INVITED PAPER

A Survey of Virtual Machine Management in Edge Computing

This article provides an overview of the industrial and research projects on virtual machine (VM) management in edge computing, focusing on the virtualization frameworks and virtualization techniques, the serverless management, and the security advantages and issues that virtualization brings to edge computing.

By Zeyi Tao, Qi Xia, Zijiang Hao, Cheng Li, Lele Ma, Shanhe Yi, and Qun Li

ABSTRACT | Many edge computing systems rely on virtual machines (VMs) to deliver their services. It is challenging, however, to deploy the virtualization mechanisms on edge computing hardware infrastructures. In this paper, we introduce the engineering and research trends of achieving efficient VM management in edge computing. We elaborate on: 1) the virtualization frameworks for edge computing developed in both the industry and the academia; 2) the virtualization techniques tailored for edge computing; 3) the placement and scheduling algorithms optimized for edge computing; and 4) the research problems in security related to virtualization of edge computing.

KEYWORDS | Algorithms; management; virtual machining.

I. INTRODUCTION

Since Amazon released its elastic compute cloud (EC2) [1] product in 2006, cloud computing has become increasingly important in people's daily life. By providing elastic hardware resources, including processing resources, storage resources, and networking resources, at the data centers residing at the core of the Internet, cloud computing enables a spectrum of applications that have profoundly impacted the contemporary computational patterns for both industrial and individual uses. Companies can benefit from cloud computing by executing large batch-oriented tasks on cloud servers, and individual users can rely on remote clouds to perform resource-intensive computations for their client devices. Because cloud computing has brought so many applications into reality, commercial cloud platforms, such as Amazon AWS [2], Microsoft Azure [3], and Google Cloud [4], have been successively put into operation in recent years.

Nevertheless, cloud computing suffers from a severe problem when serving client devices at the edge of the Internet. Since cloud data centers usually reside at the core of the Internet, it is always the case that client devices have a long network distance to the remote clouds, which leads to significant network delay perceived by the end users. This is unacceptable for many application scenarios, especially for latency-sensitive mobile-cloud and IoT-cloud applications, where the client devices are mobile devices such as smartphones and Internet-of-Things (IoT) devices, respectively.

In light of this situation, edge computing [5]–[7] (also known as fog computing [8], [9] and cloudlets [10], [11]) was proposed as an extension of cloud computing. By providing hardware sources at the edge of the Internet, edge computing serves client devices with much lower network latency than cloud computing, thereby greatly improving the user experience for delay-sensitive client-remote applications. More importantly, edge computing has recently become a concept beyond merely an extension of cloud computing [8]. Indeed, edge computing is now frequently mentioned as an enabling technology of IoT [12], [13] and is widely adopted by a range of rapidly develop-

Manuscript received February 9, 2019; revised April 13, 2019 and June 20, 2019; accepted June 20, 2019. Date of publication July 25, 2019; date of current version August 5, 2019. This work was supported in part by the U.S. National Science Foundation under Grant CNS-1816399. *(Corresponding author: Zeyi Tao.)*

Z. Tao, **Q. Xia**, **Z. Hao**, **C. Li**, **L. Ma**, and **Q. Li** are with the College of William and Mary, Williamsburg, VA 23185 USA (e-mail: ztao@cs.wm.edu; qxia@cs.wm.edu; hebo@cs.wm.edu; cli04@cs.wm.edu; lelema@cs.wm.edu; liqun@cs.wm.edu).
S. Yi is with VMware, Palo Alto, CA 94304 USA (e-mail: yshanhe@vmware.com).

Digital Object Identifier 10.1109/JPROC.2019.2927919

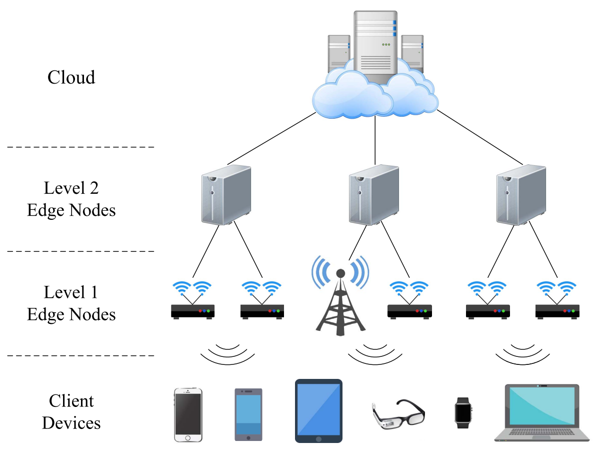

Fig. 1. ***Example architecture of edge computing.***

ing applications, such as big data analytics [14], video surveillance [15]–[17], virtual reality [18]–[20], and augmented reality (AR) [21]–[23].

Fig. 1 shows an example architecture of edge computing. As shown in Fig. 1, client devices are wirelessly connected to the level 1 edge nodes, which are Wi-Fi access points (APs) and cellular towers. Behind the level 1 edge nodes are the level 2 edge nodes, which have a longer network distance to the client devices than the level 1 edge nodes. There is also a backend cloud behind the level 2 edge nodes that resides at the core of the Internet. The client devices perceive lower delay when the computations are done at the level 1 edge than at the level 2 edge. On the other hand, the level 2 edge nodes possess more powerful hardware than the level 1 edge nodes and can thus execute more computational tasks simultaneously. Note that Fig. 1 merely shows a possible architecture of edge computing. In other edge computing architectures, it is possible that there are only one level or more than two levels of edge nodes and that there is no backend cloud in the system.

Generally, cloud computing is built on several fundamental technologies, including virtualization, distributed mass storage, and parallel programming models [24]. Among these technologies, virtualization plays an important role in resource provisioning, task scheduling, and computation sandboxing in cloud computing environments. Although built on less powerful hardware, edge computing faces similar challenges as cloud computing in effectively managing the hardware resources. Therefore, edge computing also employs virtualization as one of its fundamental technologies. At a high level, the virtualization technology, no matter in the form of virtual machines (VMs) or containers, provides the following features that are critical in delivering flexible and reliable edge computing services.

1) *Platform Independence:* The hardware infrastructure of edge computing can be highly heterogeneous, given that the edge nodes are possibly provided by multiple third parties and/or individuals. Virtualization is the most widely adopted solution to this problem. VMs and containers in the same specifications always produce the identical execution environment regardless of the heterogeneity of the underlying hardware infrastructure. Therefore, applications developed for a particular set of VM specifications can be surely executed in an edge computing environment that provisions VM instances in those specifications.
2) *Resource Abstraction:* VMs and containers are not executed on bare metal; they are executed on hypervisors. Hypervisors manage the underlying hardware resources and provide emulated hardware devices for the VMs and containers running atop. By doing so, hypervisors effectively hide the heterogeneity of the underlying hardware infrastructure, which greatly simplifies the development of cloud applications as well as the management of the hardware resources.
3) *Isolation:* VMs achieve hardware-level isolation; the abnormal status of a VM such as a software failure will not affect the correctness (sometimes even the performance) of the programs running in other VMs running on the same host [25]. This feature is highly desirable in edge computing because edge nodes are supposed to serve a number of end users and execute many tasks simultaneously. In comparison, containers only achieve operating system (OS)-level isolation; in normal cases, a program running inside a container can only access the resources assigned to that container. However, in abnormal cases, such as when a container crashes or has been compromised, the containers in problem may affect the entire host machine [26], [27]. The primary advantage of containers is that they are more lightweight than VMs, and the OS-level isolation that they achieve is usually sufficient for edge computing.

Despite the advantages of adopting the virtualization technology, edge computing has several characteristics that distinguish itself from cloud computing, which pose new challenges to the management of VMs. We summarize such characteristics and the challenges they pose as follows.

1) *Sparse Distribution:* Edge nodes are distributed more sparsely in a region than the cloud servers locating at the same data center. Therefore, it takes a longer time to migrate VMs and containers in an edge computing environment than in a cloud computing environment, which poses severe challenges in providing seamless services for edge computing users.
2) *Limited Resources:* Unlike cloud servers that possess conceptually infinite hardware resources, edge nodes usually possess limited hardware resources. Therefore, each edge node can only support a limited number of VMs and containers, which makes it quite challenging to schedule the VMs/containers in an edge computing environment.
3) *Limited Service Range:* Each edge node only covers a limited service range via wireless media. Due to the

mobility of end devices, VMs and containers may need to be frequently migrated between the adjacent edge nodes, which places high demands on the efficiency of VM/container migration in edge computing.

4) *Mobility of Edge Nodes:* Edge nodes can be mobile devices/servers in some scenarios; they can be smartphones, wearables, drones, vehicles, and so on. The mobility of edge nodes offers opportunities for achieving flexible resource provisioning and poses severe challenges to VM/container scheduling and migration in edge computing.
5) *Poor Reliability:* Edge nodes are not as reliable as cloud servers; they may frequently experience failures and network partitions. Therefore, an edge computing system must provide effective failover mechanisms for the VMs/containers running inside in order to deliver reliable services.
6) *Vulnerability to Attacks:* Most edge nodes are vulnerable to attacks because they are operated by third parties or individuals with far weaker technical strengths than the technical companies operating large cloud data centers. Therefore, the VMs and containers running in an edge computing system usually face severe challenges in achieving secure computations.

There are currently a number of efforts toward solving the aforementioned problems. Some engineers and researchers have tried to provide a full-fledged solution. They build general edge computing systems or frameworks to efficiently manage the life cycle of the hosted VMs/containers. Others focus on a particular set of the problems in edge computing virtualization. One body of work investigates how to improve the state-of-the-art virtualization techniques for edge computing environments, e.g., by reducing the size of VMs/containers and devising more efficient methods for VM/container migration. Other work strives to design VM/container placement/scheduling algorithms that are tailored for edge computing infrastructures and scenarios. In addition, security problems are studied in executing VMs/containers on vulnerable edge nodes.

The rest of this paper is organized as follows. Section II introduces the virtualization frameworks for edge computing from both the industry and the academia. Section III discusses several virtualization techniques tailored for edge computing. Section IV elaborates on the placement and scheduling algorithms optimized for edge computing scenarios. Section VI discusses the security advantages and issues that virtualization brings to edge computing. Finally, Section VII concludes this paper.

II. VIRTUALIZATION-BASED FRAMEWORKS

Since its inception, edge computing has quickly gained popularity from both the industry and the academia. To address the new challenges in edge computing, engineers and researchers have proposed their virtualization-based frameworks as solutions. In this section, we first highlight the challenges in designing such frameworks and the key solutions to them and then shed light on the state of the art of building edge computing frameworks for both business and research purposes.

A. Challenges

Designing virtualization-based frameworks for edge computing faces several key challenges. In the following, we summarize the challenges as well as the related techniques that can be utilized to build solutions to the challenges. Most of the techniques have been adopted in the design of edge computing frameworks.

1) *Large Resource Footprint of VMs:* Edge nodes are commodity personal computers that possess much less powerful hardware than clustered cloud servers. On the other hand, VMs consume considerable hardware resources for execution [28]–[30]. Therefore, although a cluster of cloud servers can host many active VMs simultaneously, it is likely that an edge node can only support a limited number of active VMs at one time. This poses a severe challenge to the design of edge computing frameworks in practice.
   To address this challenge, more lightweight virtualization techniques other than VMs, such as Linux Containers (LXCs) [31] and Docker [32], can be employed in the design of edge computing frameworks. Additionally, techniques that reduce the resource footprint of VMs, such as unikernels [33], are also effective in addressing the aforementioned challenge.
2) *Large Data Size of VMs:* Due to the mobility of end devices and sometimes even the edge nodes, many edge computing frameworks are expected to support frequent VM migrations in a timely manner. However, the large data size of VMs, both in memory and on disk [34], poses severe challenges to VM migrations in edge computing.
   Similar to the previous challenge, the performance problem imposed by this challenge can also be mitigated by using more lightweight virtualization techniques or by reducing the memory and disk footprint of VMs. Moreover, techniques that reduce the transmitted data size during VM migrations, such as data deduplication [35] and VM synthesis [10], can also be utilized to further mitigate the performance problem.
3) *Security Issues:* Even though cloud data centers are operated by companies with strong technical strengths and good reputations, they are not trustworthy in many cases [36], not to mention the edge nodes provided by individuals and small third parties. Therefore, edge computing frameworks always face significant security challenges to VM management.
   To address these challenges, techniques that guarantee the accountability of VMs/containers, such as computational auditing [37] and Intel Software Guard Extensions (SGX) [38], can be utilized in the

Table 1 Core Components of OpenStack

Category	Component	Description
Compute	Nova	Managing compute resources including virtual machines and containers
Storage	Cinder	Virtualizing the management of block storage devices
	Swift	Providing a distributed, eventually consistent object storage service
Networking	Neutron	Delivering networking-as-a-service (NaaS) based on SDN technologies
Shared Services	KeyStone	Providing authentication, service discovery, and authorization services
	Glance	Discovering, registering, and retrieving virtual machine images
Orchestration	Heat	Orchestrating the resources for applications based on templates
Telemetry	Ceilometer	Collecting information for customer billing and resource tracking
Web Frontend	Horizon	Implementing a web-based user interface to the OpenStack services

design of edge computing frameworks. Moreover, techniques, such as homomorphic encryption [39], can also be employed to protect the user privacy in edge computing environments.

B. Industrial Software Architectures

To date, there are several pioneer projects proposed by the industry that aims at building general-purpose edge computing frameworks. In this section, we elaborate on three important ones of such industrial projects: OpenStack, KubeEdge, and OpenEdge.

1) OpenStack: OpenStack [40] is an open-source platform developed by Rackspace Inc. and NASA. By deploying OpenStack as infrastructure-as-a-service (IaaS), virtual servers are made available to end users. Although the original design goal of OpenStack is to support cloud computing in data centers, the OpenStack community recently claimed that OpenStack can naturally support cloud-edge computing (CEC) thanks to its flexible and modular design [41]. OpenStack currently (Version 2018.06.01) contains 31 components in total, where 9 of them are designed for core functionalities. Table 1 demonstrates these core components of OpenStack.

In essence, the Nova compute component is designed for managing VMs and containers. It works along with the glance image management component, the neutron networking component, the cinder block storage component, and possibly other components to manage the lifecycle of the VM/container instances. For example, the Zun component provides an API to launch and manage the container services backed by third-party container technologies as OpenStack-managed resource, and the Magnum component is designed to support third-party container orchestration engines, such as Docker Swarm, Kubernetes, and Apache Mesos in OpenStack environments.

OpenStack supports various virtualization technologies by incorporating their hypervisor drivers. At the time of writing this survey, OpenStack supports more than ten kinds of hypervisors, including KVM, QEMU, UML, XenServer (and other XAPI-based Xen variants), Xen (via libvert), VMware vSphere, Hyper-V, Virtuozzo, PowerVM, LXC, and Docker.

2) KubeEdge: Kubernetes [42], [43] is an open-source container orchestration system for cloud computing. It was designed by Google and then donated to the Cloud Native Computing Foundation, a community belonging to the Linux Foundation. The design goal of Kubernetes is to automatically deploy and manage large-scale cloud applications using container runtimes such as Docker. Targeting at cloud computing scenarios, however, Kubernetes lacks proper system support for edge computing.

In order to take advantage of Kubernetes in edge computing environments, Huawei has open-sourced a cloud-edge platform called KubeEdge [44]. KubeEdge is based on Kubernetes while providing functionalities for achieving fast communication between the cloud and the edge as well as supporting resource-constrained devices at the edge. With KubeEdge, edge computing applications can be implemented using the conventional Kubernetes API and can work with Kubernetes clusters residing at remote data centers.

Fig. 2 shows the high-level design of KubeEdge and shows how KubeEdge is incorporated into a remote

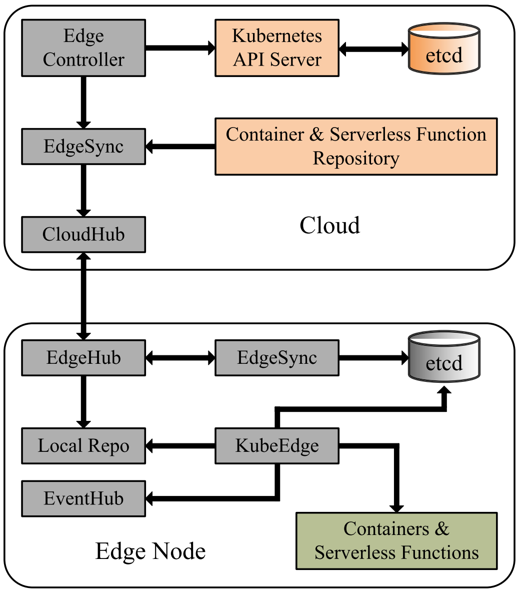

Fig. 2. ***High-level design of KubeEdge.***

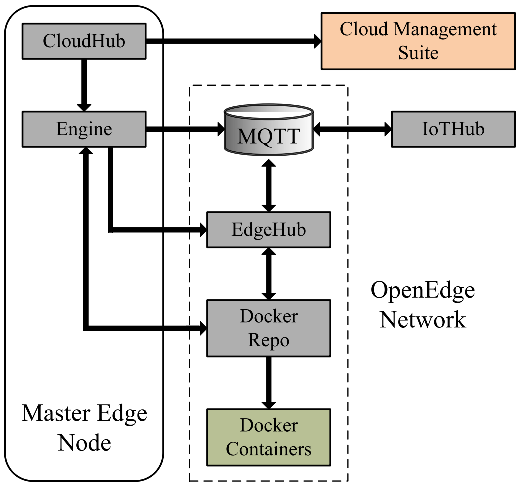

Fig. 3. ***High-level design of OpenEdge.***

Kubernetes system. Blocks in orange are components of Kubernetes, and blocks in gray are those of KubeEdge. Specifically, KubeEdge communicates with the cloud via the EdgeHub component, which fetches the containers and serverless functions from the remote repository and stores them locally. Information about the containers and serverless functions is synchronized by the EdgeSync component among the edge nodes through the etcd distributed key-value store [45]. Events from end devices are collected by the EventHub component and retrieved by KubeEdge, which will in turn launch containers and serverless functions on the edge node to perform the corresponding computations.

3) OpenEdge: OpenEdge [46] is an open-source project led by Baidu. The goal of OpenEdge is to build a flexible edge computing platform that extends cloud computing seamlessly to edge devices. To that end, OpenEdge is designed to work with the Cloud Management Suite of Baidu IntelliEdge (BIE) for achieving high performance in various cloud-edge computing scenarios.

OpenEdge adopts a modular and containerized design. An OpenEdge platform essentially contains two parts running at the edge: a master process working as an overall controller and a set of loadable modules working as plugins. In addition, OpenEdge can work in two modes at present: the Docker container mode and the native process mode. In the former mode, applications are executed in Docker containers, while in the latter mode, applications are executed as native Linux processes. Fig. 3 shows the key components of OpenEdge when working in the Docker container mode.

The gray blocks shown in Fig. 3 are components of OpenEdge, while the orange block is for cloud. To begin with, the engine component running on the master edge node fetches IoT events from the IoTHub component via the Message Queuing Telemetry Transport (MQTT) message queue [47]. In order to handle these events, the Engine component communicates with the EdgeHub component as well as the Docker Repo component to perform the corresponding computations in Docker containers. The Engine component may also communicate with the Cloud Management Suite component for synchronizing the master edge node with the cloud, fetching uncached Docker images, and so on.

C. Research Projects

A considerable effort has also been put into the research on building edge computing frameworks. In this section, we introduce several representative research projects from the academia.

1) Paradrop: Paradrop is a specific edge computing platform proposed by Liu *et al.* [48]. In essence, Paradrop aims at establishing an edge computing environment on Wi-Fi APs, because Wi-Fi APs locate at the first network hop to the end devices and are hence a splendid location to deploy edge computing.

Paradrop contains three key components: 1) a virtualization substrate running on Wi-Fi APs that performs computations for end users in Docker containers; 2) a backend cloud working as the control node to manage the lifecycle of the Docker containers on the Wi-Fi APs; and 3) a developer API through which developers can manage the resources of the Wi-Fi APs and monitor the status of the Docker containers. In particular, developers can build applications that access the IoT devices co-locating with the Wi-Fi APs, such as wireless cameras and temperature/humidity sensors, for end users. Such computations will be automatically loaded into Docker containers when executed in order to achieve isolation and reproducibility.

2) AirBox: AirBox [49] is a so-called edge function (EF) platform proposed by Bhardwaj *et al.* [49], where an EF is essentially a service provided by edge computing. AirBox was designed to achieve: 1) fast and scalable EF provisioning and 2) strong guarantees for the integrity of the EFs.

For the first design goal, Howell *et al.* [50] chose Docker containers to encapsulate EFs after conducting a comprehensive measurement study on a range of candidate technologies, including VM synthesis used by cloudlets, OS-level containers such as Docker, and user-level sandboxes such as Embassies. For the second design goal, Intel [38] employed the Intel SGX technology. SGX is a hardware feature of Intel processors introduced in 2015, which guarantees the integrity of computations with low overhead. Every AirBox EF contains two parts, a trusted part and an untrusted part, and the trusted part is protected by an SGX enclave. When an EF is executed, both the trusted part and the untrusted part will be loaded into a Docker container to simplify the management of computations.

3) Middleware for IoT Clouds: Nastic *et al.* [51] have studied the provisioning problem of IoT clouds, i.e., how to bring an IoT cloud to a state where it is usable for end users. The authors claimed that the contemporary provisioning solutions are insufficient for IoT clouds because they do not fully consider the vast heterogeneity, the geographical distribution, and the large scale of IoT clouds. To this end, the authors proposed a middleware that relies on the software-defined gateways (SDGs) as their solution.

An SDG is essentially a software-defined, lightweight VM. SDGs virtualize edge resources for IoT clouds to provide isolated and managed execution environments for edge applications. Each SDG contains a provisioning agent that interacts with the outside control plane for handling requests from end users and loading the necessary libraries for the edge application running inside. In addition, each SDG contains an SDG Monitor that works along with the outside control plane and the provisioning agent to manage the lifecycle of the SDG. The communication between the SDGs and the control plane is done via the system API, while that between the SDGs and the end devices is done through an MQTT message queue.

Noteworthy is the mobile-edge computing framework proposed by Jararweh *et al.* [52], which also embraces a software-defined design for VM management.

4) FocusStack: FocusStack is a cloud-edge orchestration platform proposed by Amento *et al.* [53]. The main problem that FocusStack tries to solve is that the traditional method of building the control plane for cloud computing is not suitable for cloud-edge computing due to the large scale of the cloud-edge systems and the mobility of the edge nodes. To address this problem, FocusStack adopts a location-based situation-aware design. To be more specific, FocusStack assumes that end devices are also edge nodes. When an end device generates a computational request, it will seek for an end device (including itself) nearby that meets the criteria of performing the computation.

FocusStack is built atop OpenStack and Docker containers. When a request is generated by an end device, the device will contact a local conductor, which will in turn contact the OpenStack server residing at the cloud via the geocast primitives [54]. After deciding which end device will perform the computation for the request, the conductor will encapsulate the computation into a Docker container and then deploy the Docker container to the target device for execution.

5) Amino: Amino is a cloud-edge runtime system designed by Xiong *et al.* [55]. The authors figured out that building cloud-edge applications is difficult because developers must implement complex distributed mechanisms, such as concurrence, consistency, and replication across the entire system. This is a challenging and time-consuming task.

In order to tackle this problem, the authors employed Sapphire [56] as the substrate of Amino. Sapphire embraces a modular design, allowing distributed applications to be easily customized and orchestrated using system libraries that implement the distributed mechanisms mentioned earlier. By incorporating Sapphire, Amino solves the aforementioned problem in cloud-edge computing. Furthermore, Amino utilizes GraalVM [57], a lightweight VM that supports interoperability between different programming languages, to execute and manage multilanguage Sapphire objects with low overhead.

6) Lightweight Service Replication for Mobile-Edge Computing: Farris *et al.* [58] have studied the challenges in implementing mobile-edge computing in practice. They figured out that the most significant challenges are: 1) the limited hardware resources at the edge and 2) the mobility of the end users.

To address these challenges, the authors: 1) utilized Docker containers to encapsulate computations, because Docker containers provide similar features to VMs such as isolation and resource abstraction but with a remarkably lower overhead and 2) proposed a lightweight service replication method based on the live migration and checkpoint/restore features provided by LXC (upon which Docker is built). By proactively migrating the Docker containers for end users while they are passing across the boundary of adjacent edge nodes and executing redundant Docker containers during the migration, performance downgrade stemmed from user mobility can be largely mitigated.

7) SOUL: SOUL is a cloud-edge framework for mobile applications proposed by Jang *et al.* [59]. The primary goal of SOUL is to build an edge environment for mobile devices such as Android phones, sharing the sensors on the mobile devices to support innovative use cases in edge computing scenarios.

In essence, SOUL virtualizes the sensors on the mobile devices that are connected to the edge. By doing so, SOUL unifies all the sensors as a sensor pool. When a mobile application is launched on a mobile device while the mobile device lacks (some of) the necessary sensors, SOUL will utilize the virtualized sensors on the mobile device to enable the execution of the application. From a lower level point of view, this is achieved by the edge that redirects the input/output of the sensors between the mobile device running the application and those providing the sensors. The authors also designed SOUL to virtualize most of the conventional sensors on mobile devices, such that off-the-shelf mobile applications can be executed via the edge without modification.

8) LAVEA: LAVEA is an edge computing platform for real-time video analytics proposed by Yi *et al.* [60]. As their primary concern is the user-perceived latency, the authors designed LAVEA in a way that it schedules tasks in a fine-grained and highly flexible manner.

More specifically, LAVEA analyzes video streams by using the OpenALPR tool [61]. The authors studied OpenALPR and statically divided it into small computational pieces, which are the unit of scheduling and migration. End devices opportunistically offload computations to the edge and execute the remaining part by themselves. They

Table 2 Virtualization Techniques in Edge Computing

Category	Technique	Key Characteristics
System-Level Virtualization	VM-based Cloudlets	Offering efficient computation offloading services
	Unikernels	Utilizing relative small kernels for virtualization purposes
Kernel-Level Virtualization	Linux Containers	OS-level virtualization technique with small images
Virtualization-based Migration	VM-based Migration	Keeping a stably high QoS in the offloading stage
	Container-based Migration	Main stream implementations are based on CRIU and *rsync*
Networking Virtualization	VM-based SDN	Providing highly customizable policies on networking typologies

selectively offload the computational pieces to achieve as low task latency as possible. Several scheduling schemes on the edge were also designed to make LAVEA efficient under different working conditions.

9) Cloudlet-Based Frameworks: The research team led by Dr. Satyanarayanan at Carnegie Mellon University has been conducting research projects on cloudlets [10] since 2009. In this section, we introduce two cloudlet-based edge computing frameworks proposed by his research team.

A cloudlet-based VM provisioning system was proposed by Echeverría *et al.* [62]. Similar to other cloudlet-based systems, their system constructs VMs at the edge via VM synthesis, i.e., by applying an overlay received from the end devices to a base VM image to restore a VM instance. However, the system distinguishes itself by opportunistically leveraging the provisioning tools that are usually found in enterprise environments to automate the construction of VMs, thereby improving the performance of VM synthesis and achieving better flexibility.

OpenStack++ was proposed by Ha and Satyanarayanan [63]. It is a cloudlets' deployment system based on OpenStack [40]. To be more specific, OpenStack++ exploits the extension mechanism of OpenStack to implement the VM synthesis scheme of cloudlets. In this way, OpenStack++ can automatically construct the VM instance after receiving the corresponding VM overlay from the end device and launch it on the underlying OpenStack platform. Additionally, other functionalities for VM synthesis, such as the generation, compression, and transmission of the VM overlay between the end device and the edge, were also fully considered and implemented in OpenStack++.

10) IoT Camera Virtualization: Edge computing has been viewed as the enabling technology for IoT since its early years [12]. Recently, there is a research trend that virtualizes IoT cameras with edge computing to support innovative applications. Jang *et al.* [64] proposed an edge platform for video analytics via an IoT camera. This platform has two key features: 1) the IoT camera is virtualized by the edge to support multiple edge applications simultaneously and 2) the edge can quickly re-configure the IoT camera to adapt to the environmental changes such as a brightness decline, thereby guaranteeing the quality of service (QoS) of video analytics. Notably, each virtual IoT camera was implemented by a camera driver encapsulated in a Docker container, and a hypervisor was implemented to manage all the Docker containers (i.e., the virtual IoT cameras).

III. VIRTUALIZATION TECHNIQUES

Virtualization is the fundamental technique that drives the fast development of cloud computing. As an extension of cloud computing, the emerging of edge computing has not only mitigated the drawbacks of cloud computing but also carried on a large amount of features of cloud, especially in the underlying system paradigms and their techniques. For example, the system-level virtualization (SLV) techniques are widely used in most edge computing projects. SLV techniques provide the platforms that allow multiple users to share the same physic computing resources (CPUs, GPUs, and so on), as if those users occupy the resources individually. In fact, SLV techniques are still in the dominating position and are also effective in most of the current edge computing models where edge nodes are designed to be general-purpose machines for public access.

However, in edge computing, the execution environment is significantly different from that in cloud computing. Therefore, traditional VM techniques in the cloud domain do not fully meet the requirements of the edge domain. For this reason, many pioneer research projects have been conducted in terms of tailoring the traditional virtualization techniques for edge computing, including SLV, kernel-level virtualization, generalized geographical virtualization for service mobility, and software-defined networking (SDN). Table 2 lists the virtualization techniques, which we will discuss in detail in the sequel.

A. Virtual Machines—System Virtualization

VMs, as a hypervisor-based virtualization technique, is one of the popular options for the existing edge platforms where each VM hosts a full-fledged OS and therefore provides an isolated application running environment. Pioneers who work on edge computing have been embracing VMs as their fundamental infrastructure. Cyber foraging, first introduced by Satyanarayanan *et al.* [65], described the technique of offloading resource-intensive tasks from end devices to nearby cloudlets, where the surrogate VMs are one-hop away from the end devices. A reference architecture for cyber-foraging is shown in Fig. 4. By offloading the tasks to the nearby VMs, a mobile device prolongs its

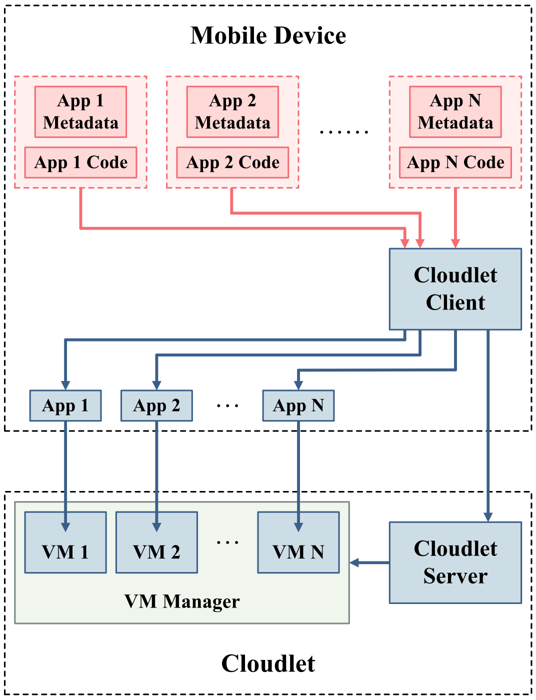

Fig. 4. ***Reference architecture for cyber foraging. The cloudlet client determines the appropriate cloudlet for offloading and connects to the cloudlet according to the application metadata. Once a cloudlet is identified for offloading, the cloudlet client sends the application code and the application metadata to the cloudlet server. The cloudlet server then deploys the application code inside a guest VM under the control of the VM Manager.***

battery life and benefits from resource-intensive services. In addition, placing the edge server one-hop away from the end users will reduce the network latency perceived by the users and greatly improve the user experience.

VM-based cloudlets offer the computation offloading services in one-hop away to the end users. By placing the servers nearby, it achieves significantly lower latency and higher bandwidth than the cloud services. Cloudlets allow an end user to exploit VM technology to rapidly instantiate a customized service on a nearby server and then use that service over LAN for crisp responsiveness of resource-intensive computing tasks. In this paradigm, cloudlet servers are discoverable, generic, and resource-rich workstations that could provide seamless augmentation of computation-intensive real-time applications such as human cognition. This approach is transient customization of cloudlet infrastructure using the VM technology [65].

Two kinds of the existing VM techniques can deliver the VM states to cloudlet infrastructure: VM migration and VM synthesis. The VM migration has been considered as a basic functionality in cloudlet infrastructure. We will discuss VM migration in Section III-C. The dynamic VM synthesis is more appealing because of the fast synthesis speed. Proof-of-concept experiments [65] yielded 60–90-s provisioning time by using this approach without optimization.

VM synthesis divides a VM image into two layers, a base VM image and its VM overlay layer. Base VM image is a base system image that contains the common OS kernels and libraries for supporting different applications. Therefore, we can install and execute different applications via a launch VM image. In contrast, the VM overlay is stored on the device of the end user. When offloading is needed, the end user will transfer the VM overlay to a nearby cloudlet where the corresponding base VM image has already been deployed. By combining those two layers, the original launch VM image is created and ready to offer the service. Fig. 5 shows a typical VM-based cloudlets' synthesis process. In order to binding the VM synthesis technique to cloudlet infrastructure, the original prototype called Kimberley [65] requires an extra management controller residing in both mobile device and cloudlet. Another major issue reported by Simanta *et al.* [66] indicated that the large size of the overlays always involved in VM synthesis at runtime could downgrade the performance.

Ha *et al.* [67] later pointed out that the process of VM synthesis contains three time-consuming and serialized steps, including VM overlay transmission, VM overlay decompression, and decompressed VM overlay deployment. They tried to accelerate VM synthesis by reducing the size of VM overlay via aggressive deduplication and bridging the semantic gap between the VM and the guest OS. Moreover, they tried to modify the synthesis pipeline to accelerate the launch of the VM image. Messinger and Lewis [68] raised the idea of application virtualization to meet the requirements in resource-constrained cyber foraging environments. Then, toward a more practical point of view, Lewis *et al.* [69] further presented several strategies for cloudlet-based cyber foraging. They regarded application virtualization as a method to decrease payload size. At runtime, system calls from an application are intercepted and redirected

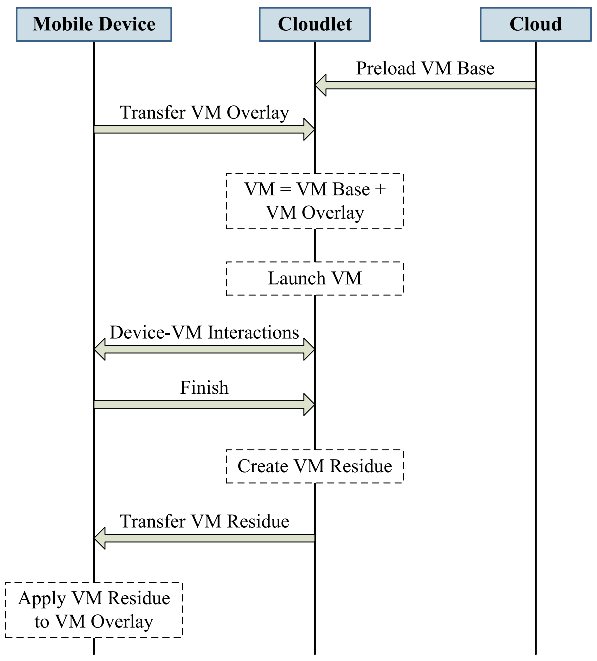

Fig. 5. ***Dynamic VM synthesis in cloudlets.***

to the resources inside the virtualized application, thereby reducing the transmission cost. A series of alternative provisioning strategies was also proposed by Lewis *et al.* [69], such as cached VMs, on-demand VM provisioning, and cloudlet push.

Unikernels are the other type of virtualization techniques that fit for edge computing. Compared with VMs, which are running on large general-purpose OSs, unikernels with smaller customized kernels draw much research attention. Jitsu [70] used unikernels to achieve secure multitenant isolation on embedded infrastructures. Indeed, unikernels are promising for resource-constrained edge computing environments because of their features of low memory footprints, compact size, and memory-safe VMs written in a high-level programming language. The responsiveness and booting speed of unikernels can be further improved by optimizations, such as using shared memory channels. Furthermore, small memory footprints allow unikernel-based system to be more power-efficient than the traditional VMs. With the help of Type-I hypervisor, strong isolation can be achieved similar to other OS-level virtualizations, such as LXCs.

B. Linux Containers—Kernel Virtualization

LXC is an OS-level virtualization technique. It does not require full isolation of different OSs. Instead, it allows all containers to share an OS kernel. Each container has a virtualized kernel for itself, while the underlying system has only one copy of the kernel. Since the kernel is virtualized and shared among containers, the image size is much smaller than those of VM images. By utilizing kernel features such as cgroups and namespaces, a container can provide an environment as similar as possible to that of a VM without the overhead that comes with running a separate kernel and simulating all the hardware [31].

Applying container techniques to the edge environment is a natural trend because of the facts of rapid construction, instantiation, and initialization of virtualized instances [71]. In addition, using the containers can achieve high application/service densities via small dimensions of the virtual images, which allows more instances of containers to be deployed on the same host [29]. As a mature technology, reasonably using containers on edge is the key challenge. A series of evaluations has been conducted to explore the feasibility and maximize the performance. Ismail *et al.* [72] evaluated the containers as an enabling technology for deploying an edge computing platform. They provided four feature metrics covering the entire container life cycle: deployment and termination of services, resource and service management, fault tolerance, and caching capabilities. Later, Morabito [71] evaluated the performance of container-based virtualization on IoT devices on the edge. They conducted more practical experiments on Advanced RISC machine (ARM)-based IoT end devices (Raspberry Pi). Performance evaluation on the CPU, memory, disk I/O, and network shows that container-based virtualization can represent an efficient and promising way to enhance the features of edge architectures.

C. Service Mobility—Geographical Virtualization

End users benefit from edge services by offloading their computation-intensive tasks to nearby edge nodes. However, when the end user moves away from the nearby edge server, the QoS of edge services will significantly decline due to the interruption of the network. Keeping a constant high QoS in the offloading stage is one of the key challenges in edge computing. Ideally, when the end user moves, the services on the edge server should also migrate to a new nearby server. Therefore, efficient live migration is vital to enable the mobility of edge services in edge computing environments. There are several approaches proposed for live migration. We classify these approaches according to their underlying virtualization techniques, such as VM-based migration, container-based migration, and process-based migration.

1) VM-Based Migration: Based on the work of VM synthesis [65], Ha *et al.* [73] proposed VM handoff as a technique for seamlessly transferring VM encapsulated computations to an optimal offload site as users move. Machen *et al.* [74] proposed to organize VM images into two or three layers through pseudo-incremental layering, and then, the layers were synchronized by using the rsync incremental file synchronization feature. Chaufournier *et al.* [75] and Teka *et al.* [76] used multipath TCP to transfer VM images over different paths in parallel. Bittencourt *et al.* [77] elaborated on the roles of VMs in fog computing cloudlets. In particular, they establish a fog architecture to support VM/container migration. In recent, Machen *et al.* [78] presented a layered framework for migrating active service applications that are encapsulated either in VMs or containers in mobile-edge cloud (MEC) settings.

2) Container-Based Migration: Qiu *et al.* [79] explored the potentialities of LXCs to be applied to cloudlets and adopted a container migration technique based on Checkpoint/Restore in Userspace (CRIU) and the rsync command. Ma *et al.* [80] proposed an efficient migration method by sharing most of the storage layers between different edge nodes and incrementally transferring the runtime memory to reduce the downtime of live migration. It was shown that the migration downtime could be reduced by more than 56% compared to VM synthesis. We notice that the container migration in edge computing environments is still an under-explored area and has many research potentials.

3) Process-Based Migration: Process-based migration is one of the key techniques in distributed systems in large data centers, especially important for task scheduling. Early days, Milojicic *et al.* [81] generated a full picture of process migration by summarizing the key concepts and giving an overview of the most important implementations. Now, many mature tools, such as CRIU [82], have been widely used by both industry and academia. However, current process migration paradigms assume a shared or distributed networking storage [82], where all the migration targets have an identical view of the underlying file system. This is a valid assumption for large data centers in the cloud, but no longer suitable for edge computing environments. Therefore, we anticipate that efficient storage sharing on edge computing environments is the key challenging obstacle for achieving feasible process migration in edge computing.

D. VM-Based SDN—Networking Virtualization

Another domain of virtualization in edge computing is networking virtualization. Emerging applications on edge computing, such as cognitive assistance programs, usually require high bandwidth, low latency, as well as the dynamic networking configurations. Configurable networking becomes much more challenging on the edge that is on the last mile of network and faces various kinds of mobile applications. SDN provides highly customized network policies that could meet the requirements of the ever fast growing complexity of networking typologies and routing schemes. SDN is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it perfect for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions.

VM techniques also play an important role in the SDN architecture. NetVM [83] retrofits the VM platform to virtualize the network interface card (NIC), which achieves far better performance than the traditional VM platforms for single, nonvirtualized host. It drives the NIC, forwards the packets inside a customized hypervisor, and allows each VM to share the networking resources through a huge page in the hypervisor, thus reducing the overhead of copying packet data across different layers of the system. The scheme allows zero-copy delivery of data from NIC to VM, which significantly improves the throughput of the data plane in the SDN architecture.

One limitation of SDN-based networking solutions is that it requires hardware support. There are still network devices that are not SDN-enabled at the edge of networks. These SDN-based solutions cannot be deployed to these devices. Updating network devices can solve this problem, but sometimes, it costs too much for updating those devices.

E. Virtualization Challenges for Edge Computing

Virtualization techniques play an important role in edge computing for its capability of providing high scalability, easy-to-deploy frameworks, and the stable view of the network and computing environment, while the end users are rapidly changing locations. We notice that the traditional research and industry applications of virtualization techniques on the cloud platforms are valuable experiences as a starting point for the edge computing environment. However, differences between the cloud and the edge, such as the networking conditions, hardware resources, and maintenance costs, impose new challenges for those techniques to be useful on the edge.

With many of the assumptions for the cloud environment no longer being valid in the edge, more efficient virtualization solutions need to be explored. For example, efficient service migration in the edge computing environment is still an unexplored area and has many research potentials, while networking virtualization, such as SDN, needs to be revised for its compatibility with legacy devices and easy deployment across the large geographical distribution for the edge. Virtualization techniques need to be more energy and performance efficient with relatively low hardware requirements. When utilized in a wide area network (WAN) environment, virtualization techniques also need to be bandwidth efficient to resist the relatively poor network conditions.

IV. PLACEMENT AND SCHEDULING

In a long period of time, cloud-based applications have enriched user experiences by offering powerful, reliable, manageable, and cost-saving services. The computing capacity of end devices no longer has the confinement. The end devices can reduce their workload and prolong their battery life by offloading the computation-intensive tasks to a remote cloud. However, in recent years, the computation-intensive tasks, such as machine learning-based applications and AR, always require an amount of data supply. Offloading such tasks to remote cloud, apparently, is undesirable due to the long distance between the end devices and cloud, and the service downgrading caused by unpredictable network conditions. As a consequence, a lot of research has been conducted to explore the optimization opportunities in edge computing paradigm. Interestingly, most of VM placement and scheduling problems can be formulated as nonlinear optimization problems. However, such optimization problems have several nontrivial solutions and their solution spaces are huge. We summarize the problems in VM placement and scheduling in an edge computing environment by their objective functions in Table 3 and discuss in the following.

A. Optimizing Installation Expenditure

End devices frequently communicate with their assigned edge node and continuously exchange various types of data over the time. Therefore, installing the edge nodes to the suitable locations becomes a primary research problem for service providers (SPs). Although user experiences could be improved by increasing the number of edge nodes, in fact, it is impractical because the budget of SP is limited. Moreover, edge node installation expenditures not only include the edge server facilities but also the fee of subsequent maintenance and management. To the

Table 3 Placement and Scheduling

Optimizing Objective	Model	Technique	Reference
Installation Expenditure	Mixed-Integer Non-Linear Programming(MINLP)	ILP-based Algorithm/Solver	[84]–[86]
Response Time	K-Cloudlets Placement Optimization Problem	K-Mean, Min-Max Solver	[87]–[90]
Energy Consumption	Bin Packing Problem	Particle Swarm Optimization(PSO)	[91], [92]
Revenue of Service Provider	General Optimization	Auction-based Profit Maximization semi-Markov decision process(SMDP)	[93]–[95] [96]

best services, the budget of SP therefore should be taken into consideration. Two common factors of edge node installation expenditures are site rentals and area-specific computation demands. The site rentals can significantly impact on the deployment expense since they are various in terms of different geographic locations from big city to rural. We also notice that there exists a positive correlation between the site rentals and potential computation demands. For example, in the high population density area, site rentals and computation demands are both high. This untreated problem was first discussed by Ceselli *et al.* [84], where they brought the original insights on the planning of edge nodes without any user level or VM mobility considerations. Their static planning reveals a link-path mixed linear relation between edge node installation cost and QoS. A more concrete edge node deployment problem was discussed in [85]. In the context of mobile-cloud computing (MCC) with a centralized data center, end devices offload the tasks to a data center, which causes the high end-to-end (E2E) delay. Although cloudlet can potentially mitigate this problem, how to place cloudlets to minimize the E2E delay as well as the deployment cost has not been addressed. In fact, the E2E delay is hard to be avoided in implementation because the big data aggregation tries to gather data from the numberless users. Mondal *et al.* [86] explored the edge node deployment problem over optical access networks. They proposed a hybrid cloudlets' placement framework CCOMPASSION over the existing TDMPON infrastructures to prevent under- or over-utilization of the resources. In essence, most of the edge node deployment optimization problems can be formulated as a mixed-integer nonlinear programming (MINLP) problem with the form of

$$\min \ \mathcal{E}_{\text{total}} = \sum_{p_i \in \mathcal{I}} p_i s_i + \gamma \sum \mathcal{C}(\lambda, t) \tag{1}$$

where $\mathcal{E}_{\text{total}}$ is the total cost in CEC, $\mathcal{I}$ denotes as the set of possible locations of edge node, and s_i is the fixed cost of an edge server at p_i. Here, $\mathcal{C}(\lambda, t)$ is a generalized constraint function that usually uses time t and task arrival rate λ as parameters with cost coefficient γ. For example, in [85], the constraint function is defined as the user requested E2E delay.

B. Optimizing Response Time

The emerging of cloudlet techniques fills the gap between the remote cloud and end devices. Cloudlet refers to a trusted, resource-rich computer or cluster located in the area close to a variety of proximity devices. Hence, one can offload a heavy computational task to nearby VM and regale a low-latency access of the rich computing resource. One of the mainstream research problem of VM scheduling and placement in CEC paradigm is focusing on optimizing access delay and minimizing response time. This kind of studies is nontrivial and challenging due to: 1) the end devices frequently communicate with the different VMs in a nonstatic manner, and therefore, it is barely possible to predict their movements; 2) the number of end devices is in a big range over the time, and although it may have some regularity in short period of time, their number is unpredictable from long-term perspective; and 3) the tasks have dynamic workload, and hence, it is not feasible for scheduling simply by their execution time. In general, it is impractical to think that the VM can provide as much computation resource as the user expected in edge computing. In fact, the VM is resource limited due to economic reasons and physical constraints.

To keep the promise of the best QoS and reducing the resource contention, Cardellini *et al.* [97] intuitively mentioned that one could take advantage of the general multiuser three-tier mobile CEC platforms via delivering the tasks to both cloud and cloudlets. We could assume that task offloading is a non-cooperative game among selfish users, and as a result, the optimal solution can be achieved by solving a generalized Nash equilibrium problem (GNEP) from the game theory. Although their work is almost methodology that targets at determining which tasks should be properly delivered to which cloud tier, it inspires researchers to explore using the shared resource to improve the edge computing performance and the user experiences. Verbelen *et al.* [98] presented a novel edge computing architecture, where the applications were managed in the component level. The application components can be independently distributed among the VM and executed in a parallel manner.

In recent years, cloudlets have been considered as the best practically suited technique for wireless metropolitan area networks (WMANs) [87]–[90]. This is due to the fact that the WMAN is deployed to a high population density area with adequate computation demands, and hence, the economic benefits of using cost-efficient edge VM can be amplified tremendously. Another significant fact is that the price of using this adorable technique could be low, since it has potentially large user groups. Jia *et al.* [87] generalized a K Cloudlets Placement

Optimization Problem, that is, given the limited number of edge nodes in WMAN, they are trying to find the optimal VM placement strategy and the effective user-to-cloudlet assignment in a WMAN, therefore reducing the average wait time of offloaded tasks. They point out that the traditional VM placement follows a classical principle that the VM operators always assign tasks to the closest edge VM, which causes the uneven distribution of tasks and computing resource contention. The overloaded tasks lead to a poor quality of experience (QoE). Jia *et al.* [87] systematically analyzed the VM placement in WMAN and introduced the $M/M/c$ queue for balancing workload and average response time. Moreover, Xu *et al.* [88] proved the K Cloudlets Placement problem to be an NP-hard and proposed a heuristic algorithm to ensure the minimized average edge node access delay. To solve the problem in WMAN-based VM (cloudlets) placement and offloading, Jia *et al.* [90] presented an optimization algorithm to reduce the average response time of offloading tasks over the entire system. They noticed that the VM placement problem followed the wooden barrel principle, that is, the system performance depends on the longest task response time. When an unexpected number of user requests overwhelms a VM, the response time dramatically increases, and as a result, the QoS is downgraded. To avoid such situation happening in WMAN, they first formulated a VM-based load balancing problem in a given network graph. Then, they kept monitoring the task response time and dynamically redirecting the task to achieve workload balance of each VM, so that minimizing the maximum response time of tasks in edge VM when offloading surge occurred. Generally, the WMAN-based VM optimization problems are planning to minimize the system response time (SRT) by using the following model:

$$\min \bar{t} = \frac{1}{n}\sum_{i=1}^{n} t_i \tag{2}$$

where $t \sim (G, A, W, D, T_c, f(\lambda), B, \mu, c)$. Here, G is the given network graph. A refers to workload. T_c is a predefined time threshold. The delay between AP and end device is D. The task queue time has the form of Erlang's formula $f(\lambda) = ((M/M/c\,\text{queue})/(c\mu - \lambda))$ with c VMs and task arrival rate λ.

C. Optimizing Energy Consumption

Energy consumption should never be neglected because it is an essential metric in QoS evaluation. Especially in the CEC paradigm, all connected live users are equipping with a portable devices but having limited battery supply. Although edge nodes provide a perfect breeding ground where the end devices can offload the energy-hunger tasks that always accompany with complex data or require high-frequency communication with edge node, those applications use approximately 50% of the total energy consumption, while the proportion of under-utilized service capacity is only from 5% to 25% [99]. Kumar and Lu [100] tried to answer the question whether offloading computation can save energy. They gave a positive answer to this question. The offloading on CEC can potentially save energy for end devices. In the early literature, Gelenbe *et al.* [101] addressed the energy consumption problem from the pure technical aspect. Task offloading could happen in either remote cloud or in edge nodes according to average task response time, task arrival rate, and average energy consumption. Later, Wang and Xia [102] introduced the power consumption model of physical machines (PMs) for analyzing the energy consumption in the big data center.

Recently, researchers pay more attention to the energy consumption of VM placement. VM placement can be simply regarded as a process to find the optimal network path to allocate VM, and therefore, the task can be quickly executed and energy usage can be reduced. In such a problem, the energy cost mainly depends on: 1) the CPU or GPU intrinsic energy cost of processing each offloading task; 2) the number of alive VMs of service; 3) the number of idle servers; and 4) the transmission distance and the frequency of VM consolidation. In this kind of literature, researchers formulate a series of integer linear programming (ILP) problems with the constraints of: 1) the limited offloading capacity; 2) the limited number of edge nodes; and 3) the upper bound of distance between the end device and the edge node. Kaur and Bawa [103] figured out that energy-aware VM placement was an NP-hard, and the following literature then considered it to be a bin packing problem with different dimensions. The first industry-level VM placement problem with energy concerns is from IBM [104] who built a collaborative manager system to minimize the service-level agreement (SLA) violations and total power consumption. Kaur and Kalra [105] focused on the topic of energy cost when deploying the VM to a PM occurred. By reducing the number of live servers and redundant migration operations, their experiments showed the superiority in comparing to the industry-level standard algorithm, modified best fit decreasing (MBFD) algorithm [106].

There is another trend that the energy-aware VM placement optimization approaches can be solved via particle swarm optimization (PSO), for example, in [91] and [92]. In order to fully utilize the PSO, they concluded that: 1) the CPU utilization and energy consumption were in linear relation and 2) the idle servers consumed more than 60% total energy [107]. Li and Wang [92] additionally proposed the method to find out a placement scheme that could reduce the total energy consumption and keep the access delay in a reasonable range. Moreover, WMAN becomes a good study case since it has a plenty of wireless APs and countless connected end devices. Ren *et al.* [108] overcame the energy drain by solving an equivalent minimum dominating set problem in the graph theory and, therefore, choosing the minimum number of edge nodes and cutting down the energy cost.

D. Optimizing Revenue of Service Provider

SPs play an important and irreplaceable role in the CEC and MEC paradigm. The primary concerns of SP include how to reduce the operational cost, boost the market share, and enlarge the scale of its enterprise. Given the above-mentioned facts, an effective admission control scheme should be issued by SP to ensure the QoS under the constraints of computing resource, network bandwidth, and the budget of PS.

For a long time, admission control models from SP are being underestimated. Admission control can be expressed as many different models. Hoang *et al.* [96] proposed a prototype of dual-class user-level admission model. To optimize an admission control model, they used the semi-Markov decision process (SMDP) to maximize the revenue of SP. The nonlinear objective function tries to allocate the resource such as network bandwidth and edge VM to users with different priority levels (member or nonmember) and ensure QoS requirements. By following Hoang *et al.* pioneer work, a series of SLA-driven VM placement approaches [109], [110] had been studied in the generalized cloud. Katsalis *et al.* [111] investigated the problem of multiple MEC-IaaS providers with a mixture of different time-sensitive tasks for scheduling. They formulated a joint optimization problem based on a decision process where the MEC-IaaS providers aimed to maximize their revenue. It is also worth to mention here that Katsalis *et al.* [111] derived a fast VM placement method by taking the advantage of Lyapunov optimization; therefore, there was no necessity to know the task arrival rate in advance.

Another track of discovering the maximization SP revenue problem is through employing an auction mechanism. In essence, the auction-based profit maximization approaches, such as [93]–[95], aimed to find an efficient VM pricing model. Zhang *et al.* [93] suggested an auction mechanism by formulating the pricing model of resource allocation process on VMs. In addition, Zheng *et al.* [95] developed an optimization model for the spot pricing system. Furthermore, a three-hierarchical-cloudlet-tier (field, shallow, and deep) design had been explored in [112]. This paper is inspired by the equilibrium pricing models and users can bid for resource from edge VM according to their demands.

E. Optimizing Others

The optimization problems in VM scheduling and placement are not limited to the topics that we highlighted earlier. Xia *et al.* [113] focused on maximizing the system-level throughput. They defined an online request throughput maximization problem to determine the incoming task whether should be accepted or rejected according to the admission cost threshold. Sun and Ansari [114] and Sun *et al.* [115] proposed the VM migration strategy to optimize the tradeoff between the migration gain and the migration cost. They quantitatively analyzed the process of VM migration and generated the migration cost-gain models. A VM placement problem in data centers with multiple deterministic and stochastic resources was discussed in [116]. They found an approximation solution of multidimensional stochastic VM placement (MSVP) to reduce the number of required servers.

V. SERVERLESS MANAGEMENT IN EDGE COMPUTING

Serverless, also known as Function-as-a-Service (FaaS), is an abstraction layer for applications built on the top of services running in virtualized environments. FaaS is hardly a success without the advancements in lightweight virtualization techniques and wide adoption of cloud computing.

Edge computing platforms are embracing serverless services to provide a unified abstraction of applications that can run anywhere at anytime. Most popular serverless services, such as AWS Lamda, Azure Function, and Google Cloud Functions, are provided as cloud services. However, a wide variety of serverless services can be deployed both in cloud and on-premise, such as Apache Openwhisk, Knative, OpenFaaS, and Dispatch. There are also dedicated edge computing platforms incorporating serverless as the default or alternative computing paradigm, such as KubeEdge and OpenEdge. As a newly emerging technique, we discuss its management in this section. The serverless management is, in essence, the virtualization instance management, as all the serverless frameworks are built based on at least one or two virtualization techniques.

1) *Life Cycle Management:* The common workload type for FaaS is event driven. An event triggers the launching of a serverless instance (container or VM) to fulfill certain types of task handling, report the results, and terminate or recycle itself. When massive events hit the same function on a serverless platform, the same type of serverless instances will auto-scale to adapt to the incoming workload. As a result, serverless service usually has to limit the computational resource and memory that a single instance can be allocated. Therefore, life cycle management is crucially important for serverless computing systems to lift the limits in terms of improving resource utilization and latency mitigation. Sprocket [117] is a video processing system built on the AWS Lambda serverless cloud infrastructure. The Sprocket relies on coordinator as a control plane for life cycle management of Lambda instance. Sprocket also combines speculative execution and proactively execution to make sure that straggler is mitigated.
2) *Cost-Oriented Management:* Unlike traditional computing service, serverless expense depends on the requested functions' executing time and the amount of memory allocated to that function. Therefore, an optimized pricing model of serverless computing

is needed. The two major categories of optimization problems in serverless are function fusion and function placement. Functions' fusion is a nontrivial research problem. Merging multiple functions together can greatly reduce the transition cost and provide the flexibility for function placement. What happens underneath is the consolidation of resources at the VM level, which brings down the total cost. Costless [118] targeted at balancing between price and execution time by proposing two techniques, function fusion and placement, based on the variance of the pricing models of AWS Lamda (cloud side) and AWS Greengrass (edge side). Their solution relies on a cost graph model to formulate the problem as a constrained shortest path problem for finding the best solution that can make a tradeoff between latency and budget. For example, Feng *et al.* [119] built a serverless computing system for neural network training and also studied how to minimize monetary cost and maximize the performance–cost ratio.

Although the FaaS finds its place in CEC, the research study on FaaS optimization still remains unexplored. Now, we illustrate some open serverless management challenges at edge.

1) *Function Deployment:* In serverless edge computing, function deployment that needs more attention as resource at the edge is limited, which means that we cannot even pull a huge size container image or build such image at edge. There is also a strong need to make function management at the edge side agnostic to the cloud provider. For example, Aske and Zhao [120] proposed an edge computing client that supported multiple serverless SPs.
2) *Function Invocation:* In order to invoke a function [121], the naive way is to instantiate the container or VM with an image containing the code of the function, which is known as cold start. This approach is efficient on resource usage. However, the latency or responsiveness is hard to guarantee during the startup process. To address this issue, the warm start is proposed in which a pool of containers or VMs will be ready ahead of invocation. In the cloud-edge environment, pre-warming container or VM at edge pays a higher cost than in the cloud. Because resources are limited at the edge side, one may not be able to pre-warm all kinds of runtime environments in container or VM. However, an edge computing system has its advantage to leverage collaborations between edge node and cloud node or among nearby edge nodes.
3) *Function Chaining:* Function chaining is necessary as it in nature captures the relationship of functions among any complex applications. It is a feature that will be requested more often since more complicated applications run in a serverless manner. SAND [122] was built to provide lower latency and better resource efficiency for a serverless computing system. SAND approaches the function chaining by a workflow system supported by a hierarchical message bus system. If a workflow will invoke function across the hosts, the output of the previous step will be published into the global message queue; otherwise, the output will remain in the local message queue.
4) *Function Composition and Decomposition:* Function composition means that we combine or fuse multiple functions into a single function, while function decomposition means that we separate a complex function into smaller functions. Given the imbalance of resources owned between edge and cloud, we foresee that function composition and decomposition will play an important role in performance tuning for serverless spanning on edge and cloud.

VI. VIRTUALIZATION AND SECURITY

Similar to cloud computing systems, security plays a big role in edge computing systems. Because edge nodes store users' sensitive data (e.g., video streams and location data) and offer critical services (e.g., security surveillance and automation control in factories), security becomes even more important in edge computing. Moreover, the applications of virtualization techniques in edge computing, though bring lots of benefits for security, also introduce potential security vulnerabilities. Therefore, security is another hot topic in edge virtualization management. In this section, we will cover challenges and works related to security problems in the context of edge virtualization. These works fall into two categories: 1) works that use virtualization techniques to solve security and privacy problems and 2) works that address potential vulnerabilities in the virtualization ecosystems in edge computing. For the second group, we will first introduce the potential security issues in virtualization under edge computing scenario. In the end, some of the countermeasures to address these security issues are introduced.

A. Apply Virtualization for Security Reason

Because edge computing shares one edge node among different services, it is necessary to isolate resources and OS among services. Furthermore, because of the needs to protect private user data, edge nodes need to isolate service data between each other. Due to the nature of isolation in VMs, virtualization becomes one of the most straightforward approaches to guarantee security and privacy. There are three types of isolation in edge computing: 1) the isolation of user data; 2) the isolation of OSs; and 3) the isolation of resources. We introduce these three isolations separately in detail.

1) User Data Isolation: Protecting users' data from unauthorized access is one of the critical features in edge platforms. Virtualization techniques offer easier ways to edge administrators to manage the privacy of users' data. Because processes in one VM cannot directly access data in other VMs, edge administrators just simply assign a VM to each task in an edge node and run these tasks inside their own VM environment. EdgeCourier [123] is one such example using virtualization techniques to keep users' privacies. EdgeCourier is an edge platform for documentation synchronization. Each synchronization is assigned with an edge-hosed personal service (EPS), which is implemented as a unikernel. Because user tasks are wrapped in EPS, EdgeCourier can schedule user services by simply manipulating EPS.

2) OS Isolation: Virtualization techniques not only make service management easier but also protect tasks running from failures of other services. By isolating services into different OSs, even if one service is crushed, it only ruins its own virtual running environment. As long as the host is in good status, other services will continue running without any interference. LAVEA [60] is a video analysis application in edge computing. It uses Docker to segregate different videos analyzing processes to different containers. This design helps both easy management of video analyze services and service protection from the unexpected errors of other services.

3) Resource Isolation: Allocating resources among services is a hot topic in edge computing. By assigning resources to VMs, virtualization techniques transform service resource allocation problem to VM resource allocation problem. Allocating resources to VMs is more straightforward since edge administrators do not need to consider privilege issues and race conditions between services. ParaDrop [48] is a multitenant edge platform that uses virtualization techniques to achieve the isolation of resource allocations. Jang *et al.* [64] propose an IoT camera virtualization architecture in edge computing to achieve camera allocation isolations.

B. Security Issues in Edge VM

Although virtualization techniques bring benefits to edge computing security, these techniques also introduce security challenges to edge computing platforms. In this section, we will introduce the security challenges in edge computing virtualization.

The first security challenge of virtualization is the security of the host machine, i.e., the edge node. Because the host machine can configure every detail of the VMs, a compromised host machine is a disaster for all VMs. Unlike cloud servers that are in the secured places, edge nodes are not always physically secured. It is easier for advisories to physically access edge nodes and compromise the edge nodes by exploiting kernel vulnerabilities. Moreover, since services may migrate among several edge nodes and these edge nodes are heterogeneous in terms of the owner and the platform, the attack surface for a service is larger than the one in could computing. Another security threat inside of the edge node comes from the residence of malicious VMs. Adversaries can create a fake edge service in an edge node and perform side-channel attacks inside the edge node trying to infer sensitive data from other VMs. Therefore, securing the host machine is the first challenge of edge computing systems with virtualization enabled.

The next security threat of virtualization is from the network. As we mentioned earlier, edge nodes usually migrate VMs to other edge nodes. During the transformation of these VMs, adversaries can perform attacks to interference VM migrations. Moreover, these VMs may face the risks of leaking their contents to adversaries. For example, if an adversary successfully launches Man-in-the-Middle (MitM) attacks on a communication channel between two edge nodes, it can get the entire content or image of a VM. Another attack example is the phishing attack, in which adversary deploys a fake edge node and waits for other edge nodes falsely migrate VMs to them. Edge nodes may also face denial-of-service (DoS) attack from the network. Because initializing a VM for one service costs many resources and edge nodes have striker resource limitation than cloud server, it is easier for adversaries performing DoS attacks in edge computing.

C. Countermeasure

With the respect of the above-mentioned security challenges, researchers proposed approaches to secure the VM-based edge computing platforms. In this section, we will cover the existing and potential countermeasures for these security threats. Specifically, we introduce countermeasures from the following four security techniques: 1) identity and authentication mechanism; 2) intrusion detection system (IDS); 3) trust management; and 4) fault-tolerance mechanism.

1) Identity and Authentication Management: One of the best ways to address MitM attacks and phishing attacks is to use identity credentials from a trusted third party to identify trusted edge nodes. Two edge nodes need to get credential from the other side and verify its identity whenever they are going to establish a communication channel. In this case, adversaries cannot pretend to be other genuine edge nodes. Echeverría *et al.* [124] proposed a solution for building trusted identities in disconnected environments, which is suitable for edge computing scenario. Besides verifying identities, using classical encryption schemes can also help secure the message from adversaries. Identity-based encryption (IBE) [125], for example, is a great encryption scheme for edge nodes. In IBE, a user's public key can be directly generated from users' own identities. Therefore, there is no public key exchange phase in IBE. This saves lots of time and simplifies the key distribution protocol. Attribute-based encryption [126], [127] is another useful encryption scheme in which a user's public key is computed according to that user's attributes. Both IBE and ABE can be applied to VM-based edge computing platforms for establishing encrypted communication channels between edge nodes.

2) Intrusion Detection System: An effective way to protect edge nodes from attacks, such as DoS attack, is deploying IDSs on edge nodes. An IDS is a system that monitors active attacks and abnormal activities in a network. It analyzes the signature of traffic or abnormal events to decide whether there are any attacks inside the network. Lin *et al.* [128] proposed an IDS for edge computing. Because edge nodes may have limited resources, resource allocation in IDS is not a trivial problem, together with a system for efficient and fair resource allocation within the IDS. Arshad *et al.* [129] proposed COLIDE, a framework for intrusion detection in machine-to-machine (M2M) networks with hosts and edge nodes that have low energy and communication cost. Because edge nodes are usually located in M2M networks, COLIDE is very suitable for edge computing scenario. Furthermore, IDSes in cloud computing [130]–[132] are also good candidates for edge computing.

3) Trust Management: Before running services on an edge node, the owner of the services needs to know whether the edge node is trustworthy. Here, trustworthy does not only mean the identity of the edge node, which can be verified by authentication mechanisms introduced earlier, but also means the reputation of the edge node, that is, users need to be sure that edge nodes perform as they claim before scheduling services. Airbox [49] is a platform for onloading edge services. Airbox leverages Intel SGX [38] to secure the integrity and privacy of edge services, which can be wrapped by a VM, on edge platforms. Hussain and Almourad [133] proposed a trust management scheme for users to choose edge nodes with high reputation. Some of the trust management works in cloud computing [134]–[136] can also be introduced in edge computing.

4) Fault-Tolerance System: Making edge nodes more robust and resilient to errors is always a good direction for defending edge nodes. Besides the security mechanisms applied to edge nodes, they themselves should be fault tolerance and fast recovery from the fault status. Satria *et al.* [137] proposed the recovery schemes for overloaded edge nodes. The basic idea behind these recovery schemes is offloading neighboring edge nodes. Aral and Brandic [138] introduced a method for assessing the probability of edge service interruption. The service interruption probability is inferred from historical failure logs of all the edge nodes.

VII. CONCLUSION

In this paper, we shed light on the industrial and research projects on VM management in edge computing. As an extension of cloud computing, edge computing relies on virtualization to deliver its services. Nevertheless, the distinguishing traits of edge computing make it a different story from that of cloud computing in establishing virtualization mechanisms for the entire system. We hence introduce the engineering and research trends of achieving efficient VM/container management in edge computing. More specifically, we elaborate on: 1) the virtualization frameworks for edge computing from both the industry and the academia; 2) the virtualization techniques tailored for edge computing; 3) the placement and scheduling algorithms optimized for edge computing scenarios; and 4) the security advantages and issues that virtualization brings to edge computing.

We envision that edge computing will keep evolving in the future, bringing more opportunities, and imposing more challenges on VM/container management. More effort from both the industry and the academia should be put into this promising research topic in the future. ■

Acknowledgments

The authors would like to thank all the reviewers for their helpful comments.

REFERENCES

[1] Amazon. (2019). *Elastic Compute Cloud*. [Online]. Available: https://aws.amazon.com/ec2/
[2] (2019). *Amazon Web Services (AWS)—Cloud Computing Services*. [Online]. Available: https://aws.amazon.com/
[3] Microsoft. (2019). *Microsoft Azure*. [Online]. Available: https://azure.microsoft.com/
[4] Google. (2019). *Google Cloud Including GCP & G Suite| Google Cloud*. [Online]. Available: https://cloud.google.com/
[5] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[7] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.
[8] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014. doi: 10.1145/2677046.2677052.
[9] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, 2014, pp. 1–8.
[10] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.
[11] M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: At the leading edge of mobile-cloud convergence," in *Proc. 6th Int. Conf. Mobile Comput., Appl. Services (MobiCASE)*, 2014, pp. 1–4.
[12] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC)*, New York, NY, USA, 2012, pp. 13–16. doi: 10.1145/2342509.2342513.
[13] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.
[14] E. Zeydan *et al.*, "Big data caching for networking: Moving from cloud to edge," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 36–42, Sep. 2016.
[15] H. Sun, X. Liang, and W. Shi, "Vu: Video usefulness and its application in large-scale video surveillance systems: An early experience," in *Proc. Workshop Smart Internet Things (SmartIoT)*, New York, NY, USA, 2017, pp. 6:1–6:6. doi: 10.1145/3132479.3132485.
[16] J. Wang, J. Pan, and F. Esposito, "Elastic urban video surveillance system using edge computing," in *Proc. Workshop Smart Internet Things (SmartIoT)*, New York, NY, USA, 2017, pp. 7:1–7:6. doi: 10.1145/3132479.3132490.
[17] R. Xu *et al.*, "Real-time human objects tracking for smart surveillance at the edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[18] W. Zhang, J. Chen, Y. Zhang, and D. Raychaudhuri, "Towards efficient edge cloud augmentation for virtual reality MMOGs," in *Proc. 2nd ACM/IEEE Symp. Edge Comput. (SEC)*, New York, NY, USA, 2017, pp. 8:1–8:14. doi: 10.1145/3132211.3134463.
[19] Y. Li and W. Gao, "MUVR: Supporting multi-user mobile virtual reality with resource constrained edge cloud," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 1–16.
[20] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service Entity placement for social virtual reality

applications in edge computing," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 468–476.

[21] Q. Liu, S. Huang, and T. Han, "Fast and accurate object analysis at the edge for mobile augmented reality: Demo," in *Proc. 2nd ACM/IEEE Symp. Edge Comput. (SEC)*, New York, NY, USA, Oct. 2017, pp. 33:1–33:2. doi: 10.1145/3132211.3132458.

[22] P. Ren, X. Qiao, J. Chen, and S. Dustdar, "Mobile edge computing—A booster for the practical provisioning approach of Web-based augmented reality," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 349–350.

[23] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, "An edge-computing based architecture for mobile augmented reality," *IEEE Netw.*, pp. 12–19, Jan. 2019. doi: 10.1109/MNET.2018.1800132.

[24] S. Zhang, H. Yan, and X. Chen, "Research on key technologies of cloud computing," *Phys. Procedia*, vol. 33, pp. 1791–1797, Jan. 2012.

[25] J. E. Smith and R. Nair, "The architecture of virtual machines," *Computer*, vol. 38, no. 5, pp. 32–38, May 2005.

[26] National Vulnerability Database. (2018). *CVE-2018-9862*. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2018-9862

[27] (2019). *CVE-2019-5736*. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-5736

[28] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Philadelphia, PA, USA, Mar. 2015, pp. 171–172.

[29] R. Morabito, J. Kjällman, and M. Komu, "Hypervisors vs. lightweight virtualization: A performance comparison," in *Proc. IEEE Int. Conf. Cloud Eng.*, Tempe, AZ, USA, Mar. 2015, pp. 386–393.

[30] A. M. Joy, "Performance comparison between Linux containers and virtual machines," in *Proc. Int. Conf. Adv. Comput. Eng. Appl.*, Mar. 2015, pp. 342–346.

[31] R. Rosen, "Linux containers and the future cloud," *Linux J.*, vol. 240, Jun. 2014, Art. no. 3.

[32] Docker Inc. (2019). *Docker: Enterprise Application Container Platform*. [Online]. Available: https://www.docker.com/

[33] A. Madhavapeddy *et al.*, "Unikernels: Library operating systems for the cloud," in *Proc. 18th Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS)*, New York, NY, USA, 2013, pp. 461–472. doi: 10.1145/2451116.2451167.

[34] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *Proc. 1st Int. Conf. Cloud Comput. (CloudCom)*. Berlin, Germany: Springer-Verlag, 2009, pp. 254–265. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10665-1_23

[35] X. Zhang, Z. Huo, J. Ma, and D. Meng, "Exploiting data deduplication to accelerate live virtual machine migration," in *Proc. IEEE Int. Conf. Cluster Comput.*, Sep. 2010, pp. 88–96.

[36] K. M. Khan and Q. Malluhi, "Establishing trust in cloud computing," *IT Prof.*, vol. 12, no. 5, pp. 20–27, 2010.

[37] A. Haeberlen, P. Aditya, R. Rodrigues, and P. Druschel, "Accountable virtual machines," in *Proc. 9th USENIX Conf. Operating Syst. Design Implement. (OSDI)*, Berkeley, CA, USA, 2010, pp. 119–134. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924952

[38] *Software Guard Extensions Programming Reference, Revision 2*, Intel, Santa Clara, CA, USA, 2014.

[39] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA, 2009, pp. 169–178. doi: 10.1145/1536414.1536440.

[40] OpenStack. (2019). *Build the Future of Open Infrastructure*. [Online]. Available: https://www.openstack.org/

[41] (2019). *Openstack and Edge Computing: Uses Cases and Community Collaboration*. [Online]. Available: https://www.openstack.org/edge-computing/

[42] Kubernetes. (2019). *Production-Grade Container Orchestration—Kubernetes*. [Online]. Available: https://kubernetes.io/

[43] Y. Xiong, Y. Sun, L. Xing, and Y. Huang, "Extend cloud to edge with kubeedge," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 373–377.

[44] (2019). *KubeEdge*. [Online]. Available: https://kubeedge.io/

[45] CoreOS. (2019). *Using ETCD*. [Online]. Available: https://coreos.com/etcd/

[46] (2019). *OpenEdge*. [Online]. Available: https://openedge.tech/

[47] (2019). *MQTT*. [Online]. Available: http://mqtt.org/

[48] P. Liu, D. Willis, and S. Banerjee, "ParaDrop: Enabling lightweight multi-tenancy at the network's extreme edge," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2016, pp. 1–13.

[49] K. Bhardwaj, M.-W. Shih, P. Agarwal, A. Gavrilovska, T. Kim, and K. Schwan, "Fast, scalable and secure onloading of edge functions using airbox," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2016, pp. 14–27.

[50] J. Howell, B. Parno, and J. R. Douceur, "Embassies: Radically refactoring the Web," in *Proc. 10th USENIX Conf. Netw. Syst. Design Implement. (NSDI)*, Berkeley, CA, USA, 2013, pp. 529–546. [Online]. Available: http://dl.acm.org/citation.cfm?id=2482626.2482676

[51] S. Nastic, H. Truong, and S. Dustdar, "A middleware infrastructure for utility-based provisioning of IoT cloud systems," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2016, pp. 28–40.

[52] Y. Jararweh, A. Doulat, A. Darabseh, M. Alsmirat, M. Al-Ayyoub, and E. Benkhelifa, "SDMEC: Software defined system for mobile edge computing," in *Proc. IEEE Int. Conf. Cloud Eng. Workshop (IC2EW)*, Apr. 2016, pp. 88–93.

[53] B. Amento, B. Balasubramanian, R. J. Hall, K. Joshi, G. Jung, and K. H. Purdy, "FocusStack: Orchestrating edge clouds using location-based focus of attention," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2016, pp. 179–191.

[54] R. J. Hall, J. Auzins, J. Chapin, and B. Fell, "Scaling up a geographic addressing system," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, Nov. 2013, pp. 143–149.

[55] Y. Xiong, D. Zhuo, S. Moon, M. Xie, I. Ackerman, and Q. Hoole, "Amino—A distributed runtime for applications running dynamically across device, edge and cloud," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 361–366.

[56] I. Zhang *et al.*, "Customizable and extensible deployment for mobile/cloud applications," in *Proc. 11th USENIX Conf. Operating Syst. Design Implement. (OSDI)*, Berkeley, CA, USA, 2014, pp. 97–112. [Online]. Available: http://dl.acm.org/citation.cfm?id=2685048.2685057

[57] Oracle. (2019). *GraaLVM*. [Online]. Available: https://www.graalvm.org/

[58] I. Farris, T. Taleb, A. Iera, and H. Flinck, "Lightweight service replication for ultra-short latency applications in mobile edge networks," in *Proc. IEEE Int. Conf. Commun.*, May 2017, pp. 1–6.

[59] M. Jang, H. Lee, K. Schwan, and K. Bhardwaj, "SOUL: An edge-cloud system for mobile applications in a sensor-rich world," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2016, pp. 155–167.

[60] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "LAVEA: Latency-aware video analytics on edge computing platform," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Jun. 2017, p. 15.

[61] (2019). *OpenALPR—Automatic License Plate Recognition*. [Online]. Available: https://www.openalpr.com/

[62] S. Echeverría, J. Root, B. Bradshaw, and G. Lewis, "On-demand VM provisioning for cloudlet-based cyber-foraging in resource-constrained environments," in *Proc. 6th Int. Conf. Mobile Comput., Appl. Services (MobiCASE)*, Nov. 2014, pp. 116–124.

[63] K. Ha and M. Satyanarayanan, "Openstack++ for cloudlet deployment," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-15-123, 2015.

[64] S. Y. Jang, Y. Lee, B. Shin, and D. Lee, "Application-aware IoT camera virtualization for video analytics edge computing," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 132–144.

[65] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[66] S. Simanta, G. A. Lewis, E. Morris, K. Ha, and M. Satyanarayanan, "A reference architecture for mobile code offload in hostile environments," in *Proc. Joint Work. IEEE/IFIP Conf. Softw. Archit. Eur. Conf. Softw. Archit.*, Aug. 2012, pp. 282–286.

[67] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan, "Just-in-time provisioning for cyber foraging," in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Services*, 2013, pp. 153–166.

[68] D. Messinger and G. Lewis, "Application virtualization as a strategy for cyber foraging in resource-constrained environments," Softw. Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Note CMU/SEI-2013-TN-007, 2013.

[69] G. A. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root, "Cloudlet-based cyber-foraging for mobile systems in resource-constrained edge environments," in *Proc. Companion 36th Int. Conf. Softw. Eng.*, 2014, pp. 412–415.

[70] A. Madhavapeddy *et al.*, "Jitsu: Just-in-time summoning of unikernels," in *Proc. NSDI*, 2015, pp. 559–573.

[71] R. Morabito, "Virtualization on Internet of Things edge devices with container technologies: A performance evaluation," *IEEE Access*, vol. 5, pp. 8835–8850, 2017.

[72] B. I. Ismail *et al.*, "Evaluation of docker as edge computing platform," in *Proc. IEEE Conf. Open Syst.*, Aug. 2015, pp. 130–135.

[73] K. Ha *et al.*, "Adaptive VM handoff across cloudlets," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-15-113, 2015.

[74] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 140–147, Feb. 2017.

[75] L. Chaufournier, P. Sharma, F. Le, E. Nahum, P. Shenoy, and D. Towsley, "Fast transparent virtual machine migration in distributed edge clouds," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, p. 10.

[76] F. Teka, C.-H. Lung, and S. Ajila, "Seamless live virtual machine migration with cloudlets and multipath TCP," in *Proc. Int. Comput. Softw. Appl. Conf.*, vol. 2, Jul. 2015, pp. 607–616.

[77] L. F. Bittencourt, M. M. Lopes, I. Petri, and O. F. Rana, "Towards virtual machine migration in fog computing," in *Proc. 10th Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput. (3PGCIC)*, Krakow, Poland, Nov. 2015, pp. 1–8.

[78] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 140–147, Feb. 2018.

[79] Y. Qiu, C. H. Lung, S. Ajila, and P. Srivastava, "LXC container migration in cloudlets under multipath TCP," in *Proc. Int. Comput. Softw. Appl. Conf.*, vol. 2, Jul. 2017, pp. 31–36.

[80] L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via docker container migration," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, p. 11.

[81] D. S. Milojicic, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou, "Process migration," *ACM Comput. Surv.*, vol. 32, no. 3, pp. 241–299, Sep. 2000. doi: 10.1145/367701.367728.

[82] *Checkpoint/Restore in Userspace (CRIU)*. Accessed: Feb. 20, 2019. [Online]. Available: https://www.criu.org/Main_Page

[83] J. Hwang, K. K. Ramakrishnan, and T. Wood,

"NetVM: High performance and flexible networking using virtualization on commodity platforms," *IEEE Trans. Netw. Service Manage.*, vol. 12, no. 1, pp. 34–47, Mar. 2015.

[84] A. Ceselli, P. Premoli, and S. Secci, "Cloudlet network design optimization," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, May 2015, pp. 1–9.

[85] Q. Fan and N. Ansari, "Cost aware cloudlet placement for big data processing at the edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[86] S. Mondal, G. Das, and E. Wong, "CCOMPASSION: A hybrid cloudlet placement framework over passive optical access networks," in *Proc. IEEE INFOCOM IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 216–224.

[87] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct. 2017.

[88] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Capacitated cloudlet placements in wireless metropolitan area networks," in *Proc. IEEE 40th Conf. Local Comput. Netw. (LCN)*, Oct. 2015, pp. 570–578.

[89] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 2866–2880, Oct. 2016.

[90] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[91] S. Wang, Z. Liu, Z. Zheng, Q. Sun, and F. Yang, "Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers," in *Proc. Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2013, pp. 102–109.

[92] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 66–73.

[93] Y. Zhang, D. Niyato, and P. Wang, "An auction mechanism for resource allocation in mobile cloud computing systems," in *Wireless Algorithms, Systems, and Applications*, K. Ren, X. Liu, W. Liang, M. Xu, X. Jia, and K. Xing, Eds. Berlin, Germany: Springer, 2013, pp. 76–87.

[94] U. Lampe, M. Siebenhaar, A. Papageorgiou, D. Schuller, and R. Steinmetz, "Maximizing cloud provider profit from equilibrium price auctions," in *Proc. IEEE 5th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2012, pp. 83–90.

[95] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang, "How to bid the cloud," in *Proc. ACM Conf. Special Interest Group Data Commun. (SIGCOMM)*, New York, NY, USA, 2015, pp. 71–84. doi: 10.1145/2785956.2787473.

[96] D. T. Hoang, D. Niyato, and P. Wang, "Optimal admission control policy for mobile cloud computing hotspot with cloudlet," in *Proc. IEEE WCNC*, Apr. 2012, pp. 3145–3149.

[97] V. Cardellini *et al.*, "A game-theoretic approach to computation offloading in mobile cloud computing," *Math. Program.*, vol. 157, no. 2, pp. 421–449, Jun. 2016.

[98] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proc. 3rd ACM Workshop Mobile Cloud Comput. Services (MCS)*, New York, NY, USA, 2012, pp. 29–36. doi: 10.1145/2307849.2307858.

[99] A. Carrega, S. Singh, R. Bruschi, and R. Bolla, "Traffic merging for energy-efficient datacenter networks," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst. (SPECTS)*, Jul. 2012, pp. 1–5.

[100] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[101] E. Gelenbe, R. Lent, and M. Douratsos, "Choosing a local or remote cloud," in *Proc. 2nd Symp. Netw. Cloud Comput. Appl. (NCCA)*, London, U.K., Dec. 2012, pp. 25–30.

[102] Y. Wang and Y. Xia, "Energy optimal VM placement in the cloud," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2016, pp. 84–91.

[103] S. Kaur and S. Bawa, "A review on energy aware VM placement and consolidation techniques," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, vol. 3, Aug. 2016, pp. 1–7.

[104] A. Verma, P. Ahuja, and A. Neogi, "pmapper: Power and migration cost aware application placement in virtualized systems," in *Middleware 2008*, V. Issarny and R. Schantz, Eds. Berlin, Germany: Springer, 2008, pp. 243–264.

[105] A. Kaur and M. Kalra, "Energy optimized VM placement in cloud environment," in *Proc. 6th Int. Conf. Cloud Syst. Big Data Eng. (Confluence)*, 2016, pp. 141–145.

[106] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generat. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X11000689

[107] G. Chen *et al.*, "Energy-aware server provisioning and load dispatching for connection-intensive Internet services," in *Proc. 5th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Berkeley, CA, USA, 2008, pp. 337–350. [Online]. Available: http://dl.acm.org/citation.cfm?id=1387589.1387613

[108] Y. Ren, F. Zeng, W. Li, and L. Meng, "A low-cost edge server placement strategy in wireless metropolitan area networks," in *Proc. 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2018, pp. 1–6.

[109] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "CloudScale: Elastic resource scaling for multi-tenant cloud systems," in *Proc. 2nd ACM Symp. Cloud Comput. (SOCC)*, New York, NY, USA, 2011, pp. 5:1–5:14. doi: 10.1145/2038916.2038921.

[110] S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar, and R. Buyya, "SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter," *J. Netw. Comput. Appl.*, vol. 45, pp. 108–120, Oct. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804514001787

[111] K. Katsalis, T. G. Papaioannou, N. Nikaein, and L. Tassiulas, "SLA-driven VM scheduling in mobile edge computing," in *Proc. IEEE 9th Int. Conf. Cloud (CLOUD)*, Jun. 2016, pp. 750–757.

[112] A. Kiani and N. Ansari, "Toward hierarchical mobile edge computing: An auction-based profit maximization approach," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2082–2091, Dec. 2017.

[113] Q. Xia, W. Liang, and W. Xu, "Throughput maximization for online request admissions in mobile cloudlets," in *Proc. IEEE 38th Conf. Local Comput. Netw. (LCN)*, Oct. 2013, pp. 589–596.

[114] X. Sun and N. Ansari, "PRIMAL: Profit maximization avatar placement for mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6. doi: 10.1109/ICC.2016.7511131.

[115] X. Sun, N. Ansari, and Q. Fan, "Green energy aware avatar migration strategy in green cloudlet networks," in *Proc. IEEE 7th Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Nov. 2015, pp. 139–146. doi: 10.1109/CloudCom.2015.23.

[116] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient VM placement with multiple deterministic and stochastic resources in data centers," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2012, pp. 2505–2510.

[117] L. Ao, L. Izhikevich, G. M. Voelker, and G. Porter, "Sprocket: A serverless video processing framework," in *Proc. ACM Symp. Cloud Comput.*, 2018, pp. 263–274.

[118] T. Elgamal, "Costless: Optimizing cost of serverless computing through function fusion and placement," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 300–312.

[119] L. Feng, P. Kudva, D. Da Silva, and J. Hu, "Exploring serverless computing for neural network training," in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2018, pp. 334–341.

[120] A. Aske and X. Zhao, "Supporting multi-provider serverless computing on the edge," in *Proc. 47th Int. Conf. Parallel Process. Companion*, 2018, p. 20.

[121] Amazon. (2019). *AWS—Invoke*. [Online]. Available: https://serverless.com/framework/docs/providers/aws/cli-reference/invoke/

[122] I. E. Akkus *et al.*, "SAND: Towards high-performance serverless computing," in *Proc.USENIX Annu. Tech. Conf. (USENIX ATC)*, 2018, pp. 923–935.

[123] P. Hao, Y. Bai, X. Zhang, and Y. Zhang, "Edgecourier: An edge-hosted personal service for low-bandwidth document synchronization in mobile cloud storage services," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, p. 7.

[124] S. Echeverría, D. Klinedinst, K. Williams, and G. A. Lewis, "Establishing trusted identities in disconnected edge environments," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2016, pp. 51–63.

[125] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2001, pp. 213–229.

[126] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.* Berlin, Germany: Springer, 2011, pp. 53–70.

[127] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, 2010, pp. 735–737.

[128] F. Lin, Y. Zhou, X. An, I. You, and K. Choo, "Fair resource allocation in an intrusion-detection system for edge computing: Ensuring the security of Internet of Things devices," *IEEE Consum. Electron. Mag.*, vol. 7, no. 6, pp. 45–50, Nov. 2018.

[129] J. Arshad, M. M. Abdellatif, M. M. Khan, and M. A. Azad, "A novel framework for collaborative intrusion detection for m2m networks," in *Proc. 9th Int. Conf. Inf. Commun. Syst. (ICICS)*, 2018, pp. 12–17.

[130] C.-C. Lo, C.-C. Huang, and J. Ku, "A cooperative intrusion detection system framework for cloud computing networks," in *Proc. 39th Int. Conf. Parallel Process. Workshops (ICPPW)*, 2010, pp. 280–284.

[131] S. Roschke, F. Cheng, and C. Meinel, "An extensible and virtualization-compatible ids management architecture," in *Proc. 5th Int. Conf. Inf. Assurance Secur. (IAS)*, vol. 2, 2009, pp. 130–134.

[132] C. Mazzariello, R. Bifulco, and R. Canonico, "Integrating a network ids into an open source cloud computing environment," in *Proc. 9th Int. Conf. Inf. Assurance Secur. (IAS)*, 2010, pp. 265–270.

[133] M. Hussain and B. M. Almourad, "Trust in mobile cloud computing with lte-based deployment," in *Proc. IEEE 11th Int. Conf. Ubiquitous Intell. Comput., IEEE 11th Int. Conf. Autonomic Trusted Comput., IEEE 14th Int. Conf. Scalable Comput. Commun. Associated Workshops (UTC-ATC-ScalCom)*, Dec. 2014, pp. 643–648.

[134] S. Chen, G. Wang, and W. Jia, "A trust model using implicit call behavioral graph for mobile cloud computing," in *Cyberspace Safety Security*. Cham, Switzerland: Springer, 2013, pp. 387–402.

[135] J. Kantert, S. Edenhofer, S. Tomforde, and C. Müller-Schloer, "Representation of trust and reputation in self-managed computing systems," in *Proc. Comput. Inf. Technol., Ubiquitous Comput. Commun., Dependable, Autonomic Secure Comput., IEEE Int. Conf. Pervasive Intell. Comput. (CIT/IUCC/DASC/PICOM)*, Oct. 2015, pp. 1827–1834.

[136] N. Bennani, K. Boukadi, and C. Ghedira-Guegan, "A trust management solution in the context of hybrid clouds," in *Proc. IEEE 23rd Int. WETICE Conf. (WETICE)*, Jun. 2014, pp. 339–344.

[137] D. Satria, D. Park, and M. Jo, "Recovery for overloaded mobile edge computing," *Future Gener. Comput. Syst.*, vol. 70, pp. 138–147, May 2017.

[138] A. Aral and I. Brandic, "Dependency mining for service resilience at the edge," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 228–242.