.8824983

[9] Íñigo Goiri, William Katsak, Kien Le, Thu D. Nguyen, and Ricardo Bianchini. 2013. Parasol and GreenSwitch: Managing Datacenters Powered by Renewable Energy *(ASPLOS '13)*. Association for Computing Machinery, New York, NY, USA, 51–64. https://doi.org/10.1145/2451116.2451123

[10] Íñigo Goiri, Kien Le, Md. E. Haque, Ryan Beauchea, Thu D. Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. 2011. GreenSlot: Scheduling Energy Consumption in Green Datacenters. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*. Association for Computing Machinery, New York, NY, USA, Article 20, 11 pages. https://doi.org/10.1145/2063384.2063411

[11] Íñigo Goiri, Kien Le, Thu D. Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. 2012. GreenHadoop: Leveraging Green Energy in Data-Processing Frameworks. In *Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys '12)*. Association for Computing Machinery, New York, NY, USA, 57–70. https://doi.org/10.1145/2168836.2168843

[12] M. Goudarzi, H. Wu, M. S. Palaniswami, and R. Buyya. 2020. An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments. *IEEE Transactions on Mobile Computing* (2020), 1–1. https://doi.org/10.1109/TMC.2020.2967041

[13] Lin Gu, Jingjing Cai, Deze Zeng, Yu Zhang, Hai Jin, and Weiqi Dai. 2019. Energy efficient task allocation and energy scheduling in green energy powered edge computing. *Future Generation Computer Systems* 95 (2019), 89–99.

[14] Eric Hittinger and Paulina Jaramillo. 2019. Internet of Things: Energy boon or bane? *Science* 364, 6438 (2019), 326–328.

[15] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker. 2016. Fog Computing May Help to Save Energy in Cloud Computing. *IEEE Journal on Selected Areas in Communications* 34, 5 (2016), 1728–1739. https://doi.org/10.1109/JSAC.2016.2545559

[16] Aref Karimiafshar, Massoud Reza Hashemi, Mohammad Reza Heidarpour, and Adel N. Toosi. 2021. An Energy-Conservative Dispatcher for Fog-Enabled IIoT systems: When Stability and Timeliness Matter. *IEEE Transactions on Services Computing* (2021), 1–1. https://doi.org/10.1109/TSC.2021.3114964

[17] Aref Karimiafshar, Massoud Reza Hashemi, Mohammad Reza Heidarpour, and Adel N. Toosi. 2020. Effective Utilization of Renewable Energy Sources in Fog Computing Environment via Frequency and Modulation Level Scaling. *IEEE Internet of Things Journal* 7, 11 (2020), 10912–10921. https://doi.org/10.1109/JIOT.2020.2993276

[18] Aref Karimiafshar, Massoud Reza Hashemi, Mohammad Reza Heidarpour, and Adel N. Toosi. 2021. A request dispatching method for efficient use of renewable energy in fog computing environments. *Future Generation Computer Systems* 114 (2021), 631–646.

[19] Dominique Larcher and Jean-Marie Tarascon. 2015. Towards greener and more sustainable batteries for electrical energy storage. *Nature chemistry* 7, 1 (2015), 19–29.

[20] Tor Lattimore and Szepesvari Csaba. 2020. *Bandit algorithms.* Cambridge University Press.

[21] W. Li, T. Yang, F. C. Delicato, P. F. Pires, Z. Tari, S. U. Khan, and A. Y. Zomaya. 2018. On Enabling Sustainable Edge Computing with Renewable Energy Resources. *IEEE Communications Magazine* 56, 5 (2018), 94–101. https://doi.org/10.1109/MCOM.2018.1700888

[22] Redowan Mahmud, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2020. Profit-aware application placement for integrated Fog–Cloud computing environments. *J. Parallel and Distrib. Comput.* 135 (2020), 177–190.

[23] Redowan Mahmud and Adel N. Toosi. 2021. Con-Pi: A Distributed Container-based Edge and Fog Computing Framework. *IEEE Internet of Things Journal* (2021), 1–1. https://doi.org/10.1109/JIOT.2021.3103053

[24] Redowan Mahmud and Adel N. Toosi. 2022. Con-Pi: A Distributed Container-Based Edge and Fog Computing Framework. *IEEE Internet of Things Journal* 9, 6 (2022), 4125–4138. https://doi.org/10.1109/JIOT.2021.3103053

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[26] Asfa Toor, Saif ul Islam, Nimra Sohail, Adnan Akhunzada, Jalil Boudjadar, Hasan Ali Khattak, Ikram Ud Din, and Joel J.P.C. Rodrigues. 2019. Energy and performance aware fog computing: A case of DVFS and green renewable energy. *Future Generation Computer Systems* 101 (2019), 1112–1121.

[27] Ted Trainer. 2017. Some problems in storing renewable energy. *Energy Policy* 110 (2017), 386–393.

[28] Xailient. 2019. CARBON IMPACT. Available: https://www.xailient.com/technical-details.

[29] Minxian Xu, Adel N Toosi, and Rajkumar Buyya. 2020. A Self-adaptive Approach for Managing Applications and Harnessing Renewable Energy for Sustainable Cloud Computing. *IEEE Transactions on Sustainable Computing* (2020).

[30] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. 2019. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture* 98 (2019), 289–330.

[31] Deze Zeng, Lin Gu, and Hong Yao. 2020. Towards energy efficient service composition in green energy powered Cyber–Physical Fog Systems. *Future Generation Computer Systems* 105 (2020), 757–765.

[32] G. Zhang, Y. Chen, Z. Shen, and L. Wang. 2019. Distributed Energy Management for Multiuser Mobile-Edge Computing Systems With Energy Harvesting Devices and QoS Constraints. *IEEE Internet of Things Journal* 6, 3 (2019), 4035–4048. https://doi.org/10.1109/JIOT.2018.2875909

[33] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. 2017. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 377–392. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/zhang