This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2939534, IEEE Internet of Things Journal

KEIDS: Kubernetes based Energy and Interference Driven Scheduler for Industrial IoT in Edge-Cloud Ecosystem

Kuljeet Kaur, *Member, IEEE*, Sahil Garg, *Member, IEEE*, Georges Kaddoum, *Member, IEEE*, Syed Hassan Ahmed, *Senior Member, IEEE*, Mohammed Atiquzzaman, *Senior Member, IEEE*

Abstract— With the rapid explosion of Industrial Internet of Things (IIoT), the need for real-time data processing with enhanced flexibility and scalability has increased manifold. However, the newly evolved containerization technology offers lucrative advantages in comparison to the conventional virtual machines. However, management of these light-weight containers is a tedious task, but Google Kubernetes offers a consolidated container management and scheduling for successful execution of various lightweight containers. Nevertheless, the existing Kubernetes solutions fall short in efficiently handling the *"Interference"* and *"Energy minimization"* challenges in IIoT set-up. Hence, in this paper, we present a competent controller, named *KEIDS*, for container management on edge-cloud nodes taking into account the emission of carbon footprints, interference, and energy consumption. The problem of task scheduling has been formulated using *Integer Linear Programming* based on multi objective optimization problem. In detail, *KEIDS* minimizes the energy utilization of edge-cloud nodes in IIoT for optimal green energy utilization. Henceforth, the applications are scheduled on the available nodes in less time with minimum interference from other applications, which in turn guarantees an optimal performance to the end-users. An extensive evaluation of the proposed *KEIDS* scheduler in comparison to the existing state-of-the-art schemes indicates its superior performance on real-time data acquired from Google compute cluster.

Index Terms—Cloud Computing, Edge Computing, energy minimization, Industrial Internet of Things, job scheduling, Kubernetes and multi-objective optimization

1 Introduction

With the rapid development of information technology, the Industrial Internet of Things (IIoT) has become an important component of industrial systems. It is a new ecosystem that connects billions of devices such as intelligent and autonomous machines, smartphones, wearables, industrial equipments, etc. in Industry 4.0 context. It can improve the connectivity, efficiency, and scalability by integrating physical objects, cyber objects, and social objects in the industry [1]. As a result, these technologies have made a huge impact on a large number of smart applications such as transportation systems, healthcare systems, manufacturing systems, smart grids (SG), etc [2], [3]. Therefore, the number of IoT devices is increasing dramatically which generates a substantial amount of industrial data [4], [5]. According to the Cisco Global Cloud Index report, the data produced by connected things, machines and people will reach 500 zettabytes by the end of 2019 [6]. Further, the applications running on these devices are also becoming complex and computationally intensive thereby demanding unlimited resources. In order to cater this problem, the majority of the IoT products and services are supported by cloud platforms, which have larger computing capacity [7]. However, offloading all the tasks to the remote location can put significant burden on the network and cause important latency and bandwidth related challenges. Thus, these platforms are not suitable for IIoT applications that often demand services such as self-monitoring, self-diagnosing, self-healing, self-directing, etc [8], [9].

In order to tackle this challenge, an emerging computing paradigm, called edge computing has emerged [10]. Generally, it is a geographically distributed computing architecture in which heterogeneous devices are ubiquitously connected. In this computing paradigm, an edge device resides between the data sources and cloud-based data centers (DCs) to support both downstream (in context to cloud services) and upstream data (in context to IoT services) [11]. Thus, network-intensive data can be processed one hop away from end-devices, which reduces the bandwidth demands and high communication latency between IoT devices and the cloud [12]. Further, offloading the tasks to the edge also incur fewer costs as compared to offloading the tasks to the cloud [13]. Consequently, edge-computing platforms also facilitate the host of interactive applications that support user mobility [14]. Thus, the majority of technology giants such as IBM and Cisco have already started to push the computing capabilities to the edge of networks. In this direction, Ericsson Inc. predicted that more than 50 billion devices will be connected to the Internet by the year 2025 and most of these devices will be located at the edge of Internet [15]. Some devices that are already available include

- *K. Kaur, S. Garg, and G. Kaddoum are with the Electrical Engineering Department, École de technologie supérieure, Université du Québec, Montréal, QC H3C 1K3, Canada. (e-mail: kuljeet.kaur@ieee.org, sahil.garg@ieee.org and georges.kaddoum@etsmtl.ca)*
- *S.H. Ahmed is with the Department of Computer Science, Georgia Southern University, Statesboro, GA 30460, USA. (e-mail: sh.ahmed@ieee.org)*
- *M. Atiquzzaman is with the School of Computer Science, University of Oklahoma, Norman, OK 73019-6151, USA. (e-mail: atiq@ou.edu)*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2939534, IEEE Internet of Things Journal

Apple watches, Fitbit sports trackers, Google Glasses and Oculus Rift helmets [16].

Nonetheless, these smart devices demand real-time analysis and storage in order to provide scalable and flexible services. The most sought after technique in this regard is offered by the Cloud's Platform-as-a-Service (PaaS) model; wherein the resources are used by the intended users as and when required. In PaaS, the concept of virtualization is extensively exploited to offer real-time services. However, with the progressive evolution, the concept of containerization has recently emerged as a powerful paradigm to process streaming Big Data from IIoT ecosystems [17], [18]. Additionally, with the evolution of virtualization to the light weight containers, the deployment of micro-services/applications has become quite easy, however this comes at the cost of increased granularity of the deployed applications [19]. With this, the number of manageable components has also escalated, so has the need to effectively manage the underlying containers in terms of deployment, mapping, inter networking, resource allocation, etc. Fortunately, Google Kubernetes (popularly referred as K8S) has come to rescue by constantly improving container management [20].

Kubernetes is an open source platform which is optimized for all infrastructures and applications due to its inherent ability for portability and extensibility [21]. It also enables automated deployment, scaling and management of containerized applications. It is based on a master-slave architecture where the communication between the master and slaves is done using a kubelet device [22]. Since this simplifies data-center operations, several industries now use this technique as their core cloud deployment technology [20]. Examples include: Google for its Google Cloud Engine (GCE), Microsoft for Microsoft Azure, IBM for Softlayer and OpenStack, Red Hat for its OpenStack distribution, etc.

1.1 Motivation

Despite the progress made by the academia and industry with respect to container management and the wide-scale acceptance of Kubernetes, the following inferences can be drawn. First of all, the deployment of Kubernetes in Edge-Cloud environments is still in its infancy and needs a well justified approach to tackle the challenges associated to it. In this direction, *interference* and *energy minimization* are considered as important research challenges by the cloud computing fraternity. The existing literature presents some research proposals highlighting the issues caused by interference in the context of Kubernetes [23], but a concrete solution is still lacking. Furthermore, the concept of energy minimization has not been explored by the research community in the context of Kubernetes. Thus, the exploration of interference and energy minimization with peculiar focus on green energy utilization [24], is a first attempt in the context of the emerging edge-cloud environment equipped with Kubernetes. Thus, motivated by the above mentioned reasons, we design a comprehensive resource management and job synchronization model for Kubernetes, that can be deployed on IIoT set-ups. The designed model is initially formulated as a multi-objective optimization problem with primary focus on minimizing the energy consumption of computing devices (deployed at the edge and the cloud). Additionally, it also aims to achieve the best possible use of available green energy resources; while simultaneously achieving effective resource management and job synchronization with minimal interference among co-located containers on the same or different nodes.

1.2 Contributions

The major contributions of the proposed work are highlighted below:

1) We present a scalable and comprehensive controller for Kubernetes, *KEIDS*, in edge-cloud environments for IIoT with the view to minimize the carbon footprint rate while enhancing the performance and energy-efficiency.
2) We formulate a multi-objective optimization problem for efficient scheduling/mapping of containers to the available nodes in multi-constraint Kubernetes set-ups. The designed scheme considers three competing functions, *i.e.,* energy minimization with reduced carbon footprint emission and minimal performance interference; and is expressed as an integer linear programming (ILP) problem.
3) We evaluate the performance of the proposed scheme and compare it with the other competing schemes using real-time data traces acquired from Google Cluster on Mosek solver [25].

1.3 Organization

The rest of the paper is structured as follows. Section 3 presents the system architecture and the operation of the proposed *KEIDS* scheduler. The problem formulation with the objective functions are presented in Section 4. Section 5 illustrates the performance evaluation of the proposed scheduler using real-time data traces. Finally, the paper is concluded in Section 6 along with future scope.

2 Related Work

Some of the significant contributions made by the research fraternity in the context of Kubernetes, scheduling strategies, and green energy utilization are illustrated as under.

In Kubernetes, a built-in mechanism is provided for dynamic resource provisioning. However, the resource-provisioning algorithm of this mechanism only considers CPU utilization to determine its provisioning strategy and therefore it is not very effective. Further, in a container-based environment such as cloud/edge, it is essential to monitor the resource requirements and manage the resources accordingly. Thus, in order to overcome this issue, Chang *et al.* [26] developed a dynamic resource-provisioning platform for cloud based on Kubernetes. This platform contains three important features, namely a comprehensive monitoring mechanism, a deployment flexibility and an automatic operation. The experiments in this work were conducted on an open source cloud application platform, *i.e.,* OpenShift Origin using an online ticketing demo application named Ticket Monster. However, these experiments suggested that their proposed platform incurs high performance in terms

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2939534, IEEE Internet of Things Journal

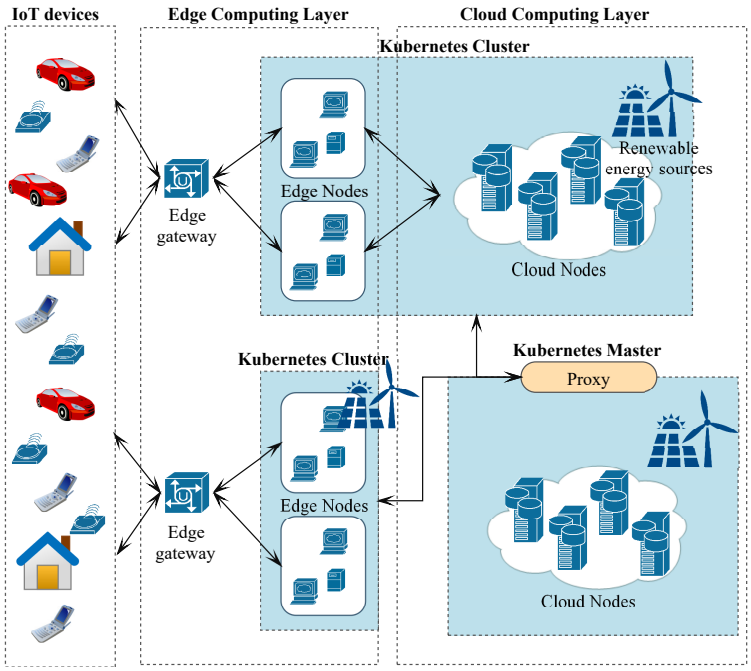


Fig. 1: *KEIDS*'s system model for Kubernetes deployment in IoT ecosystem equipped with Edge-Cloud functionality.

of computing-resource usage and network response time. On a similar line, the orchestration concept based on Kubernetes has been extended to the fog computing platform by Wobker *et al.* in [27]. The designed framework addresses the application deployment challenges in fog set-ups which are characterised by distribution, connectivity, availability, heterogeneity, and real-time needs [27]. In [28], authors identified the importance of load balancing in fog computing setups to reduce service response time and better the overall resource utilization levels. In this direction, the authors devised an efficient approach to identify the less loaded compute servers for load distribution and task allocation.

However, one of the significant challenges associated with Kubernetes is *"Interference"* which is generally caused by co-allocating multiple workloads on a single node. The interference caused by the limited isolation in shared resources can disrupt the capabilities of cloud-based deployments. Hence, to better characterize such performance issues, Medel *et al.* [23] developed a reference net-based model of resource management within Kubernetes. Furthermore, the authors suggested that an interference oblivious scheduler for Kubernetes can have an impact on the overall performance of application workloads. Such scheduler may slow down the workloads by almost 34%, and even up to 2 times in some cases. This may also arise for the applications that are executed on different cores, if they either share the memory, cache or bandwidth.

In [29], Medel *et al.* analyzed the performance of Kubernetes using a Petri net-based performance model. The associated analysis depicted that the overheads incurred during the deployment, termination and maintenance of resources; which in turn have a significant impact on the performance of Kubernetes and the underlying Cloud environment. In [30], Kumar *et al.* proposed a workflow scheduling mechanism as an important technique to keep the performance of the underlying systems in check. The proposed scheduling approach exploited the the parent-child relationship data to intelligent map the incoming task to the available array of virtual machines (VMs). The major attention of the said approach has been reducing tasks' make span and increasing the cumulative resource utilization. In this regard, the energy consumption utilization (which consumes a significant portion of the DC's operational cost) of the computing nodes can be reduced considerably. Working in this direction, Mishra *et al.* [31] also focused on optimum energy utilization by the fog computing infrastructure. In this work, authors identified service request allocation on the available VMs as a potential solution to minimize energy utilization levels. However, the said problem in fog computing setup is a NP-hard problem; thus, the authors formulated their problem as a bi-objective minimization problem with respecting to energy minimization and makespan. The formulated problem was then solved using a metaheuristic-based service allocation framework based on different evolutionary algorithms to attain the near-optimal solution. According to [18], workload co-scheduling is an important prerequisite. Precisely, the overall energy utilization by cloud/edge computing devices can be reduced if workload consolidation is performed in an effective manner; such that the minimum number of nodes are used for application processing and the rest are turned off. Such an initiative is the need of the hour, since the modern DCs are not energy-proportional and tend to consume huge amounts of energy even during non-peak hours.

Another thread of research has focused on maximizing the green energy utilization. For instance, Fan *et al.* [32] proposed an energy-aware VM migration policy, typically focusing on maximizing the green energy of cloudlet networks while reducing the energy consumption from the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2939534, IEEE Internet of Things Journal

grids. The proposed scheme focus not only on the spatial and temporal characteristics of energy consumptions but also on the generation of clean energy. Working in the similar direction, Kiani and Ansari [33] also focused their research on optimum utilization of green energy with minimal brown energy consumption. In this context, the authors proposed an effective workload distribution algorithm for cloud DCs. Yeganeh *et al.* [34] devised a capacity planning approach for green DCs. The primary focus of this work has been on reducing the overall energy utilization and attaining the optimal trade-off between the operational expenses and service delay. In [35], Hasan *et al.* propounded the concept of Green Service Level Agreement (GreenSLA) based on the virtualization of green energy. The core attention of this work was on maximizing the greeness of the available clean energy for cloud DCs such that their respective SLAs are assured. Authors in [36] extensively analysed the current strategies to make IoT ecosystem green. In addition to this, authors also proposed different techniques to make the IoT ecosystem more greener worldwide by the end of 2020.

3 System Model

This section illustrates the architectural model of the proposed scheduler in multi-constraint set-ups.

3.1 Kubernetes in Edge-Cloud set-up for IIoT

The overall system model of the proposed Kubernetes deployment in edge-cloud set-ups is illustrated in Fig. 1. As detailed in the figure, the IP-enabled devices tend to access different services of the IIoT ecosystem through edge gateways. These IP-enabled devices essentially play the role of both the source and the sink in IoT ecosystems; as *"source"* by sensing the immediate environment and relaying the data to the computational platform, and as *"sink"* by executing the commands received from the computational environment. In the current context, computational units rest on the edge-cloud infrastructure which provides various services to its end users.

To provide these real-time services to the intended users, the proposed work considers the integration of the edge computing platform with the standard cloud computing platform. Here, edge can be understood as an extension of the cloud that tends to provide intended services to its users with reduced latency and improved bandwidth utilization. Edge brings the computational resources closer to its users but is limited by its reduced memory and computational power. On the other hand, cloud is a much more powerful resource provider but it is distantly located. Both the edge and the cloud nodes are assumed to deployed with containers for application deployment and execution. Further, these nodes are assumed to connected with renewable energy sources (RES) such as wind and solar energy to cater their operational needs.

The management of container in the current context is catered by Kubernetes which is based on a master-slave architecture where the communication between the master and the slaves is done using the kubelet device [22]. Here, a master node runs an Application Programming Interface (API) server to provide an entry point to the cluster. Moreover, it also uses the same API as a proxy to exhibit its

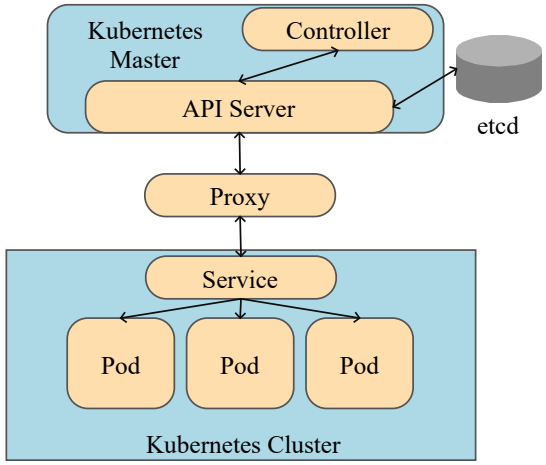


Fig. 2: Different functional components of Kubernetes.

services to external users as depicted in Fig. 2. It schedules containers to use the resources provided by Google Cloud Platform with an emphasis on supporting clusters of machines. In current set-ups, the master is essentially a cloud computing node; while the slaves can be different combinations of edge and cloud computing nodes as depicted using Fig. 1.

So, Kubernetes can be viewed as a comprehensive container scheduling and management framework. A typical Kubernetes set-up, contains C number of clusters with 'N' number of nodes (either physical or virtual). The nodes encapsulate 'P' pods; which in turn are comprised of 'Co' containers. Out of these entities, the conceptual working unit of Kubernetes is a "Pod". Hence, pods are mapped to nodes; which makes nodes more specific during monitoring and execution. However, these nodes may have various problems such as bad CPU, memory, and disk that could corrupt the file-system. In such scenarios, the node problem detector "daemon set" comes to rescue; which is used to find the faulty and misbehaving nodes.

The basic functionality of Kubernetes controller is to keep track of newly created or restarted pods; and to map the pods to suitable node(s). In summary, it involves the mapping of pods' containers to the available array of nodes (either physical or virtual). Nevertheless, the mapping is not straight forward and involves profound analysis of different interacting factors such as-resource usage/demand, service demand, data locality, deadlines and other constraints [37].

In a nutshell, the designed controller achieves its objectives in two parallel phases namely-(a) scheduling and (b) synchronization. In the former phase, the controller schedules the containerized applications to the available nodes while meeting the energy consumption and performance obligations using the formulated multi-objective optimization problem (detailed in the next Section). In the latter phase, the controller keeps a check on the deployed containers (with respect to desired state against the actual state) and synchronizes them if needed using the designed optimization problem. The proposed *KEIDS* scheduler also leverages the information available with the *etcd* module of the Kubernetes platform with respect to the current and desired state of the underlying containers to meet the above

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2939534, IEEE Internet of Things Journal

mentioned objectives.

4 Problem Definition & Proposed Scheme

This section presents the formulation of multi-objective optimization problem for container scheduling in multi-constraint Kubernetes set-ups. The related symbols and their respective definitions are defined in Table 1.

4.1 Multi-objective optimization problem

It is evident from the above discussion that the problem under consideration is a typical case of job scheduling problem with competing objectives. The defined objective incorporate optimal scheduling of containers with (a) maximal use of green energy leading to minimal emission of carbon footprints, (b) optimal performance with minimal interference, and (c) energy minimization. The mathematical formulation with the said objectives is as follows:

$$\mathbf{F}(\mathfrak{T}_{ijkl}) = \min\ f(\mathbf{F}_1(\mathfrak{T}_{ijkl}), \mathbf{F}_2(\mathfrak{T}_{ijkl}), \mathbf{F}_3(\mathfrak{T}_{ijkl})) \quad (1)$$
$$s.t. \quad \text{Constraints specified using Eqs. (6)-(10) hold}$$

In the above equation, $\mathbf{F}(\mathfrak{T}_{ijkl})$ represents the Multi-objective optimization problem (MOOP) for efficient scheduling/mapping of containers to the available nodes in multi-constraint set-ups; with $\mathfrak{T}_{ijkl}$ being the binary decision variable defined as follows:

$$\mathfrak{T}^p_{ijkl} = \begin{cases} 1: & \text{If the } i^{th} \text{ container (associated with the } p^{th} \text{ job)} \\ & \text{is mapped to the } j^{th} \text{ pod deployed on the } k^{th} \\ & \text{node of the } l^{th} \text{ cluster} \\ 0: & \text{Otherwise} \end{cases}$$

The formulated MOOP is a typical case of ILP, which is solved using Mosek Solver [38]. The detailed description of the objective functions along with their respective constraints is provided below. These objective functions and constraints are purely linear in nature; which makes the considered MOOP a linear problem.

4.2 Objective Functions

This section details the important objective functions considered in this work.

4.2.1 Minimal Emission of Carbon Footprints with Green Energy Utilization

The carbon intensity ($\mathbf{R}(\tau)$) varies significantly depending on source of electricity generation. Here, the variables $R(S), R(W),$ and $R(NR)$ have been used to represent carbon intensity related to the energy sources, *i.e.,* solar (S), wind (W) and non-RES (NR). The metric carbon intensity is expressed in tons per mega watt hour electricity used (Tons/MWh). Higher values of the carbon intensity metric denote higher emission of carbon dioxide in the environment, which is expressed as [39].

$$\mathbf{R}^p_{ijkl} = \sum_{\tau \in \{S,W,NR\}} \left(\mathbf{E}^p_{ijkl}(s,\tau) + \mathbf{E}^p_{ijkl}(o,\tau) \right) \times R^p_{ijkl}(\tau) \quad (2)$$

Here, $\mathbf{R}^p_{ijkl}$ denotes the carbon footprint associated with the scheduling of the p^{th} job on the i^{th} container of the j^{th} pod. The variables $\mathbf{E}^p_{ijkl}(s,\tau)$ and $\mathbf{E}^p_{ijkl}(o,\tau)$ denote the energy associated with the scheduling of a job and its related overhead, respectively.

TABLE 1: List of symbols and their meanings.

Symbols	Meanings
i	Indices the containers
j	Indices the pods
k	Indices the nodes
l	Indices the clusters
p	Indices the jobs
C	Number of clusters
N	Number of nodes
P	Number of pods encapsulated in nodes
Co	Number of containers
$\mathbf{E}^p_{ijkl}(s,\tau)$	Energy associated with the job scheduling of the p^{th} job on the i^{th} container
$\mathbf{E}^p_{ijkl}(o,\tau)$	Overhead energy associated with job scheduling of the p^{th} job on the i^{th} container
$\mathbf{E}^p_{ijkl}$	Energy involved in executing the i^{th} container of the l^{th} cluster.
$\mathbf{F}_1(\mathfrak{T}^p_{ijkl})$	Objective function to reduce the carbon footprint emission
$\mathbf{F}_2(\mathfrak{T}^p_{ijkl})$	Objective function to minimize interference degree
$\mathbf{F}_3(\mathfrak{T}_{ijkl})$	Objective function to minimize the overall energy consumption
$\mathfrak{T}_{ijkl}$	Binary decision variable depicting if the i^{th} container is mapped to the j^{th} pod deployed on the k^{th}
$\mathbf{I}^p_{ijkl}$	Minimal interference degree associated with the scheduling of the p^{th} job on the i^{th} container of the j^{th} pod.
$\mathbf{R}(\tau)$	Carbon intensity
$R(S)$	Variable representing carbon intensity for solar energy
$R(W)$	Variable representing carbon intensity for wind energy
$R(NR)$	Variable representing carbon intensity for NR
$\mathbf{R}^p_{ijkl}$	Carbon footprint associated with the scheduling of the p^{th} job on the i^{th} container of the j^{th} pod.
$\mathfrak{R}$	Replication counter
$\mathfrak{S}^p_{ijkl}$	Binary variable depicting if the i^{th} container (associated with the p^{th} job) application is either CPU/network intensive
$\mathfrak{Pr}^p_{ijkl}$	Processing time of executing the i^{th} container in lieu of the p^{th} job
CPU_{kl}, M_{kl} & B_{kl}	Upper bound on the CPU, memory, and network bandwidth available with the k^{th} node of the l^{th} cluster, respectively
CPU^p_{ijkl}, M^p_{ijkl} & B^p_{ijkl}	CPU, memory, and network bandwidth allocated to the i^{th} container for executing p^{th} job, respectively

The objective function ($\mathbf{F}_1(\mathfrak{T}^p_{ijkl})$) ensures the maximal use of RES in order to reduce the carbon footprint emission. Its mathematical formulation is expressed as [39].

$$\mathbf{F}_1(\mathfrak{T}^p_{ijkl}) = \sum_{i=1}^{Co} \sum_{j=1}^{P} \sum_{k=1}^{N} \sum_{l=1}^{C} \sum_{p=1}^{J} \left(\mathfrak{T}^p_{ijkl} \times \mathbf{R}^p_{ijkl} \right) \quad (3)$$

The above mentioned objective function minimizes the emission of carbon footprints in the environment by maximizing the green energy usage; while scheduling the containers in Kubernetes set-ups.

4.2.2 Optimal Performance with Minimal Interference

Motivated from the results drawn by the authors in [40], we present an interference minimization model in this paper coupled with minimum energy utilization and maximal green energy consumption. In accordance with Medel *et al.*'s

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2939534, IEEE Internet of Things Journal

work, sources of interferences (SoIs) in a typical Kubernetes platform may vary and present different sets of influences on the deployment of containers across the pods. These SoIs include and are not limited to CPU, cache memory and network utilization to I/O access by the application, among others. Based on the extensive simulations performed by the authors on micro-benchmark datasets, the following inferences were drawn:

- CPU intensive applications scale up the overall overhead by almost 14% and the research findings suggest that the grouping of similar containers in a single pod is an effective strategy to minimize the associated overhead.
- Research outcomes also suggest that altering the deployment strategy for I/O intensive applications does not have a significant impact on the overhead caused by multiple I/O accesses.
- The experimental results also highlight that deploying several pods with similar containers helps to reduce the network interference substantially in comparison to single pod with all the containers.

On the basis of the above facts and findings, the following objective function is formulated which aims to achieve minimal interference degree ($\mathbf{I}^p_{ijkl}$) while scheduling the application containers in Kubernetes set-ups.

$$\mathbf{F}_2(\mathfrak{T}^p_{ijkl}) = \sum_{i=1}^{Co}\sum_{j=1}^{P}\sum_{k=1}^{N}\sum_{l=1}^{C}\sum_{p=1}^{J}\left(\mathfrak{T}^p_{ijkl} \times \mathfrak{S}^p_{ijkl} \times \mathbf{I}^p_{ijkl}\right) \quad (4)$$

In the above equation, $\mathfrak{S}^p_{ijkl}$ represents the container application type, which is defined as follows:

$$\mathfrak{S}^p_{ijkl} = \begin{cases} 1: & \text{If the } i^{th} \text{ container application is either CPU or} \\ & \text{network intensive} \\ 0: & \text{Otherwise} \end{cases}$$

The value of $\mathfrak{S}^p_{ijkl}$ is equivalent to 1 if the i^{th} container application is either CPU or network intensive. Otherwise, $\mathfrak{S}^p_{ijkl}$ assumes the value of 0, *e.g.*, for I/O intensive applications. On the basis of the above classification, their respective I^p_{ijkl} values are computed in accordance with the facts and findings mentioned below.

$$\mathbf{I}^p_{ijkl} = \begin{cases} 0: & \text{If } \mathfrak{S}^p_{ijkl} = 0 \\ 0 & \text{If } (\mathfrak{S}^p_{ijkl} = 1) \;\&\&\; (p^{th} \text{ job is CPU intensive}) \\ & \&\& \text{ Groups similar containers on the same pod} \\ & (j = j^{'}, k = k^{'}, l = l^{'}; \forall i \text{ of } p^{th} \text{ container}) \\ 0 & \text{If } (\mathfrak{S}^p_{ijkl} = 1) \;\&\&\; (p^{th} \text{ job is network intensive}) \\ & \&\& \text{ Groups similar containers on different pods} \\ & (j \neq j^{'}, k = k^{'}/k \neq k^{'}, l = l^{'}/l \neq l^{'}; \\ & \forall i \text{ of } p^{th} \text{ container}) \\ 1: & \text{Otherwise} \end{cases}$$

The variable $\mathbf{I}^p_{ijkl}$ can take up either the value of 0 or 1; wherein former denotes no interference and the latter denotes interference. In detail, the variable $\mathbf{I}^p_{ijkl}$ assume the value 0 under three scenarios, *i.e.*, i) when the container application is neither CPU nor network intensive, ii) when the container application is CPU intensive and groups containers on small pod, and iii) when the container application is network intensive and groups containers on different pods. In rest of scenarios, the variable $\mathbf{I}^p_{ijkl}$ assumes the value of one. For instance, when $\mathfrak{S}^p_{ijkl} = 1$, job is CPU intensive and group similar container of different pods.

4.2.3 Minimum Energy Utilization

This objective function deals with the minimization of overall energy consumption of clusters while simultaneously achieving optimal scheduling of containers in Kubernetes set-ups. It is achieved using the below mentioned objective function ($\mathbf{F}_3(\mathfrak{T}_{ijkl})$):

$$\mathbf{F}_3(\mathfrak{T}^p_{ijkl}) = \sum_{i=1}^{Co}\sum_{j=1}^{P}\sum_{k=1}^{N}\sum_{l=1}^{C}\sum_{p=1}^{J}\left(\mathfrak{T}^p_{ijkl} \times \mathbf{E}^p_{ijkl}\right) \quad (5)$$

In the above equation, $\mathbf{E}^p_{ijkl}$ denotes the energy involved in executing the i^{th} container (allocated for the p^{th} job) on the j^{th} pod.

4.3 Associated Constraints

The list of constraints while scheduling jobs in Kubernetes set-ups are elaborated below.

4.3.1 Deadline Constraint

The below mentioned expression ensures that the performance of the scheduled jobs is not jeopardized in the Kubernetes by meeting the respective job deadlines across the time horizon.

$$\sum_{i=1}^{Co}\sum_{j=1}^{P}\sum_{k=1}^{N}\sum_{l=1}^{C}\mathfrak{T}^p_{ijkl} \times \mathfrak{P}r^p_{ijkl} \leq \mathfrak{D}_p; \forall p \quad (6)$$

In the above equation, $\mathfrak{P}r^p_{ijkl}$ denotes the processing time of executing the i^{th} container on the j^{th} pod deployed on the l^{th} cluster in lieu of the p^{th} job. Further, $\mathfrak{D}_p$ denotes the deadline for executing the p^{th} job.

4.3.2 Resource Constraints

The optimal scheduling of application containers on the available nodes shall ensure that the allocation process respects the available array of resources (CPU, memory and bandwidth). The following constraints depict the same.

$$\sum_{i=1}^{Co}\sum_{j=1}^{P}\sum_{p=1}^{J}\mathfrak{T}^p_{ijkl} \times CPU^p_{ijkl} \leq CPU_{kl}; \forall k, l \quad (7)$$

$$\sum_{i=1}^{Co}\sum_{j=1}^{P}\sum_{p=1}^{J}\mathfrak{T}^p_{ijkl} \times M^p_{ijkl} \leq M_{kl}; \forall k, l \quad (8)$$

$$\sum_{i=1}^{Co}\sum_{j=1}^{P}\sum_{p=1}^{J}\mathfrak{T}^p_{ijkl} \times B^p_{ijkl} \leq B_{kl}; \forall k, l \quad (9)$$

In the above mentioned constraints, the variables CPU_{kl}, M_{kl} and B_{kl} represent the upper bound on the CPU, memory and network bandwidth available with the k^{th} node of the l^{th} cluster. Similarly, CPU^p_{ijkl}, M^p_{ijkl} and B^p_{ijkl} denote the CPU, memory and network bandwidth allocated to the i^{th} container for successful execution of the p^{th} job.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2939534, IEEE Internet of Things Journal

4.3.3 Replication Constraint

Kubernetes is an excellent container management tool that pays due attention to fault tolerance mechanisms. It achieves the same by leveraging the concept of container replication. This is mathematically expressed as follows

$$\mathfrak{T}^{p}_{ijkl} \leq \mathfrak{R}; \forall i, j, k, l, p, \tag{10}$$

where $\mathfrak{R}$ denotes the replication counter. The above mentioned constraint ensures that no more than $\mathfrak{R}$ replicas of the i^{th} container are created at any time stamp.

5 Results and Discussions

This section illustrates the performance evaluation of the designed *KEIDS* scheduler on real-time data traces against the existing state-of-the-art. In addition to this, the details of the simulation environment, baseline algorithms and evaluation parameters are also detailed below.

5.1 Simulation Setup

In order to extensively evaluate the performance of the designed Kubernetes scheduler, real-time data traces have been acquired from Google cluster [25]. Further, the powerful Mosek solver has been employed to solve the designed MOOP for job scheduling [38]. In the considered simulated environment, a total of four clusters have been taken into account, which are assumed to be geographically scattered (as seen in real-time scenario) and equipped with RES. The choice of different clusters indicate that they experience different availability proportions of RES across the time horizon. The variability in the availability of RESs at these clusters, helps to effectively evaluate the proposed scheme against the existing schemes. More importantly, the proposed scheme is also scalable, since Kubernetes is based on client-server architecture; and every cluster can have a dedicated client to process the incoming jobs in an energy-aware and interference-driven manner.

5.2 Baseline Algorithms

The relative performance comparison of the proposed scheme has been carried out against three baseline algorithms. The first one is the standard First Come First Served (FCFS)-based approach. Unlike the proposed scheme, FCFS-based approach does not give any special emphasis to energy consumption minimization, carbon footprint emission and performance interference; either individually or in combination. Furthermore, two variants of KEIDS have also been considered for evaluation purpose; wherein the former scheduler targets the minimization of interference solely, while the latter gives emphasis on minimizing the energy consumption and carbon footprints. The FCFS scheduler and the considered variants of KEIDS are referred as existing schedulers 1, 2, 3 (ES-1, ES-2, ES-3), respectively.

5.3 Evaluation parameters

For assessment of the proposed scheme against the existing schemes, the following parameters were considered in the evaluation study: i) number of active servers, ii) average CPU utilization, iii) carbon foot print rate (in kg), iv) average interference between co-located containers, and v) cumulative energy utilization of clusters (in kWh). Based on these parameters, the following findings were observed on the simulated setup.

5.4 Observations

The dataset used for performance evaluation is depicted in Fig. 3(a); where the number of incoming jobs for effective job scheduling is also shown. The said dataset has been obtained from publicly available traces released by Google [25]. For instance, at 0500 hours, 3077 number of jobs were waiting to be scheduled. Out of these jobs, 277, 420 and 2380 number of jobs were classified into CPU, memory, and bandwidth intensive type jobs. The related results are highlighted using Fig. 3(b). This categorization of job types plays an essential role during performance interference computation as already detailed in Section 4.2.2.

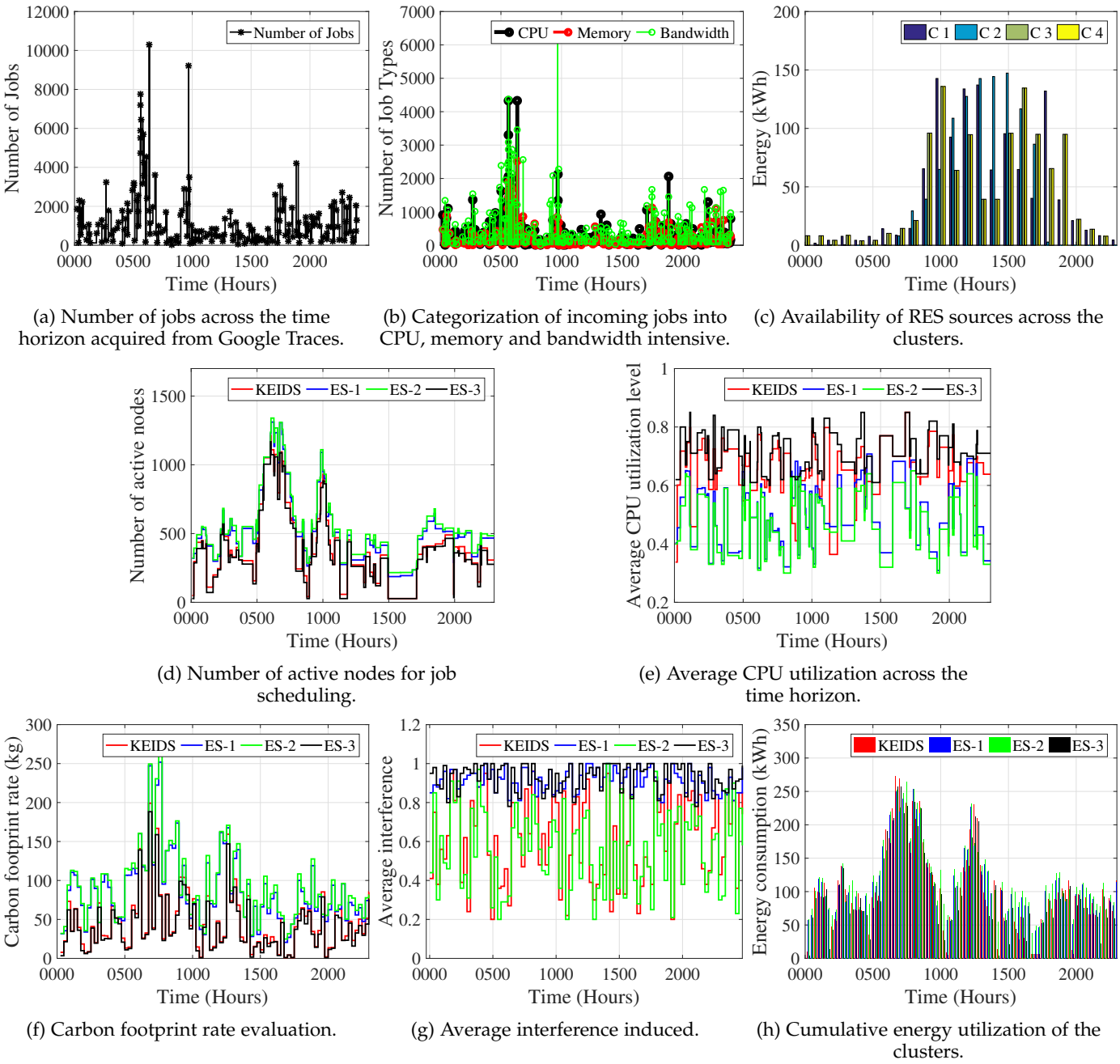
The availability of RESs across the different geographically distributed clusters namely-C1, C2, C3, and C4 is highlighted in Fig. 3(c). For example, at 0500 hours, a total of 7.66, 0, 4.26 and 4.3 kWh of energy generated from RESs was available at C1, C2, C3, and C4 respectively. Using this energy in combination with the grid generated energy, the incoming jobs were scheduled in the containerized environment provided by Kubernetes. In order to achieve the same, the containers were mapped to the available nodes scattered across the cloud and the edge. Since, one of the objectives of the proposed *KEIDS* scheduler is to ensure overall energy minimization, hence it tries to achieve the same by consolidating the incoming workload on a minimum number of nodes. This is evident from the results depicted in Fig. 3(d); where the reduced number of active nodes substantially minimize the overall energy utilization threshold. For instance, at 0500 hours, *KEIDS* utilized a total of 408 nodes against a total of 569, 601, and 358 nodes by the existing schemes, respectively.

In addition, the average CPU utilization level across the active nodes is also affected by workload consolidation principle as explained above. Consequently, the proposed *ES-3* scheduler has the highest average CPU utilization levels, followed by KEIDS, *ES-1*, and *ES-2* (as evidenced from the results summarized in Fig. 3(e)). For instance, CPU utilization levels were 67%, 53%, 56%, and 59% across the proposed and existing schedulers (*ES-1*, *ES-2*, and *ES-3*) respectively, at 0500 hours.

Due to the increased utilization of RES, *ES-3* along with *KEIDS* scheduler leads to reduced emission of carbon footprints. It is evident from Fig. 3(f), where the carbon footprint emission is expressed in terms of kilograms (kg) that *ES-3* and KEIDS lead to enhanced utilization of green energy sources relative to the *ES-1* and *ES-2*. At 0500 hours, the overall carbon footprint emission was approximately 68.181 kg by the *KEIDS* scheduler, relative to 124.01 kg by the FCFS-based approach. Overall, KEIDS led to an overall improvement of 47.08% and 49.09% in terms of reducing the emission of carbon footprint relative to *ES-1* and *ES-2*. On the contrary, *ES-3* attained 16.6% improved performance relative to KEIDS.

Moreover, the degree of interference and the total energy utilization across the schemes is illustrated in Figs. 3(g)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2019.2939534, IEEE Internet of Things Journal



(a) Number of jobs across the time horizon acquired from Google Traces.

(b) Categorization of incoming jobs into CPU, memory and bandwidth intensive.

(c) Availability of RES sources across the clusters.

(d) Number of active nodes for job scheduling.

(e) Average CPU utilization across the time horizon.

(f) Carbon footprint rate evaluation.

(g) Average interference induced.

(h) Cumulative energy utilization of the clusters.

Fig. 3: Relative performance evaluation of the proposed *KEIDS* scheduler with existing schedulers.

and 3(h) respectively. It is clearly visible from the obtained results that the proposed *KEIDS* scheduler overpowers the conventional FCFS policy across the two objective functions. On the contrary, KEIDS gives better performance in terms of energy consumption against *ES-2* (by approx. 19%) and reduced interference against *ES-3* (by approx. 33.12%). The proposed scheme outperformed the *ES-1* with almost 0.37 of average interference relative to average interface of 0.92 by the FCFS. At 0500 hours the *KEIDS* scheduler led to an overall energy consumption of 93.399 kWh in contrast to 124.01 kWh by *ES-1*.

Hence, it can be concluded that the performance of the proposed *KEIDS* scheduler is quite satisfactory relative to the existing schedulers and is thus suitable to adopted for Kubernetes set-ups. On an average, the proposed scheme leads to improved energy utilization, reduced CFR, and minimal interference by almost 14.42%, 47%, and 31.83%, respectively, in contrast to the conventional FCFS scheduler.

6 Conclusion

With an evolution of cloud computing and Big Data, the concept of virtual machines has been replaced with the light-weight containers. One of the best options to manage these containers is by leveraging Google's Kubernetes platform; which has an inbuilt job scheduler and synchronizer. However, the important factors like performance interference and energy minimization are not yet explored by the Kubernetes community; which limit their practicality in

view of the increasing energy consumption levels of cloud DCs. Hence, in this paper a composite controller, named as *KEIDS* is designed particularly for Kubernetes platforms. The designed controller is energy-efficient and has superior performance in comparison to other existing state-of-the-art proposals. The problem of energy efficiency is formulated as a multi-objective optimization problem that achieves optimal scheduling of containers with i.) maximal use of green energy leading to minimal emission of carbon footprints, ii) optimal performance with minimal interference, and iii.) overall energy minimization. The designed problem is a typical ILP and is solved using Mosek Solver. Furthermore, its performance has been evaluated against the existing schedulers on real-time Google traces. The obtained results indicate that the proposed scheme has better performance in terms of overall energy minimization with reduced carbon footprint and interference from the co-existed containers on the same or different nodes. In detail, the proposed scheduler leads to an overall improvement of 14.42%, 47%, and 31.83%, respectively against the FCFS scheduler in terms of improved energy utilization, reduce CFR, and minimal interference.

In the future, we will extend the concept of Kubernetes scheduling using renewable energy sources only; in order to substantially reduce the hazardous environmental impacts of modern data centers.

ACKNOWLEDGMENT

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). This work was also supported by the Tier 2 Canada Research Chair.

REFERENCES

[1] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] G. S. Aujla, R. Chaudhary, K. Kaur, S. Garg, N. Kumar, and R. Ranjan, "SAFE: SDN-Assisted Framework for Edge–Cloud Interplay in Secure Healthcare Ecosystem," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 469–480, 2019.

[3] S. Tyagi, M. S. Obaidat, S. Tanwar, N. Kumar, and M. Lal, "Sensor Cloud Based Measurement to Management System for Precise Irrigation," in *IEEE Global Communications Conference, Singapore*, Dec. 2017.

[4] A. Singh, S. Garg, S. Batra, N. Kumar, and J. J. Rodrigues, "Bloom filter based optimization scheme for massive data handling in IoT environment," *Future Generation Computer Systems*, vol. 82, pp. 440–449, 2018.

[5] A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, M. Maasberg, and K.-K. R. Choo, "Multimedia big data computing and Internet of Things applications: A taxonomy and process model," *Journal of Network and Computer Applications*, vol. 124, pp. 169–195, 2018.

[6] (2018, November) Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper. Cisco. [Accessed on: Nov. 2018]. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[7] G. S. Aujla, R. Chaudhary, K. Kaur, S. Garg, N. Kumar, and R. Ranjan, "Safe: Sdn-assisted framework for edge–cloud interplay in secure healthcare ecosystem," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 469–480, 2018.

[8] A. Singh, S. Garg, K. Kaur, S. Batra, N. Kumar, and K. R. Choo, "Fuzzy-Folded Bloom Filter-as-a-Service for Big Data Storage in the Cloud," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2338–2348, 2019.

[9] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[10] S. Garg, A. Singh, K. Kaur, S. Batra, N. Kumar, and M. S. Obaidat, "Edge-Based Content Delivery for Providing QoE in Wireless Networks Using Quotient Filter," in *IEEE International Conference on Communications (ICC), Kansas City, USA*, May 2018.

[11] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: a case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.

[12] S. Garg, A. Singh, S. Batra, N. Kumar, and L. T. Yang, "UAV-Empowered Edge Computing Environment for Cyber-Threat Detection in Smart Vehicles," *IEEE Network*, vol. 32, no. 3, pp. 42–51, 2018.

[13] M. Tiwary, D. Puthal, K. S. Sahoo, B. Sahoo, and L. T. Yang, "Response time optimization for cloudlets in Mobile Edge Computing," *Journal of Parallel and Distributed Computing*, vol. 119, pp. 81–91, 2018.

[14] J. Pan and J. McElhannon, "Future Edge Cloud and Edge Computing for Internet of Things Applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2018.

[15] (2010, April) CEO to shareholders: 50 billion connections 2020. Ericsson. [Accessed on: Sept. 2018]. [Online]. Available: https://www.ericsson.com/en/press-releases/2010/4/ceo-to-shareholders-50-billion-connections-2020

[16] Y. Chen, N. Zhang, Y. Zhang, and X. Chen, "Dynamic Computation Offloading in Edge Computing for Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4242–4251, 2019.

[17] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues, and M. Guizani, "Edge Computing in the Industrial Internet of Things Environment: Software Defined Networks-Based Edge-Cloud Interplay," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 44–51, Feb 2018.

[18] K. Kaur, T. Dhand, N. Kumar, and S. Zeadally, "Container-as-a-Service at the Edge: Trade-off between Energy Efficiency and Service Availability at Fog Nano Data Centers," *IEEE Wireless Communications*, vol. 24, no. 3, pp. 48–56, June 2017.

[19] D. Zhang, B.-H. Yan, Z. Feng, C. Zhang, and Y.-X. Wang, "Container oriented job scheduling using linear programming model," in *3rd International Conference on Information Management (ICIM), Chengdu, China*. IEEE, April 2017.

[20] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, Sept 2014.

[21] M. Sureshkumar and P. Rajesh, "Optimizing the docker container usage based on load scheduling," in *2nd International Conference on Computing and Communications Technologies (ICCCT), Chennai, India*. IEEE, Feb. 2017.

[22] Y. Biran, S. Pasricha, G. Collins, and J. Dubow, "Enabling green content distribution network by cloud orchestration," in *Smart Cloud Networks & Systems (SCNS), Dubai, United Arab Emirates*. IEEE, Dec. 2016.

[23] V. Medel, O. Rana, J. Á. Bañares, and U. Arronategui, "Adaptive application scheduling under interference in kubernetes," in *Proceedings of the 9th International Conference on Utility and Cloud Computing, Shanghai, China*. ACM, Dec. 2016.

[24] G. S. Aujla, N. Kumar, S. Garg, K. Kaur, R. Rajan, and S. Garg, "Renewable Energy-based Multi-Indexed Job Classification and Container Management Scheme for Sustainability of Cloud Data Centers," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2947 – 2957, 2019.

[25] J. Wilkes, "More Google cluster data," Google research blog, Nov. 2011, posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.

[26] C. C. Chang, S. R. Yang, E. H. Yeh, P. Lin, and J. Y. Jeng, "A Kubernetes-Based Monitoring Platform for Dynamic Cloud Resource Provisioning," in *IEEE Global Communications Conference (Globecom), Singapore*, Dec 2017.

[27] C. Wöbker, A. Seitz, H. Mueller, and B. Bruegge, "Fogernetes: Deployment and management of fog computing applications," in *IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan*. IEEE, April 2018.

[28] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, "Secure and sustainable load balancing of edge data centers in fog computing," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 60–65, 2018.

[29] V. Medel, R. Tolosana-Calasanz, J. Ángel Bañares, U. Arronategui, and O. F. Rana, "Characterising resource management performance in Kubernetes," *Computers & Electrical Engineering*, vol. 68, pp. 286 – 297, 2018.

[30] R. Kumar, M. Kalra, S. Tanwar, S. Tyagi, and N. Kumar, "Minparent: An effective approach to enhance resource utilization in cloud environment," in *International Conference on Advances in Computing, Communication, Automation (ICACCA) (Spring), Dehradun, India*, April 2016.

[31] S. K. Mishra, D. Puthal, J. J. Rodrigues, B. Sahoo, and E. Dutkiewicz, "Sustainable Service Allocation Using a Metaheuristic Technique in a Fog Server for Industrial Applications," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4497–4506, 2018.

[32] Q. Fan, N. Ansari, and X. Sun, "Energy driven avatar migration in green cloudlet networks," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1601–1604, 2017.

[33] A. Kiani and N. Ansari, "Toward low-cost workload distribution for integrated green data centers," *IEEE Communications Letters*, vol. 19, no. 1, pp. 26–29, 2014.

[34] H. Yeganeh, A. Salahi, and M. A. Pourmina, "A Novel Cost Optimization Method for Mobile Cloud Computing by Capacity Planning of Green Data Center With Dynamic Pricing," *Canadian Journal of Electrical and Computer Engineering*, vol. 42, no. 1, pp. 41–51, 2019.

[35] M. S. Hasan, Y. Kouki, T. Ledoux, and J.-L. Pazat, "Exploiting renewable sources: when green sla becomes a possible reality in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 5, no. 2, pp. 249–262, 2015.

[36] R. Arshad, S. Zahoor, M. A. Shah, A. Wahid, and H. Yu, "Green IoT: An investigation on energy saving practices for 2020 and beyond," *IEEE Access*, vol. 5, pp. 15 667–15 681, 2017.

[37] G. Sayfan, *Mastering Kubernetes*. Packt Publishing Ltd, 2017.

[38] "Mosek," [Accessed on: Apr. 2017]. [Online]. Available: https://www.mosek.com/resources/downloads

[39] A. Khosravi, L. Andrew, and R. Buyya, "Dynamic VM Placement Method for Minimizing Energy and Carbon Cost in Geographically Distributed Cloud Data Centers," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 183 – 196, 2017.

[40] V. Medel, O. Rana, J. Á. Bañares, and U. Arronategui, "Modelling performance & resource management in kubernetes," in *IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC), Shanghai, China*, Dec. 2016.

Kuljeet Kaur (S'13, M'18) received her Ph.D. degree from the Thapar Institute of Engineering and Technology in 2018. She is currently working as an NSERC post-doctoral research fellow at École de technologie supérieure, Université du Québec, Montréal, Canada. She has many research contributions in the areas of cloud computing, energy efficiency, smart grid, frequency support, and vehicle-to-grid. Some of her research findings are published in top cited journals such as IEEE TII, IEEE TMM, IEEE TSG, IEEE TVT, IEEE TNSM, the IEEE Systems Journal, IEEE TSUSC, IEEE Communications Magazine, IEEE Wireless Communications, IEEE Network, IEEE PS, and Springer PPNA. She also received the IEEE ICC best paper award in 2018 at Kansas City, Missouri.

Sahil Garg (S'15, M'18) received his Ph.D. degree from the Thapar Institute of Engineering and Technology, Patiala, India, in 2018. He is currently working as a post-doctoral research fellow at École de technologie supérieure, Université du Québec, Montréal, Canada. He has many research contributions in the area of machine learning, big data analytics, security & privacy, internet of things, and cloud computing. Some of his research findings have been published in top cited journals such as IEEE TII, IEEE TMM, IEEE TNSM, IEEE TVT, the IEEE Systems Journal, the IEEE Internet of Things Journal, IEEE TSUSC, IEEE Communications Magazine, IEEE Network, IEEE Wireless Communications, IEEE CE Magazine, FGCS, and Information Sciences. He also received the IEEE ICC best paper award in 2018 at Kansas City, Missouri.

Georges Kaddoum (M'11) received his Bachelor's degree in electrical engineering from the École Nationale Supérieure de Techniques Avancés (ENSTA), France, his M.S. degree in telecommunications and signal processing from Telecom Bretagne (ENSTB), Brest, in 2005, and his Ph.D. degree in signal processing and telecommunications from the National Institute of Applied Sciences (INSA), Toulouse, France, in 2009. He is currently an associate professor and research chair of electrical engineering with the École de Technologie Supérieure, University of Quebec, Montréal, Canada. His recent research activities cover wireless communication networks, resource allocations, security and space communications, and navigation.

Syed Hassan Ahmed (S'13, M'17, SM'18) completed his BS from KUST, Pakistan and his M.S./Ph.D. from Kyungpook National University, South Korea, both in computer science, in 2012 and 2017, respectively. Later, he was a post-doc with the University of Central Florida, Orlando. Currently, he is on the faculty of the CS Department of Georgia Southern University (GSU) at Statesboro, USA, where his research interests include sensor and ad hoc networks, cyber-physical systems, vehicular communications and Future Internet.

Mohammed Atiquzzaman (M'87, SM'95) received the MS and PhD degrees in electrical engineering and electronics from the University of Manchester, United Kingdom. He currently holds the Edith Kinney Gaylord Presidential professorship in the School of Computer Science at the University of Oklahoma. He is the editor-in-chief of Journal of Network and Computer Applications, founding editor-in-chief of Vehicular Communications and has served/serving on the editorial boards of various IEEE journals and co-chaired numerous IEEE international conferences including IEEE Globecom. His research interests are in communications switching, transport protocols, wireless and mobile networks, satellite networks, and optical communications. His research has been funded by National Science Foundation, NASA, US Air Force, Cisco, Honeywell and other funding agencies. Most of his publications can be found at www.cs.ou.edu/ atiq.