INVITED PAPER

Computation Offloading Toward Edge Computing

This article surveys recent research efforts made on exploring computation offloading toward edge computing.

By Li Lin, *Member IEEE*, Xiaofei Liao, *Member IEEE*, Hai Jin, *Fellow IEEE*, and Peng Li, *Member IEEE*

ABSTRACT | We are living in a world where massive end devices perform computing everywhere and everyday. However, these devices are constrained by the battery and computational resources. With the increasing number of intelligent applications (e.g., augmented reality and face recognition) that require much more computational power, they shift to perform computation offloading to the cloud, known as mobile cloud computing (MCC). Unfortunately, the cloud is usually far away from end devices, leading to a high latency as well as the bad quality of experience (QoE) for latency-sensitive applications. In this context, the emergence of edge computing is no coincidence. Edge computing extends the cloud to the edge of the network, close to end users, bringing ultra-low latency and high bandwidth. Consequently, there is a trend of computation offloading toward edge computing. In this paper, we provide a comprehensive perspective on this trend. First, we give an insight into the architecture refactoring in edge computing. Based on that insight, this paper reviews the state-of-the-art research on computation offloading in terms of application partitioning, task allocation, resource management, and distributed execution, with highlighting features for edge computing. Then, we illustrate some disruptive application scenarios that we envision as critical drivers for the flourish of edge computing, such as real-time video analytics, smart "things" (e.g., smart city and smart home), vehicle applications, and cloud gaming. Finally, we discuss the opportunities and future research directions.

KEYWORDS | Computation offloading; edge computing; Internet of Things (IoT); mobile cloud computing (MCC); mobile edge computing (MEC)

I. INTRODUCTION

In the last two decades, we have witnessed the emergence of a variety of revolutionary mobile devices, e.g., smartphones and wearable devices. These devices bring the prosperity of mobile computing, enabling computing and communication anywhere and anytime [1]. However, with the appearance of new breeds of applications, such as AR, gaming, and especially artificial intelligence (AI)-based applications, which have high computing resource demands, mobile devices face challenges in terms of computational power, storage, and battery life.

Meanwhile, on the flip side, cloud computing, a centralized computing model, has shown its great power of infinite computing capability and on-demand resource provisioning [2], [3]. Therefore, to augment the power of mobile devices, the concept of MCC is introduced [4]. MCC is the integration of cloud computing and mobile computing, in which mobile devices perform computation offloading to leverage the power of the cloud for speeding up application execution and saving energy consumption.

In computation offloading, a mobile device migrates part of its computing to the cloud. This process involves

Manuscript received January 31, 2019; revised May 14, 2019; accepted May 31, 2019. Date of publication July 9, 2019; date of current version August 5, 2019. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1003500; in part by the National Natural Science Foundation of China under Grant 61832006, Grant 61825202, and Grant 61502103; and in part by the Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Young Scientific Research under Grant 19K20258.
(Corresponding author: Xiaofei Liao.)

L. Lin is with the Service Computing Technology and System Laboratory, Cluster and Grid Computing Laboratory, National Engineering Research Center for Big Data Technology and System, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China.

X. Liao and **H. Jin** are with the Service Computing Technology and System Laboratory, Cluster and Grid Computing Laboratory, National Engineering Research Center for Big Data Technology and System, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: xfliao@hust.edu.cn).

P. Li is with the School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu 965-8580, Japan.

Digital Object Identifier 10.1109/JPROC.2019.2922285

application partitioning, offloading decision, and distributed task execution. MCC has been recognized as a promising way for augmenting mobile devices, and many research efforts have been made on it [5]–[8]. However, the latency is the Achilles heel of cloud computing, as the cloud is usually far away from mobile devices. The latency of public cloud providers is usually over 100 ms [9], which would be disastrous for latency-sensitive applications, such as autonomous driving and real-time video analytics [10].

In this context, the emergence of edge computing is no coincidence. Edge computing, which performs computing at the edge of the network, has created a great buzz in both the academia and the industry [11], [12], and it is now at the bleed edge of computing paradigms [13]. A definition from Futurum [14] states edge computing as: "unlike cloud computing, which depends on DCs and communication bandwidth to process and analyze data, edge computing keeps processing and analysis near the edge of a network, where the data were initially collected."

Coinciding with some similar concepts such as "fog computing" [15]–[17] or "mist computing" [18], edge computing pushes the computing from the centralized cloud to decentralized edges, close to the data source (e.g., mobile devices). By extensive deployment of massive edge nodes, such as cloudlets [19], base stations, and micro DCs, edge computing builds hierarchical computing layers: mobile devices, edge nodes, and the cloud, forming the edge-centric computing [20], [21]. The remarkable characteristic of edge computing is that it provides ultra-low latency and reduces the influx of data to the backbone. In addition, it enables location awareness, on-premises deployment, network context understanding [22], and so on. Consequently, leveraging edge computing with computation offloading is a promising way to address the latency issue in MCC.

However, computation offloading in edge computing bears a superficial resemblance to MCC. The new architecture of edge computing drives a rethinking of computation offloading. First, task placement is not only two options, i.e., either at local or in the cloud, but also possible on any edge node. Furthermore, edge nodes connect in a loosely coupled way on heterogeneous wireless networks, making the offloading decision and resource management more complicated. Second, given that task execution is distributed among mobile devices, multiple edge nodes, and the cloud, it is challenging to make flexible and robust code partitioning. Finally, unlike DCs that are well organized and connected, there is no standardized deployment for edge nodes. Therefore, it is nontrivial to build a consistent execution environment upon these heterogeneous platforms and further support user mobility, which requires the migration of offloading services from one edge node to another.

In this paper, we survey recent research efforts made on exploring computation offloading toward edge computing.

Table 1 List of Important Acronyms

Acronym	Definition
API	Application Programming Interface
AR	Augmented Reality
CDN	Content Delivery Network
DC	Data Center
DNN	Deep Neural Networks
ETSI	European Telecommunications Standards Institute
FaaS	Function as a Service
ICN	Information-centric Networking
IoT	Internet of Things
LAN	Local Area Network
MCC	Mobile Cloud Computing
mDC	micro Data Center
MEC	Mobile Edge Computing
NDN	Named Data Networking
NFaaS	Named Function as a Service
NFV	Network Functions Virtualization
OaaS	Offloading as a Service
OS	Operating System
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RPC	Remote Procedure Call
RTT	Round-trip Time
VM	Virtual Machine
VR	Virtual Reality
WAN	Wide Area Network
5G	5th Generation Wireless

The most related to this topic is MEC [23]. However, the origin of MEC is to provide the capabilities of cloud computing within the RAN at the edge of the network [22], and it is considered as the key enabler toward 5G [24]. Nevertheless, we focus on how to augment mobile devices by computation offloading, which is regarded as a key computing paradigm in edge computing [11]. Unlike existing surveys [25], [26], which put forward to issues from the communication perspective, in this paper, we highlight the challenges of computation offloading with respect to task partitioning, allocation, and execution over the new architecture of edge computing and further investigate disruptive application scenarios, such as real-time video analytics, autonomous driving, smart home, and cloud gaming. Specifically, we list important acronyms in Table 1.

The rest of this paper is organized as follows. Section II presents the related concepts and some background. Section III provides an overview of computation offloading in the edge computing paradigm, followed by Section IV summarizing the challenges. Section V showcases some disruptive application scenarios. Section VI offers the views of opportunities and future directions. Finally, Section VII concludes this paper.

II. BACKGROUND AND MOTIVATION

In this section, we first introduce some related concepts and then offer a brief review of the literature about the evolution of computation offloading; finally, key motivations that drive the need of edge computing are presented.

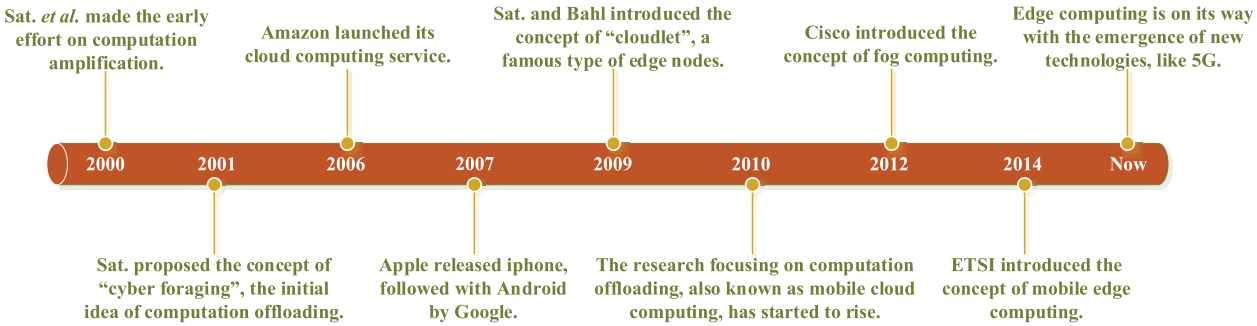

Fig. 1. *Brief timeline of computation offloading. In the late 1990s, computation offloading was a natural idea to leverage powerful remote infrastructures to augment the computing of thin mobile devices. Then, with the development of cloud computing and mobile computing in the mid-2000s, it became the paradigm of MCC. Now, with the emergence of edge computing, computation offloading enters the era of the integration of mobile, edge, and cloud computing.*

A. Related Concepts

As previously discussed, edge computing is a paradigm that the computing happens close to the data source, and it builds a hierarchy of decentralized data processing. Its objective is to eliminate the long latency from end devices to the centralized cloud and alleviate the overhead of data transmission.

To crystallize the idea of edge computing, the ETSI launched an introductory technical white paper of MEC in 2014. More specifically, MEC defines a collection of architectures, technologies, platform APIs and interfaces, and business models associated with RAN for edge computing [22], [24]. Furthermore, ETSI creates a blueprint of an MEC server platform, which provides the functions of computing, networking, and storage based on VMs and NFV. The goal of MEC is to provide ultra-low latency and high-bandwidth computing environments for latency-sensitive applications.

Computation offloading is a key computing paradigm used in edge computing, and it is intrinsically a distributed computing over heterogeneous networks. It is first introduced in MCC and then widely used in edge computing now. We provide more details about the evolution of computation offloading in Section II-B.

Besides, edge computing is not the only game in town. Fog computing, created by Cisco in 2012 [15], is another key player, which extends the cloud closer to the network edge. Fog computing highlights the characteristics, such as distributed processing, online analytics, interplay with the cloud, and network management. A point of view from industry [27] describes the difference between edge computing and fog computing as: "*while edge computing refers more specifically to the computational processes being done at or near the edge of a network, fog computing refers to the network connections between the edge devices and the cloud.*" In other words, edge computing builds the architecture of computing at the edge, while fog computing uses edge computing and further defines the network connection over edge devices, edge servers, and the cloud. However, fog computing and edge computing have the same design philosophy, and they are overlapping in their functions. As a result, the principles discussed in this paper for edge computing can also be applied to fog computing.

Last but not least, 5G is another "term" frequently mentioned in edge computing, and they are symbiotic technologies. Edge computing is a key technology toward 5G [24], and 5G accelerates the development of edge computing; 5G defines three usage scenarios: enhanced mobile broadband (eMBB), ultra-reliable and low-latency communications (URLLCs), and massive machine-type communications (mMTCs) [28]. These scenarios coincide with those from edge computing, such as video analytics, smart cities, and autonomous vehicles.

B. Evolution of Computation Offloading

The principle of computation offloading is to leverage powerful infrastructures (e.g., remote servers) to augment the computing capability of less powerful devices (e.g., mobile devices). This paradigm has evolved over the last 20 years with the emergence of various computing technologies. Fig. 1 shows the evolution of computation offloading organized by the timeline.

In the late 1990s, Noble *et al.* [29] and Flinn and Satyanarayanan [30] studied on how to improve the performance of mobile applications and save the energy of mobile clients by collaborative computing with remote servers. They built a prototype, *Odyssey*, as a proof of concept to support the adaptive execution of mobile applications. This application-aware adaptation can change application behaviors dynamically in light of resources, such as CPU cycles, bandwidth, and battery power.

In 2001, Satyanarayanan characterized pervasive computing as the integration of distributed computing and mobile computing, and then he coined the term "cyber foraging." The idea of cyber foraging is to "augment the computing capability of a thin mobile device dynamically by leveraging the power of wired hardware infrastructure." It is the origin of computation offloading.

In the mid-2000s, two computing paradigms—cloud computing and mobile computing—had started to show their power in the industry, business, and human life. Amazon created its elastic compute cloud (EC2) in 2006 [32], and it kicked off the commercial use of cloud computing [2]. One year later, as the release of mobile OSs, iOS [33] and Android [34], smartphones rich in applications and sensors come in people's life, and it leads to the prosperity of mobile computing. Consequently, the combination of mobile computing and cloud computing becomes natural, also known as MCC [4], [35]. MCC inspires the academic research, and it is the de facto paradigm of computation offloading at that time. Since 2010, many works have emerged, focusing on computation offloading, such as MAUI [5], Cuckoo [36], CloneCloud [6], ThinkAir [7], and COMET [8]. These works show how to achieve performance improvement and energy saving of mobile devices by offloading computation to the cloud according to the network connectivity and resource provisioning [37].

However, with the proliferation of real-time applications, e.g., AR, video analytics, and gaming, computation offloading to the cloud solely cannot satisfy the stringent latency requirement. To solve this problem, researchers have started to make efforts to place the computing close to mobile devices to reduce network latency. In 2009, Satyanarayanan *et al.* [19] introduced the concept of "cloudlet"—a small DC nearby mobile devices, which is a proof of concept for edge computing. Based on cloudlets, they advocated a three-tier architecture consisting of mobile devices, cloudlets, and the cloud, aiming to "bring the cloud closer" [38]. Then, Cisco announced the role of fog computing in 2012 [15], and ETSI published the technical white paper for MEC in 2014 [22]. These two momentous events unveil the research flourish of edge computing in the academic community.

Edge computing is now still in its infancy but has gained great attention. The emergence of new technologies, especially 5G, drives continuously prosperous development of edge computing. As mentioned earlier, 5G is characterized with fiber-like high speed and ultra-low latency, and these characteristics provide significant advantages for computation offloading.

C. Need for Edge Computing

What is the driving force of computation offloading toward edge computing? The latency matters [39], especially in the cases of latency-sensitive applications, such as autonomous driving, real-time video analytics, and AR/VR. For economic consideration, DCs of public cloud providers are placed in specific locations, usually far from end users, leading to high WAN latency. In 2010, an investigation by Li *et al.* [9] showed that the average network RTT of Amazon's cloud instances is 74 ms. After six years, in 2016, the average latency of five leading public cloud providers in four regions of the United States is 78 ms, as shown in Fig. 2, which means that the latency of the public cloud does not significantly improve as six years pass.

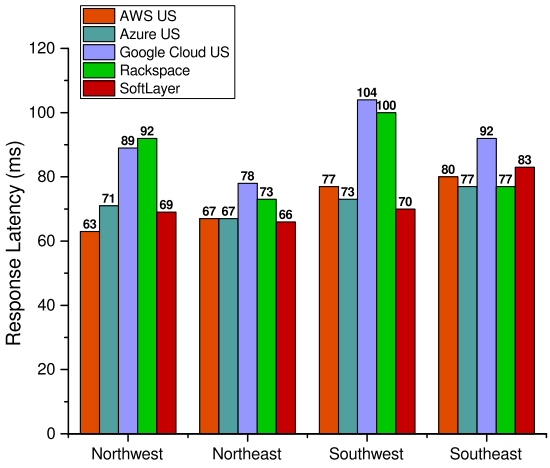

Fig. 2. ***Average response latency of five top public cloud providers in the United States for serving in four regions of the country. Source from Cedexis/Network World [40].***

Table 2 Data Rate of Major Radio Access Technologies for Cellular Networks [49]

Technology	2.5G(Edge)	3G+(HSPA)	4G(LTE)	5G
Data rate	236 Kbps	22-56 Mbps	100 Mbps	1 Gbps

On the contrary, the development of radio access technologies is on the road. Table 2 lists the change of data rate for cellular networks, and we can see the fast growth of data capacity. For example, the latest radio access technology, 5G, which will soon come to the mainstream of consumers, can have the data capacity of more than 1 Gb/s and support 1-ms latency [41]. As a result, processing data within the RAN is considered to be a promising way to solve the latency issue for real-time applications in the visible future.

Furthermore, computing at the edge can reduce the network traffic of the core network [11], driving data analytics toward geo-distributed processing, known as edge analytics [42]–[44]. The intelligent edge analytics takes advantage of low latency and prevents the influx of information to DCs, especially in the field of IoT [45], [46]. Moreover, distributed edge infrastructures can provide robust node connectivity.

Consider two potential applications of edge computing: autonomous vehicles and AI-based video surveillance. Autonomous and connected vehicles equipped with sensors, from radars to cameras to lidars, are changing the way people travel. When a self-driving vehicle is on the road, it is calculating the speed, sensing the distance to surrounding vehicles and pedestrians, communicating with other vehicles and roadside units. Then, it issues operations to make the vehicle driving safely with stringent latency requirements. Moreover, estimated by Intel,

an autonomous vehicle will produce 4000-GB data each day [47]. "The coming flood of data" can easily overwhelm the existing network and cloud facilities. As previously discussed, these autonomous vehicles can utilize edge infrastructures (such as onboard computers and enhanced base stations) to process data immediately with the fast network connection (such as 5G).

In the scenario of video surveillance, cameras continuously gather a considerable amount of data, and these data contain valuable information. Shipping all the data to the cloud for analysis is undoubtedly impossible to meet the latency requirement. Instead, AI-integrated real-time video analytics at the edge can help address this challenge. IT companies already have off-the-shelf products, including hardware devices and software solutions, such as Irvine Sensors' ALERT [48]. ALERT is a distributed video analytics platform that provides real-time object detection and analysis by joint back-end training and front-end cognitive processing. For example, in a train station, ALERT recognizes a bag left behind and identifies the bag as a potential threat, and then, it notifies authorities. After detecting the bag that is removed, ALERT deescalates the threat.

Not limited to the above-mentioned applications, edge computing can also bring new value in the fields of manufacturing, healthcare, energy and grid control, and agriculture. The ever growing of data and the continuous rise of latency requirements drive the development of edge computing.

III. OVERVIEW OF COMPUTATION OFFLOADING

In this section, we provide a brief introduction of computation offloading in edge computing, including the architecture, offloading analysis, offloading granularity, and the impact of network connectivity.

A. Architecture

The goal of edge computing is to reduce the latency by placing computing close to the data source. It is achieved by constructing a hierarchical architecture consisting of a wide variety of computing devices. As shown in Fig. 3, it is a typical three-tier architecture, including the cloud, edge nodes, and edge devices [50]. Edge devices represent devices' computing at the end of the network [51], and these devices are also named end devices [15], mobile devices, and user equipments [52]. Specifically, most of the time, we use the terminology "mobile devices" to refer to edge devices in this paper. Smartphones, wearable devices (e.g., google glass and apple watch), cameras, vehicles, and so on are typical types of edge devices. Billions of edge devices that are geographically dispersed compute everywhere and produce zillions of data everyday. On the flip side, the cloud, standing at the core of the network, is a critical centralized node, and it can help to discover

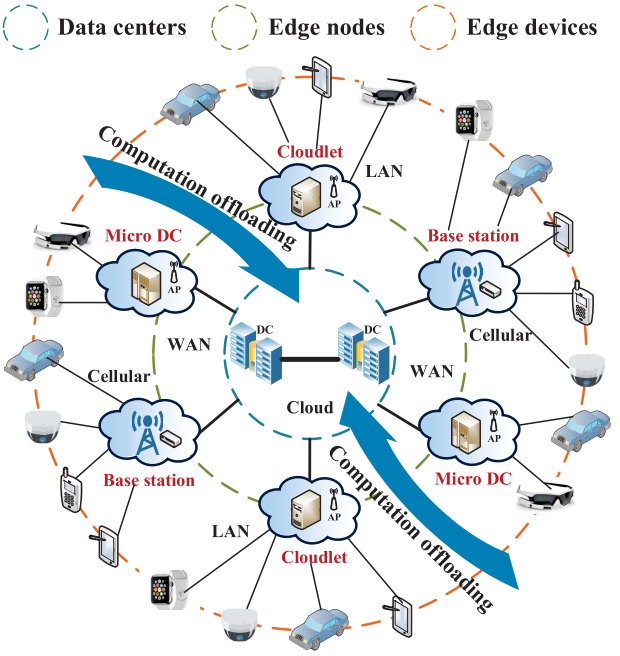

Fig. 3. *Architecture of edge computing with the flow of computation offloading. It is a typical three-tier architecture consisting of edge devices, edge nodes, and the cloud.*

edge nodes [53], conduct resource management [54], and perform global big data analytics [55].

An edge node or an edge infrastructure, at the middle layer, is a new shape of the computing node, which locates at the communication link between the end and the core of the network. Edge nodes have more powerful computational capacity than edge devices and provide lower latency than the cloud. However, edge nodes differ in form factors and the types of hardware. The following is the classification of edge nodes.

1) *Cloudlets:* The cloudlet is first introduced by Satyanarayanan *et al.* [19] to address the latency obstacle of computation offloading to the cloud over WAN. A cloudlet is featured with resource-rich, trusted, and one-hop network latency to nearby mobile devices, resembling a "DC in a box." Cloudlets are deployed at coffee shops, hospitals [56], and airports [57] to provide proximity computing, and they are now well known as typical edge infrastructures. In particular, we usually use "cloudlets" to indicate edge nodes in this paper.
2) *Micro Data Centers:* Unlike the cloud that has mega DCs with tens of thousands of servers [58], mDCs are geo-distributed small DCs, which have only the small or moderate order of servers. By deploying a lot of mDCs around the world at strategic places, it can lower the network latency, save bandwidth consumption, provide reliable connectivity, and reduce the overhead of the cloud [39]. A successful example of using mDCs is Akamai [59],

who provides the service of CDN by deploying mDCs around the world [60].

3) *Base Stations:* The development of communication technologies, such as LTE and 5G [61], has driven the evolution of the base station, which is an ideal type of the edge node. These enhanced base stations, densely deployed in close proximity to mobile users, could provide the computing service in addition to essential communication functions. Base stations have already been used as edge servers in forms of LTE macro base stations (eNodeB) [24] and small cell clouds [62], [63], and these infrastructures leverage the technology of NFV [64] to offer scalable computing services.
4) *Other Types of Edge Nodes:* Theoretically, any computing infrastructures, located at the communication link bridging the cloud and edge devices to smooth the gap of latency and computing capability, can be considered as edge nodes. Accordingly, vehicles equipped with onboard computers [65], IoT gateways associated with the cellular network [66], and even smartphones [67] are the infrastructures of edge nodes.

Edge nodes are deployed by different organizations, such as schools, shops, hospitals, and companies. A cluster of edge nodes, which are well connected and with standard configurations, is a kind of edge cloud [68]. Edge cloud is a concept relative to the centralized cloud (e.g., Amazon AWS and Microsoft Azure), and it means "edge cloud takes compute capacity to where the traffic is."

Given the three-tier architecture in Fig. 3, computation offloading usually happens across the layers from outside in or inside specific layers. Take a real-time object recognition application as an example. Smartphones can directly migrate the execution of object recognition to the cloud, as the paradigm of MCC, or they employ cloudlets to perform the same computation offloading for latency reducing, which is the form of edge computing [69]. Besides, offloading can happen on the same tier. For instance, if a smartphone uses the compute capacity of other smartphones in a peer-to-peer model [70], it is a smartphone-to-smartphone offloading [71], [72]. Furthermore, edge nodes also ship computation to the cloud. As an example, in GigaSight [73], a video analytics framework, it runs offloading requests from mobile devices on cloudlets and then migrates part of the computation to the cloud.

B. Simple Analysis of Offloading

Computation offloading is essentially a distributed computing paradigm, in which mobile devices leverage remote servers to speed up computing and save energy. In computation offloading, mobile devices need to transmit data over the network. As a result, it is a tradeoff between the benefit of remote execution and the cost of data transmission. First, computation offloading happens only if the time of the local execution is longer than its total offloading time, which includes the communication time and the remote execution time. It is formulated as

$$T_{\text{local}} > T_{\text{offloading}} = T_{\text{comm}} + T_{\text{remote}} \tag{1}$$

where T_{comm} indicates the communication time and T_{local} and T_{remote} refer to the execution time at local and remote, respectively. Meanwhile, the energy benefit is

$$E_{\text{local}} > E_{\text{comm}} \tag{2}$$

i.e., the energy consumption of data transmission is smaller than the local execution. Therefore, offloading is beneficial if performing much computation with transmitting a relatively small quantity of data [37].

Then, we perform a further analysis on $T_{\text{offloading}}$ with the cloud and the cloudlet, which refer to $T_{\text{comm\_cloud}} + T_{\text{cloud}}$ and $T_{\text{comm\_cloudlet}} + T_{\text{cloudlet}}$, respectively. Edge devices have the communication time with the cloud $T_{\text{comm\_cloud}}$ larger than $T_{\text{comm\_cloudlet}}$ for cloudlets. However, the cloud has several orders of magnitude computing speed faster than cloudlets, and then, $T_{\text{cloud}} < T_{\text{cloudlet}}$. This simple analysis provides an insight that applications, which are bandwidth-hungry and latency-sensitive but moderately compute-intensive, can achieve much more benefit with offloading to cloudlets than to the cloud. Such applications include video analytics [10], gaming [74], and IoT analytics [75]. On the contrary, applications requiring large-scale computing are still cloud-friendly, for example, Web search.

C. Granularity of Offloading

The granularity of offloading can be classified into three categories: full computation offloading, task/component, and method/thread. Fig. 4 shows these forms of offloading.

1) Full Offloading: As shown in Fig. 4(a), full computation offloading migrates the whole computing part of applications, leaving mobile devices that are only responsible for UI, input/output, and data sensing. This paradigm is known as the model of thin clients. A prominent example is web-app [76], in which thin clients only get user input and browse the results.

2) Task/Component: Offloading at the task/component granularity divides the applications into different tasks and components and then places compute-intensive tasks/components to remote infrastructures, as shown in Fig. 4(b). To achieve this goal, developers analyze the workflow of the application execution to gain a deep insight into the program behavior. Therefore, they can have the right task/component partitions. These partitions are application-dependent; even the same application could have different partitioning strategies [77].

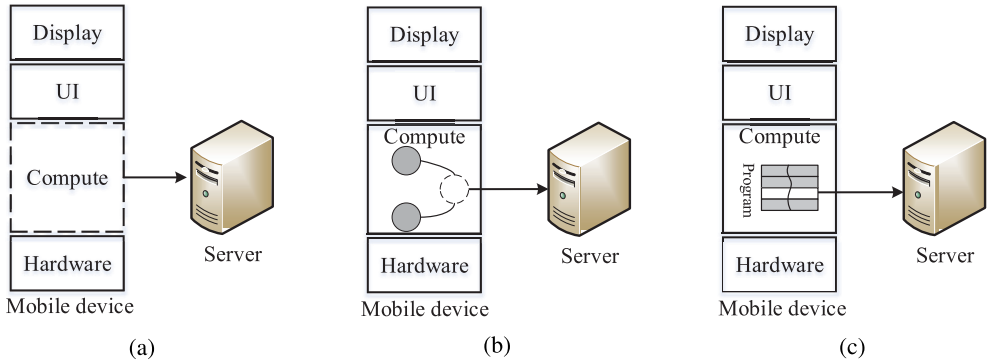

Fig. 4. ***Granularity of computation offloading. Generally, mobile devices can offload computation at (a) full offloading, (b) task/component, and (c) method/thread levels.***

Take the AR application as an example. The execution of an AR application can be divided into four steps: video captured by the camera, the object tracking, the scene rendering, and display. Since the object tracking is the critical and most complicated component [78], offloading this task can have a great benefit. Furthermore, in an extreme case, if we offload both the tracking and rendering components, i.e., the whole computing tasks, then it is full offloading.

3) Method/Thread: Offloading at the method/thread granularity is a fine-grained computing migration, as shown in Fig. 4(c). Generally, an application contains a considerable amount of methods, and then, offloading at the method/thread level needs partitioning mechanisms to assist (elaborated in Section IV-A), whereby the most beneficial methods can be found. Such methods are usually compute-intensive with a small or moderate amount of data transmission. For example, in [79], a function performing optical character recognition (OCR) in an AR application is offloaded.

D. Impact of Network Connectivity

The network connectivity has a significant impact on the performance of offloading for both the latency and the energy consumption. Offloading with Wi-Fi is usually considered to have better performance than with 3G and 4G LTE, as Wi-Fi has higher downlink and uplink in practice. For example, in MAUI [5], the experiments show that offloading methods to a nearby server with Wi-Fi can reduce latency by 83%, but the performance with 3G is even worse than local execution in some cases. The similar results are found in [80], and the authors observed that 4G LTE has up to 40% longer network latency than Wi-Fi.

For energy consumption, Wi-Fi is also more energy efficient than 3G and 4G LTE. Wi-Fi typically employs power save mode on smartphones. Therefore, the energy consumption of maintaining data transmission is small [81]. Moreover, 3G and 4G LTE have high tail energy [82] that incurs additional energy cost after completing data transfer.

IV. CHALLENGES FOR COMPUTATION OFFLOADING IN EDGE COMPUTING

Computation offloading in edge computing faces many challenges: application partitioning, task allocation, and task execution. In this section, we investigate research efforts made on these challenges. In particular, for each challenge, we will highlight unique obstacles in edge computing compared with MCC. Fig. 5 presents a summary of these challenges.

A. Application Partitioning

The first step toward computation offloading is partitioning, which divides the application code into several parts that will be executed on different platforms, i.e., mobile devices, cloudlets, or the cloud. It is well investigated in MCC [83]–[85]. However, it shows some new features in edge computing.

1) Partitioning in MCC: Generally, there are two approaches to application partitioning: automatic program analysis and programmer-specified partitioning. In the former one, static or dynamic program analysis techniques are employed to inspect the data flow of application code and analyze potential offloadable parts. In practice, a graph-based model is the most commonly used in a program analysis to represent the interactions among methods, objects, or tasks/components. In such an interaction graph, vertices indicate the above-mentioned entities and the weights of edges define interaction costs (e.g., data sizes, communication time, and bandwidth). Then, graph partitioning algorithms are applied to cut the graph by achieving some goals, such as saving energy [5], [86], [87], reducing execution time [88], minimizing network traffic [89], or maximizing throughput [90].

One of the earliest offloading systems, Coign [91], has developed this technology in 1999. Based on program behavior profiling, Coign builds a graph model of the application's inter-component interaction. Then, Coign employs a graph-cutting algorithm, named lift-to-front

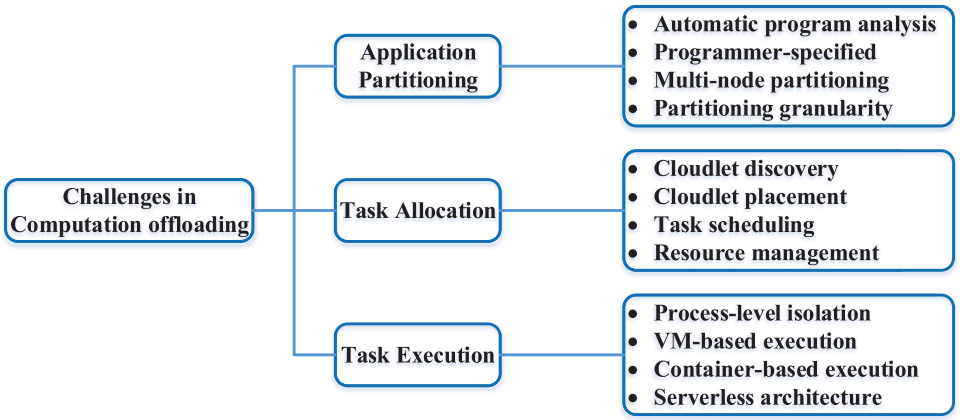

Fig. 5. *Summary of challenges for computation offloading toward edge computing.*

minimum-cut, to select a partition to minimize communication time. Coign is the first system to substantiate the concept of automatic program analysis, and its goal is to ease the programmer effort.

Furthermore, CloneCloud [6] offers the best example to achieve the goal of automatic partitioning in recent years. CloneCloud combines the static program analysis and dynamic runtime profiling to partition applications at the thread granularity. Specifically, the static analysis leverages the program control-flow graph to explore the internal invoking of classes/methods. Then, additional constraints are added to the analysis; for example, methods accessing the hardware must be anchored at mobile devices. Finally, an integer linear programming-based solver is employed to choose the partitions to minimize the computation and migration cost.

The way of automatic program analysis mitigates programming effort, leaving programmers out of the details of code analysis, but it has a sacrifice of flexibility. On the contrary, partitioning by programmers explicitly specifying components or methods to be offloaded offers flexibility. This approach is based on the hypothesis that "for every application, the number of useful ways of splitting the application for remote execution is small" [88], which has been substantiated by existing studies, such as MAUI [5] and CloneCloud [6]. MAUI allows programmers to annotate methods considered to be executed remotely with the support of its customized runtime environment. In other words, its runtime will automatically identify annotated methods and offload them. Echo [92] follows this way but employs a lightweight programming model that erases the burden of runtime customization. Specifically, Echo adopts the technique of aspect-oriented programming (AOP) [93] to insert additional operations into application code. These operations (programmer-agnostic) manipulate the process of method offloading automatically.

Both MAUI and Echo perform offloading at the method level. In addition, a programmer can manually partition an application at the task or component level. To give a clear example, Liu *et al.* [94] redesign the GPS sensing applications and separate the function of raw GPS signal receiving from its signal data postprocessing. Then, they leave the GPS signal that received on mobile devices but ship signal data postprocessing to the cloud for fast and precise processing.

2) Partitioning in Edge Computing: Given that the above-mentioned partitioning techniques are off-the-shelf, what is new in computation offloading based on edge computing? One challenge is partitioning for multinode offloading [95]. Tasks in MCC have specific sites to be placed, either the local (mobile devices) or remote servers (the cloud), determined by runtime decisions. However, in edge computing, tasks can distribute across servers in the whole network link from mobile devices to the cloud, including numerous cloudlets. In this case, what is the appropriate partitioning?

Sinha and Kulkarni designed a partitioning algorithm to perform multinode offloading [96], which offloads tasks of an application to multiple remote servers. This paper is motivated by data-centric applications. For example, to perform photo recognition and image matching across multiple sites, typical applications include Flick and Facebook. They employ an object interaction graph to represent the program's behavior and use a heuristic graph-cutting algorithm to minimize communication costs.

Another challenge is the partitioning granularity. A lot of works in MCC perform computation offloading at the method, thread, and class level. Executing at these levels is not only data dependence but also context dependence, i.e., the task execution requires consistent runtime environments. However, considering heterogeneous infrastructures in edge computing, such a requirement is expensive. Therefore, offloading at the task/component level would provide better efficiency. Developers can design different ways for task/component execution in light of different platforms.

Deng *et al.* [97] provided a successful example to divide applications into workflows (tasks/components). Take the facial expression recognition application as an example. They divide the application into components, including input video, face detection, facial feature detection, face registration, facial feature extraction, action unit recognition, expression recognition, and output class. These components are data dependence; namely, the output of a former one is the input of a next one. Based on the workflow, they implemented an offloading system with robust offloading decisions to place these components. For example, a possible decision is to offload the action unit recognition, which requires much computation but with small-sized data input and output. Partitioning based on workflows has been substantiated by a lot of existing works, and we will see more examples in Section V.

B. Task Allocation

Task allocation refers to the runtime decision of task placement and scheduling associated with the resource management. In light of new features of edge computing, task allocation has a threefold challenge: cloudlet discovery, multiresource management, and decentralized scheduling.

1) Cloudlet Discovery: Cloudlet discovery is unique in edge computing, as cloudlets are distributed and not well-organized [98]. Ha [99] developed two types of discovery mechanisms: LAN discovery and WAN discovery. The LAN discovery uses the protocol of zero-configuration networking [100], such as Avahi [101] and Apple Bonjour [102], to find cloudlets nearby within LANs. The WAN one employs centralized directory servers, e.g., the cloud, to help the cloudlet registration and locating. Factors of network proximity, resource availability, cache states, and authentication are considered when performing the cloudlet discovery.

Furthermore, [99] proposed an application-aware discovery, and it adopts adaptive discovery strategies according to the types of applications. For example, for interactive applications, such as cloud gaming, which is latency-sensitive, the network proximity between mobile devices and cloudlets is the most critical factor in choosing cloudlets. Alternatively, for compute-intensive applications, such as computer vision, the computing speed of a cloudlet is the first consideration.

Similarly, EDOS [103] extends the discovery strategy with service-type aware: the types of network-intensive and compute-intensive. This way of cloudlet discovery combines the service type and available resources, and it uses a cost model to evaluate the resource consumption of the specific service type to find an appropriate cloudlet.

In addition, a related issue about cloudlet discovery is the cloudlet placement, which addresses the problem of placing cloudlets optimally in light of application requirements. Jia *et al.* [104], [105] proposed the heuristic algorithms to enable the optimal cloudlet placement to minimize the average wait time of user requests. Besides, Gedeon *et al.* [106] conducted a thorough analysis of the existing urban infrastructures, including smart lamp posts, commercial routers, and cellular base stations. Then, they proposed a placement strategy to position cloudlets on these existing infrastructures according to the cost and QoS.

2) Task Scheduling With Resource Management: In edge computing, task scheduling is daunting for the complexity of multiuser and multinode scheduling and dynamic resource management. Task allocation is not only two options of mobile devices and the cloud but also possible on any cloudlet over edge clouds, leading to more complicated for runtime decisions. Meanwhile, because of user mobility and the intricate wireless heterogeneous network, it is challenging for task scheduling under the rapid-change environment.

Recalling task scheduling in MCC, the runtime decision usually employs the strategy of the resource supply and demand prediction. Namely, making offloading decisions depends on resources available on the mobile device and resource requirements for task executing [107]. MAUI uses this strategy by combining resource profiling and decision solving. It builds device, program, and network profilers to collect data ranging from hardware to program characteristics to network conditions, which are then as the input to its global optimization solver. The solver employs the integer linear programming to find the optimal policy that minimizes the energy consumption of mobile devices by satisfying latency constraints. The final decision will determine whether methods to be executed remotely or locally. CloneCloud uses the similar way except that its optimization goal is to minimize the execution time.

Resource management in MCC is mainly from the mobile device point of view without taking into account the cloud, i.e., only considering the adaptive resource scheduling on the mobile device. For example, [108] proposed an energy-efficient resource scheduling policy by adjusting the CPU clock, namely, the dynamic voltage and frequency scaling (DVFS). Similarly, Cidon *et al.* [109] presented a scheduler to balance the CPU and network resources. This kind of scheduling strategy is reasonable, as the cloud is supposed to have unlimited resources, and offloading services can be satisfied with on-demand resource provisioning [3]. To crystallize this idea, ThinkAir [7] realizes an on-demand cloud resource allocation through dynamic VM provisioning, and it supports task parallelism based on dynamic scaling of computational power. Unlike other studies, ThinkAir focuses on the scalability and elasticity of resource supply on the cloud side, and it provides knowledge about resource management from the cloud perspective.

However, cloudlets that are geographically distributed have limited resources. Therefore, task scheduling in edge computing is a twofold challenge. First, in single-user scenarios, the scheduling is challenging by trading

off between the effectiveness and efficiency of resource provisioning. Second, if considering multiuser scenarios, multinode collaborated resource allocations and load balancing are two challenges.

To address these challenges, in the single-user scenario, [110] studied the task allocation in MEC by optimizing both the task execution time and the energy consumption. Its basic idea is to employ dynamic CPU frequency adjustment algorithms to perform the task allocation decision. This decision only considers the computational resources, while radio resources are important in MEC. Therefore, the joint optimization of computational and radio resources is necessary, and more and more researchers have paid attention to this optimization in the multiuser scenario [111]–[114]. Sardellitti *et al.* [111] exploited cellular base stations to provide the capability of edge computing and studied the joint optimization of computational and radio resources. They formulate this problem as global optimization by minimizing energy consumption while satisfying latency constraints. Then, an iterative algorithm based on successive convex approximation is proposed to solve this problem. Furthermore, to address the multiuser computation offloading for MEC, [112] proposed a distributed and game theory-based decision-making algorithm, in which the resource management can achieve a Nash equilibrium of multiuser game.

Apart from resource management, load balancing is an acute issue of task scheduling in the multiuser scenario. Unlike the cloud that has the global view of overall resources and thus usually adopts centralized scheduling policies [115], [116], cloudlets are geographically distributed without a single centralized node, leading to difficulty in performing load balancing [117]. Several strategies have been proposed for this issue. Sparrow [118] realizes a distributed and low-latency scheduling framework for large-scale data analytics, and it adopts a random sampling approach for load balancing based on the theory of "the power of two choices" [119]. More specifically, to schedule a task, its scheduler will randomly probe two servers and select the one with the lighter load to place the task. Furthermore, Hopper [120] extends the basic technologies of Sparrow and further provides scalability and predictability in task scheduling. In addition, Rashidi and Sharifian [121] specified the obstacles in cloudlet selection for task offloading; namely, given that the scheduling information and network status are hard to achieve, how to perform load balancing in this situation. As a solution, they proposed an adaptive neuro-fuzzy inference system, which can deal with the information-limited scheduling and improve the user QoS.

C. Task Execution

In computation offloading, task execution spans mobile devices and remote servers (edge nodes or the cloud), and they are heterogeneous platforms. Generally, a mobile device equips with the ARM-based chip, while a server runs with x86 CPUs. Task execution over heterogeneous architectures requires hardware and software abstractions to encapsulate the difference. Typically, a way of running a mobile application on an x86 server is to start a VM equipped with a mobile OS image (e.g., Android x86 [122]). In practice, there is a wide range of technologies to support this execution, from the early process-level isolation to VM-base solutions to the new serverless architecture. In this section, we review these technologies comprehensively.

1) Task Execution in MCC: Most of the earliest computation offloading systems use the process-level isolation technology to support task execution [107], including Chroma [88] and Spectra [123]. For example, Spectra sets up a separate process on the server for handling the code execution via the RPC mechanism. However, the process-level isolation suffers poor transparency, because it requires two separate code versions running at mobile devices and servers, respectively.

To conquer this downside, researchers have proposed VM-based solutions. VMs cleanly encapsulate the difference between hardware and software among heterogeneous infrastructures [124]. Therefore, VMs can provide consistent execution environments over mobile devices and the cloud. Kemp *et al.* [36] presented a framework, *Cuckoo*, which provides a programming model for developers to facilitate the implementation of computation offloading. Cuckoo uses an interface definition language (AIDL) in Android [125] to describe the code to be offloaded. Then, it compiles the code by Java Builder into an application package. Finally, the application can be run with Java VM (JVM) on remote servers with the offloading capacity. In this way, Cuckoo offers a dynamic runtime environment for task execution at the granularity of components, i.e., Android activities/services [126]. Similarly, CloneCloud runs tasks based on Android Dalvik VM [127], and it can automatically offload parts of an application to the clone of a mobile device in the cloud at the thread granularity. COMET adopts the basic approaches of CloneCloud and further employs distributed shared memory (DSM) to support multithread tasks. The running of CloneCloud and COMET needs a full VM migration to synchronize VM states. To reduce this runtime overhead, MAUI implements a fine-grained computation offloading at the method level based on Microsoft .NET Common Language Runtime (CLR) [128].

The above-mentioned VM-based systems build isolation at the application level. Application virtualization provides a platform-independent programming abstraction hiding the details of OS and hardware. This virtualization is programming-language dependent, and different languages rely on different runtime VMs, such as Cuckoo, CloneCloud, and COMET with JVMs (Dalvik VMs) and MAUI with C# VMs (CLR). However, this approach provides good transparency; code can be run on different hardware platforms if the same programming-language

VMs are set up on each platform. Thereby, most of the existing works in MCC adopt this technology.

2) Supports for Edge Nodes: Since the approach of VM-based task execution has been substantiated efficiently in MCC, it is a natural choice for edge nodes, such as cloudlets, to use the same way. However, technical supports for cloudlets face a twofold challenge [129], [130]. First, providing rapid and dynamic VM provisioning for cloudlets that are geo-distributed is the first challenge. Second, it is challenging to support user mobility seamlessly with uninterrupted services on the turbulent WAN environment.

To address these challenges, several solutions have been proposed. First, for quick VM provisioning, a technique named just-in-time (JIT) provisioning [131] has been designed and implemented. Its basic idea is to synthesize VM dynamically using VM overlays. Specifically, in this technique, base VM images contain guest OSs and supporting libraries. However, an executable VM requires particular software for a specific mobile device, and this part is called *VM overlay*, the amount of which is relatively small. Then, the executable VM instance can be quickly created by synthesizing a VM overlay to a preloaded base VM.

Second, to support user mobility, an agile VM handoff for edge computing has been proposed [132]. VM handoff means VM migration across cloudlets. It bears a superficial resemblance to live migration [133] and is challenging on unstable WANs. The principle of the agile VM handoff is to dynamically tune the balance of network transmission and cloudlet processing by leveraging the mechanisms of pipelined execution and dynamic adaptation of operating modes.

Third, to provide an integrated platform for cloudlet deployment, OpenStack++ [134] has been developed, which is an extended version of OpenStack. OpenStack++ implements features, including importing base VM, resuming base VM, creating VM overlay, VM synthesis, and VM handoff.

VM-based virtualization is heavyweight isolation, as each run of a VM is a full copy of an OS. It is not resource-friendly for cloudlets that are considered to be resource-constrained. In recent years, lightweight virtualization solutions, such as containers and unikernels, are coming into the mainstream. These technologies are considered to be well-suited to edge infrastructures [135]. A container provides isolation on top of the host OS sharing the OS kernel and software stacks [136]. It only takes up small space and memory footprint with low I/O overhead. Typical types of containers include LXC [137], OpenVZ [138], and Docker [139]. Instead, a unikernel uses a unique OS kernel containing the minimum OS functions to support task execution. Researchers have paid attention to these techniques for edge platforms [140]. Wu *et al.* [141] implemented a container-based platform for computation offloading and optimized its resource sharing and mobile code cache. They further built a similar solution based on unikernel [142]. Ha *et al.* [132] presented a fast service handoff via container migration, which is a container version of the previous work.

The above-mentioned works have demonstrated that the performance of the container and unikernel outperforms VMs' in terms of disk, memory footprint, and service latency. However, VMs have advantages on aspects of transparency, isolation, safety, and deployability [132]. Therefore, recently, a new approach combining containers and unikernels with VMs is an on-going solution, for example, running containers upon VMs. Companies, such as VMware, have started to work on it [143].

Both VMs and the likes of containers are server-oriented. In other words, these technologies view applications as the collections of servers with different virtualization granularities. However, servers are notorious for slow start-up and the complicated configuration and management. These characteristics limit the usage of server-based virtualization technologies for computation offloading, because the execution of computation offloading is typically a fast process of stateless functions, emphasizing the low latency and scalability. Fortunately, in recent years, serverless computing, sometimes labeled as microservices or FaaS, has emerged as a compelling architecture. Serverless computing instead drives developers to think of applications as a set of functions. Specifically, this new computing model frees developers from the details of server deployment, resource allocation, and software configuration (from OSs to runtimes to libraries), making them focus on the application business logic. Serverless computing is famous for its good scalability and easy deployment. It is very suitable for computation offloading in edge computing, especially in the area of IoT.

A prominent example of the serverless computing service or microservice is Amazon Lambda [144], which is the first and the most successful serverless platform. Other platforms include Google Cloud Functions [145], Microsoft Azure Functions [146], IBM Cloud Functions [147], and the open-source OpenLambda [148]. The origin of serverless computing is to design a scalable service model in the delivery of cloud computing. It is easy to extend this model to edge infrastructures. For example, Baresi *et al.* [149] have built a serverless edge computing architecture for computation offloading. The experimental results show that the serverless edge infrastructure can achieve a significant advantage compared to the cloud. From the cost and scalability perspective, serverless architecture is fit for edge computing. However, there is still room for improvement in the aspects of flexibility, elasticity, and security [150].

V. APPLICATION SCENARIOS

The new breeds of latency-sensitive applications give impetus to the evolution of computing paradigms, namely, from cloud computing to hybrid edge-cloud computing. In this

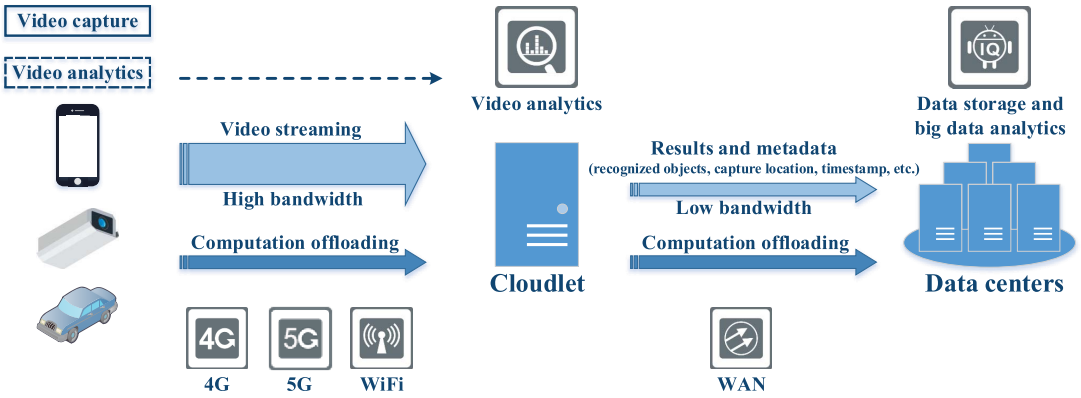

Fig. 6. *It is a typical scenario of video analytics over a three-tier architecture. In this scenario, edge devices with cameras are continuously transmitting video streaming via the high-bandwidth connection to nearby edge infrastructures (e.g., cloudlets), where quick and AI-based analysis is performed. Only the results and metadata are shipped to the cloud for further big data training and analytics.*

section, we showcase some typical application scenarios that benefit from edge computing and illustrate how computation offloading performs.

A. Video Analytics

Real-time video analytics is considered to be a killer application for edge computing [10]. With the explosive growth of cameras, such as traffic cameras, smartphones, in-vehicle cameras, and drone cameras, the analysis of video from these cameras drives the development of a variety of applications with great potential to impact human society, e.g., traffic control [151], self-driving cars [152], personal cognitive assistance [153], surveillance and security [154], and AR/VR [155]–[157].

However, large-scale live video analytics across a considerable amount of cameras is very challenging, because it is both compute-intensive and bandwidth-hungry. On the one hand, video analytics solely on edge devices (e.g., smartphones and drones) is inefficient due to their limited storage and computational power. On the other hand, sending video to the cloud entirely for analysis will easily exhaust network bandwidth and incur unacceptable latency. As a result, a distributed and hierarchical architecture of the public cloud and edge clouds is regarded as the only feasible way to meet the stringent latency and reduce network traffic in large-scale video analytics [10].

Fig. 6 shows a typical three-tier architecture of video analytics. It consists of lightweight processing at edge devices, rapid and moderate video analysis on edge infrastructures, and further large-scale big data analysis in the cloud. In this way, a large amount of video traffic is consumed by edge infrastructures, and only metadata (e.g., timestamp and capture location) and a small set of results (e.g., recognized objects) are sent to the cloud [42]. In this section, we investigate selective recent works about this topic and explore the associated computation offloading.

Hung *et al.* [158] proposed VideoEdge, a hierarchical architecture, including cameras, clusters, and the cloud for video analytics targeting at query optimizations. A video analytics query consists of a set of computer vision components running step by step. For instance, a typical pipeline is video decoding, object detecting, and object associating. Each component has various algorithms. For every video query, VideoEdge will select the optimal combination of the components' algorithms and place these components spanning clusters. VideoEdge assumes that no computing happens on cameras, and it offloads overall computer vision computation to private clusters (i.e., edge infrastructures). Evaluation results show that VideoEdge outperforms the two-tier (cameras and the cloud) video analytics system—VideoStorm [162].

First-person vision video analysis is an important part of video analytics. These videos are usually captured by head-up displays, e.g., Google Glass.[1] Sharing and analyzing these videos could create a significant value in terms of marketing and advertising, locating people, and public safety. However, first-person videos involve much personal privacy needing careful handling. GigaSight [73] is one of the prominent works for its handling. First, GigaSight defines the process of privacy (e.g., time and location) removing as *denaturing*, which is complicated as it needs in-depth analysis of every captured frame by employing sophisticated computer vision algorithms. Then, GigaSight constructs a three-tier cloudlet-cloud hybrid architecture, by which it migrates the denaturing to cloudlets with user-specific VMs and leaves some lightweight preprocessing on mobile phones (as proxies for head-up displays). In this architecture, cloudlets act as distributed processing and storage nodes to tag and index videos, and only the analysis results (for example, recognized faces and objects) associated with metadata (timestamp, location, the owner, and so on) are sent to the cloud for further big

[1]https://en.wikipedia.org/wiki/Google_Glass

Table 3 Selective Research Efforts Focusing on Video Analytics

Research	Proposed solutions	Edge devices	Edge nodes	Offloading granularity	What's to be offloaded
VideoEdge [158]	A hierarchical framework for video analytics and large-scale video query	IoT cameras	Private clusters	Full	The pipeline of computer vision components, such as video decoder, object detector, and associator
GigaSight [73]	A 3-tier architecture for first-person video storage, sharing, and content searching with privacy removing	Head-up displays, like Google Glass	Cloudlets	Tasks-Components	Video privacy removing, video tagging and indexing, and content searching
Wang *et al.* [159]	A privacy-aware and scalable live video analytics for face recognition	Devices with cameras, e.g., mobile phones	Cloudlets	Tasks-Components	Video privacy removing, the execution of analytics algorithms, e.g., DNN-based feature extraction and classification
Wang *et al.* [160]	A bandwidth-efficient drone video analytics framework employing adaptive policies to reduce video transmission	Drones	Cloudlets	Tasks-Components	The execution of computer vision algorithms
LAVEA [161]	A latency-aware 3-tier video analytics architecture enabling offloading tasks inter-edge collaboration	Smartphones and wearable devices	Container-based cloudlets	Tasks-Components	The execution of computer vision algorithms

data analysis [42]. GigaSight shows the efficiency of the cloudlet-cloud hybrid architecture in video analytics, and it is later extended by RTFace [159] to provide IoT services for face recognition.

Furthermore, with the increasing number of drones, recent efforts have started to explore the usage of drone cameras [163], [164]. Wang *et al.* [160] considered the scenario that a swarm of drones works collaboratively to execute some missions, such as survivor discovery, traffic management, and military missions, where these drones continuously transmit video for analysis. To save the bandwidth, they proposed a bandwidth-efficient video analytics architecture based on cloudlets and implemented strategies to reduce total data transmission. Specifically, they leverage the approach of DNNs to avoid transmitting "uninteresting" video frames and perform mission-specific optimizations, for instance, dynamically tuning the pipeline of video analytics according to the current mission. The implementations of these strategies are compute-intensive. Therefore, the authors employ computation offloading and ship tasks to cloudlets nearby.

Besides, Yi *et al.* [161] designed a latency-aware video analytics platform, LAVEA, which is serverless and able to provide offloading functionality. LAVEA builds a set of components for video analytics, comprising the profiler, monitor, and executor, and it optimizes task selection and placement to minimize the response delay. LAVEA is also a typical three-tier mobile-edge-cloud architecture. It offers a general video analytics processing framework and highlights its design of inter-edge collaboration for task placement, i.e., tasks are allowed to be executed across cloudlets.

Table 3 lists the comparison of these works. Fundamentally, real-time video analytics is a resource-intensive application. Thus, powerful servers could undeniably accelerate the process of analysis. For instance, a server with CPU i7 3.6 GHz can speed up object recognition 14× than a Samsung Galaxy Nexus phone and save 95% energy consumption [39]. Namely, computation offloading is highly necessary. However, in large-scale distributed and multitier video analytics, we need to address the challenges of latency, bandwidth, and resource provisioning. On the one hand, it is a tradeoff between the benefit of offloading and the cost of data transmission. If the bandwidth is limited, policies should be employed to reduce the data, for instance, filtering out less interesting segments of the video or delivering low-resolution video. These policies might lose some accuracy but make analysis work effectively in case of limited network resource. On the other hand, supporting scalability requires joint analysis and resource provisioning across multiple edges and the cloud. In other words, there is an urgent need to build platforms to provide functions for each aspect of video analytics, including video pipeline optimization, vision modules training, computational and network resource management, geo-distributed processing, and video storage.

Overall, real-time video analytics has shown its power in many aspects of human society. All the recent efforts made in this field reach a consensus that offloading compute-intensive algorithms in video analytics by exploiting the power of edge computing is a promising way or even the only way can meet the crisp latency.

B. Smart "Things"

IoT is a critical driving force behind fog computing [15] and edge computing. Integrated with edge computing and intelligent technologies, IoT generates new applications toward the fields of smart city [165], [166], smart home [167], smart grid [168], and so forth. However, IoT is challenging for device discovery, resource management, the latency of data analytics, and security/privacy [75]. In this section, we review some recent works for IoT based on edge computing.

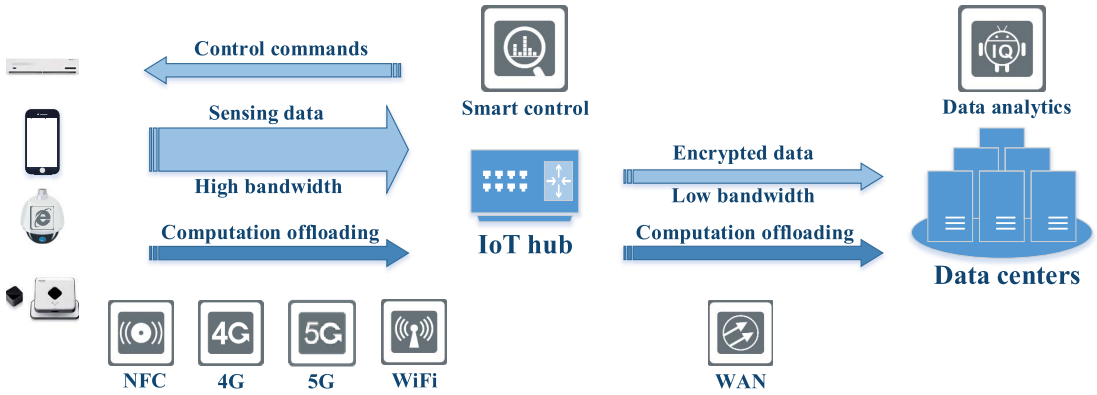

Fig. 7. *It is a typical scenario of smart home over a three-tier architecture. In this scenario, smart IoT devices in the house send sensing data via the high-speed wireless connection to the IoT hub, where immediate analysis and decision is performed to send back control commands, for example, open the air conditioner. These sensing data and commands are shipped to the cloud for further analysis and model training.*

FocusStack [169], Firework [170], and sFog [171] are general IoT frameworks. FocusStack is a multi-in-one edge-cloud collaboration platform. In the research, the authors made a key observation that location awareness is vital for resource allocation. For example, a fleet of connected drones is flying to complete some missions, and each drone is responsible for a target area. Then, if a mission happens in a specific area, only the target drone needs to answer the mission request. Based on the observation, FocusStack designs a two-tier architecture. First, the public cloud provides a location-based service to discover edge devices (e.g., drones and vehicles), which deal with specific tasks in light of location proximity. Second, FocusStack implements a container-based execution environment to equip edge devices with the capability of edge infrastructures. By this way, FocusStack offers the scalability and efficiency for IoT development.

Firework is a hybrid edge-cloud framework for video analytics. It is motivated by the scenario of collaborative video analytics, i.e., video resources are owned by different stakeholders. Sharing these resources and performing real-time video analytics could create great value. Firework builds a software architecture consisting of executor management, job management, and service management from the bottom up. It also provides programming APIs for developers to handle data processing and sharing. In addition, sFog is proposed to support seamless migrations of computation offloading services across servers, and it deals with the network congestion control and service handover.

In the field of smart home, EdgeOS$_H$ [172] and HomePad [173] are two typical solutions to address edge-based IoT issues. EdgeOS$_H$ is designed as an all-in-one OS for smart home. With EdgeOS$_H$, an IoT hub can act as an intelligent center bridging smart devices and the cloud. As an integrated system, EdgeOS$_H$ provides programming APIs and functional modules, including self-management (e.g., device registration, maintenance, and placement), data management, and privacy and security. HomePad is a security and privacy solution for smart home. It develops a programming model to run IoT applications as directed graphs of functions and empower users' security and privacy policies. Fig. 7 shows a typical architecture of smart home, in which an IoT hub bridges end devices and the cloud to deal with real-time device controls.

Furthermore, Farmbeats [174] is a data-driven IoT platform for smart agriculture. It designs a two-tier analytics architecture consisting of IoT gateways and the cloud. IoT gateways (e.g., cloudlets) deployed at farmers' home receive data from IoT base stations (responsible for data collecting) and perform local processing of immediate detailed data, for example, precision irrigation and virtual walkthroughs. Then, long-term summarized data, such as crop suggestions, seed distribution, and yield monitoring, are sent to the cloud for further analysis.

Table 4 shows the comparison of the above-mentioned works. IoT applications are typical data-driven and latency-sensitive. The traditional end-to-cloud data analytics architecture leads to a huge influx of data to the backbone network and cannot meet the stringent latency. The above-mentioned selective works demonstrate the benefit of edge computing with the multitier architecture and present some common challenges. First, in the area of IoT, privacy is a critical concern, as IoT data usually involve sensitive information. Therefore, the multitier architecture should carefully handle the tradeoff between efficiency and privacy protection. Second, IoT sensors and devices connect with heterogeneous wireless networks (such as Zigbee, Bluetooth, NFC, and Wi-Fi). To facilitate data transmission among these devices and edge servers, we need high-level network management (e.g., a software-defined networking (SDN)-based solution [175]). Third, there are a variety of data produced by IoT devices, and it is not easy

Table 4 Selective Research Efforts Focusing on Smart Things

Research	Proposed solutions	Edge nodes	Domain	What's to be offloaded
FocusStack [169]	A resource allocation and management framework for IoT with location-based situational awareness	Drones, vehicles, IoT hubs, etc.	General IoT frameworks	Typical IoT tasks, e.g., video analytics and car diagnostics
Firework [170]	A hybrid edge-cloud analytics framework for data sharing and processing	Edge servers, e.g., cloudlets	Video analytics	Compute-intensive tasks, e.g., video clipping
sFog [171]	A fog computing framework for mobile IoT applications providing seamless service handover	Base stations, IoT hubs, etc.	Mobile IoT applications, e.g., AR assisted tour guide	Compute-intensive tasks
EdgeOS$_H$ [172]	An all-in-one operating system for smart home providing functions of data management, program interface, security and privacy, etc.	IoT hubs	Smart home	Typical IoT tasks
HomePad [173]	A privacy-aware framework for smart home providing the ability to run and manage applications at IoT hubs	IoT hubs	Smart home	General tasks, e.g., picture encoding and motion detecting
Farmbeats [174]	A data-driven IoT platform for smart agriculture with the edge-cloud architecture	Base stations and IoT gateways (e.g., cloudlets)	Smart agriculture	Compute-intensive tasks, e.g., web service and the analysis of sensing data

to deal with such huge and various data. Consequently, programming models and software platforms are needed to provide flexibility and scalability.

C. Intelligent Vehicle Applications

In 2009, Google announced its plan to develop self-driving technology, which undeniably redefines the vehicle. Now, a vehicle is not only a transportation tool but also a sophisticated "computer on wheels." The connected and autonomous vehicles rich in a wide variety of vehicle applications become more intelligent. Zhang *et al.* [176] classified vehicle applications into four types: vehicle control and diagnostics, intelligent driver-assistant systems, third-party applications, and in-vehicle infotainment. Some of these applications (e.g., autonomous driving) involve intensive computing and a mass of data exchange [177], and an onboard computer is not sufficient to meet the power demand.

Meanwhile, connected vehicle analytics is now one of the critical technologies toward autonomous driving [178]. Joint analysis of information across vehicles requires ultra-low latency. Therefore, leveraging the power of edge computing is a natural choice. In this section, we investigate selective works toward vehicle applications and highlight the feature of computation offloading.

OpenVDAP [176] builds a unified vehicle data analytics platform for connected and autonomous vehicles. It supports a multitier computing architecture, consisting of onboard computers, neighboring vehicles, nearby edge nodes, and cloud servers. OpenVDAP designs a layered-pattern software architecture to provide data sensing services, OS functions, and edge-aware application libraries, and it enables features as *polymorphic service* and *elastic management*. Polymorphic service refers to the functionality to enable multiple execution pipelines. More specifically, take video analytics as an example; the process of video analytics can be split into two parts: motion detection and object recognition. Given this partitioning, OpenVDAP enables different offloading granularities. It can offload both the two parts to the edge and cloud or perform motion detection on the onboard computer but object recognition at the edge. Meanwhile, the elastic management can select the optimal execution pipeline in terms of computational resource demand, network condition, and the priority of applications.

Similar to OpenVDAP, AVE [179] builds a general framework for autonomous vehicles to support collaborative computing across vehicles on the road. The authors envision that autonomous vehicles equipped with onboard computers can share the computational resource, and then, they designed a scheduling algorithm to optimize job assignment. As a solution to connected autonomous driving, AVE enlarges the benefit of vehicle-to-vehicle communication and solves the job assignment problem based on ant colony optimization [184].

For autonomous driving control applications, [180] proposed a vehicle control system with edge-cloud collaboration. The authors consider that a vehicle control system based on remote servers is a promising way of collaborative control. Then, they introduced edge computing to solve the latency issue and built a two-tier edge-cloud infrastructure for vehicle control. Specifically, both the edge and the cloud equip with the same controllers, which perform vehicle control according to the sensing and control data from vehicles. This control can be automatically switched between the edge and the cloud in light of latency. In other words, if the latency of the cloud is lower than a threshold, the edge will hand over the control to the cloud, and vice versa. Furthermore, controllers on the edge and cloud share internal states to achieve better collaboration.

PreDriveID [181], Gremlin [182], and ParkMaster [183] are examples of third-party vehicle applications. PreDriveID creates a novel vehicle application that can distinguish drivers by human behavior analysis through in-vehicle data in terms of drivers' actions, such as seatbelt fastening, shifting gear, door opening, and door closing. This analysis facilitates in-car personalization,

Table 5 Selective Research Efforts Focusing on Vehicle Applications

Research	Proposed solutions	Edge nodes	Target applications	What's to be offloaded
OpenVDAP [176]	A unified data analytics platform to support the multi-tier architecture including on-board computers, neighboring vehicles, edge servers, and cloud servers	Enhanced base stations, roadside units, traffic signal systems, etc.	All types of applications	Compute-intensive tasks, e.g., object detection, object tracking, object recognition, and speech recognition
AVE [179]	A collaborative computing framework for autonomous vehicles enabling task scheduling across vehicles	On-board computers and other vehicles	All types of applications	General tasks
Sasaki *et al.* [180]	An edge-cloud collaboration vehicle control system for autonomous driving	Enhanced base stations	Autonomous driving control applications	The analysis of sensing and control data and the associated system control
PreDriveID [181]	A driver identification approach by human behavior analysis via in-vehicle data with the help of mobile edge nodes	Smartphones	A third-party behavior analysis application	The analysis of drivers' behavior
Gremlin [182]	A human-car interaction scheduling system to support application-initiated interactions management	Edge servers	A third-party behavior analysis application	The analysis of human-car interaction
ParkMaster [183]	A vehicle-mounted video analytics application for parking points detection	Smartphones	A third-party map and navigation application	Video analytics

insurance, and security. PreDriveID enables real-time driver differentiation with the minimal data set by constructing a two-tier architecture of smartphones and remote servers. The on-body smartphone acts as an edge infrastructure to collect sensing data and perform real-time analysis via offline models trained by remote servers. Similarly, Gremlin is an intelligent scheduling system for human–car interaction management. Specifically, Gremlin performs a joint analysis of driving conditions (e.g., driving on a sunny day with a few cars) and human–car interactions (e.g., playing music and reading an alert message) to schedule new interactions, i.e., according to the current driving condition to allow or defer a new interaction. To balance computational and bandwidth resources, Gremlin offloads the model training and the analysis of records to nearby edge servers. Besides, ParkMaster is another example that uses smartphones as edge nodes to perform data analytics. It creates a navigating and parking service by real-time video analytics using the cameras of drivers' smartphones.

Table 5 compares the above-mentioned works. We can find that intelligent vehicle applications drive the development of connected and autonomous vehicles. To realize the vision of vehicle-to-everything (V2X) [185], we need to build standardized software platforms (such as OpenVDAP) and hardware facilities (e.g., customized base stations for vehicles). Clearly, for vehicle applications, latency is the primary consideration. Therefore, policies should be designed to provide resilient computing, i.e., to choose a better computing mode in terms of latency (as [180] does). Besides, it is an urgent challenge to manage dynamic network connections since vehicles are moving fast on the road from one connection to an edge server to another. This topic is not discussed in the above-mentioned works.

D. Cloud Gaming

Cloud gaming is viewed as a killer application in cloud computing [187], in which a mobile device offloads the game running and rendering to the cloud and then plays game video sent back. Cloud gaming enables players enjoying complicated games without constantly updating their devices, and it also frees game operators to develop various versions of a game for different platforms. There have been an increasing number of companies to provide the cloud gaming service, such as Sony (PlayStation Now [188]), Nvidia (GeForce Now [189]), and Paperspace [190].

In general, cloud gaming can be broadly classified into two types: video streaming and graphics streaming, and Fig. 8 shows how they work. In the video streaming, a cloud server performs all compute-intensive tasks, such as running game logic, rendering scenes, and encoding video. Then, a mobile device only needs to send user commands and receive video streaming for displaying, as shown in Fig. 8(a). Alternatively, in the graphics streaming, the cloud server intercepts graphics commands when the game is running and compresses those commands as graphics streaming [191], [192], which is then executed on the mobile device for the game rendering, as shown in Fig. 8(b). From the view of computation offloading, they are considered as full offloading and partial offloading, respectively.

However, classic cloud gaming suffers from high network latency between cloud servers and mobile devices. An empirical study about 2504 clients on Amazon EC2 shows that over 30% of the latency is higher than 80 ms [194], which is regarded as a threshold for online games. Latency has a great impact on the game performance [195], especially for first-person shooter games [196]. Although efforts have been made on reducing the latency by changing the execution model of cloud gaming, such as Outatime [197] and Rhizome [198], the intrinsic long network RTT for the cloud cannot easily improve. In addition, cloud gaming is a typical bandwidth-hungry application. Thus, massive user connections will exert enormous pressures on the core network infrastructures of DCs.

In the last few years, there has emerged works on distributed processing for cloud gaming [199],

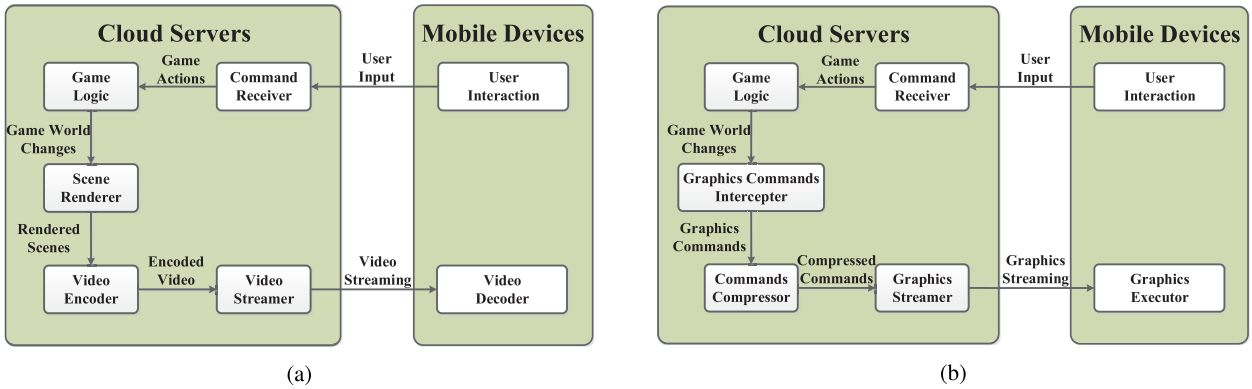

Fig. 8. *Workflows of cloud gaming with the types of (a) video streaming [186] and (b) graphics streaming.*

i.e., leveraging the power of edge clouds, such as [200], [201], and EC+ [193]. As a prominent example, EC+ provides a clear solution to this goal. Specifically, EC+ proposes a hybrid architecture of edge and cloud to augment VR-based massively multiplayer online games (VR-MMOGs). First, the authors inspect the workflow in VR-MMOGs and divide play-initiated events into two types: local view change events and global game events that are latency-sensitive and latency-tolerant, respectively. Based on this partition, EC+ leverages edge clouds to render view change events to meet the low-latency requirement and employs cloud servers for managing global game events (i.e., game logic), as shown in Fig. 9. Furthermore, it also designs an edge server selection algorithm for load balancing. In summary, EC+ offers a new insight that cloud gaming can benefit from edge computing by carefully designing the workflow of gaming and placing tasks onto suitable infrastructures (i.e., the edge or the cloud). This design achieves two advantages: bringing down the high latency and reducing the network traffic to the core network.

E. Summary and Insights

In this section, we survey selective killer applications in edge computing and explore the associated computation offloading. These applications are featured with data and compute-intensive as well as the stringent latency requirement. We gain some insights from the above-mentioned works in the following.

1) Leveraging the power of edge computing by offloading compute-intensive tasks to edge nodes can significantly reduce the latency, distribute data traffic, and finally improve the performance.
2) The hierarchical architecture of the edge and cloud is necessary. On the one hand, the edge reduces the latency. On the other hand, the traditional cloud can play the roles of edge discovery (e.g., in FocusStack), big data analytics (e.g., in GigaSight), global resource coordinating (e.g., in EC+), backup of task execution (e.g., in [180]), and so on.
3) Edge servers come in different form factors in different applications. They are usually cloudlets in video analytics, IoT hubs in IoT applications, and base stations in vehicle applications, and even smartphones can act as edge infrastructures (e.g., in PreDriveID and ParkMaster), which means whatever devices that have the computational capability to perform computing near the data source could be edge infrastructures, and it just caters to the philosophy of edge computing.

VI. OPPORTUNITIES AND FUTURE DIRECTIONS

In this section, we provide some perspectives about opportunities in the trend of computation offloading toward

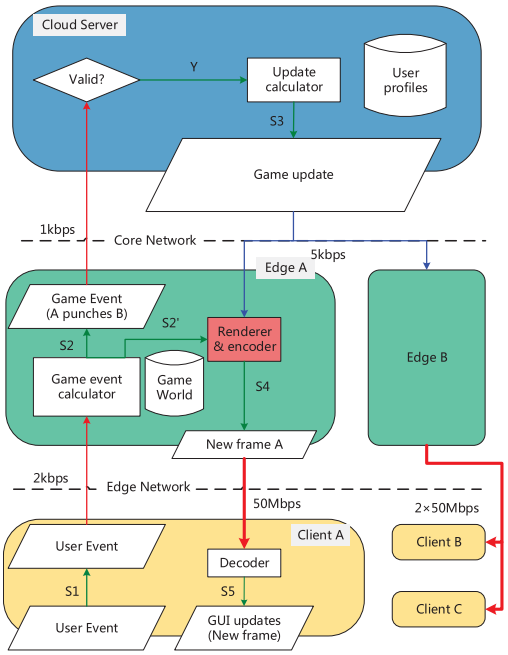

Fig. 9. *Workflow of EC+ [193]. It is a hybrid architecture, in which the cloud migrates game rendering and video encoding to the edge to reduce latency and network traffic.*

```
public sapphireclass User uses ConsistentCaching {

      String username;
      User[] followers;
      User[] friends;
      ...
      public String getUsername() {
            return username
      }
      public User[] getMyFollowers() {
            return followers;
      }
      public User[] getPeopleIFollow() {
            return friends;
      }
      ...
}
```

Fig. 10. *Example code snippet from Sapphire [202]. It demonstrates how to use the caching function by extending class* **ConsistentCaching**, *which is provided by Sapphire.*

edge computing associated with potential future directions.

A. Programming Model

Programming with computation offloading faces many challenges: runtime decision, resource management, and task execution. Early in MCC, research efforts have been made on designing programming models to address these challenges. As an example, Sapphire [202] implements a distributed programming platform that aims to mitigate programmers' effort to develop and deploy mobile/cloud applications. Sapphire separates the application logic from deployment logic, and it provides system components, such as computation offloading, serialization, fault tolerance, and caching. To equip with some functions, a developer only needs to extend specific classes, and an example code snippet is shown in Fig. 10.

However, extending this programming model for edge computing faces a twofold challenge: the handling of distributed execution across the three-tier architecture, i.e., mobile devices, cloudlets, and the cloud, and the coordinated resource managing of cloudlets over edge clouds. To address these challenges, both the academia and the industry are exploring ways to make edge computing programmable. For instance, Mortazavi *et al.* [203] proposed CloudPath, a multitier cloud computing framework toward the crystallization of edge computing. CloudPath aims to minimize the complexity of developing and deploying edge applications. It provides a RESTful [204] development model to encapsulate computation capabilities based on serverless cloud containers (similar to Amazon AWS Lambda [144]). Meanwhile, leading cloud computing providers are riding the wave of edge computing, and they are speeding up the deployment of new edge computing platforms, such as Azure IoT Edge [205], Amazon AWS IoT Greengrass [206], and Cisco IOx [207]. These platforms extend cloud capacities to edge infrastructures, enabling the data processing and analysis at the edge of the network and providing unified programming models for application development and deployment.

The above-mentioned programming models are general ones for computation offloading and resource management. However, different types of applications have different domain knowledge. Therefore, a domain-specific programming model for the target application would be more effective than a general one. SpanEdge [208] for streaming data (e.g., video streaming), EveryLite [209] for microtasks in IoT, and distributed deep neural network (DDNN) [210] for DNN are some prominent examples. SpanEdge provides a programming model for developers to specify which portions of an application should be proceeded near the data source. It builds a two-tier data analytics framework: geo-distributed analysis and centralized analysis. EveryLite is a scripting language to offer an elastic runtime for microtasks (lightweight tasks running in an embedded OS) in IoT. DDNN builds a distributed DNN training model on top of the edge-cloud hybrid architecture. In summary, these domain-specific programming models, which are tightly integrated with specific applications, have started to show the power on their fields. More and more applications will get benefit from their customized programming models.

Vehicle applications are various in their functions, data formats, computing models, and communication ways. Therefore, it is urgent to build a unified programming model to deal with these matters. In addition, mobile blockchain is also a potential future direction. Blockchain is famous for its usage of cryptographic currencies and has now found other compelling use cases, including financial services, personal identification, supply chain management, and healthcare. Because of the huge market in mobile commerce, blockchain is coming to mobile, ranging from the most popular mobile payment to any App development to share digital data. However, all of the use cases fundamentally rely on the mining in blockchain, i.e., to solve proof-of-work puzzles, which is compute-intensive and not resource-friendly to mobile devices. With the emergence of edge computing, people see the solution of offloading mining tasks to nearby edge infrastructures [211], [212]. To realize this vision, we need an integrated bottom–up programming model, including the support of blockchain mining, interfaces span mobile devices, edge, and cloud, and high-level APIs for App development. Besides, in recent years, there is a trend of software-defined something, such as SDN [213], software-defined storage architecture (SDS) [214], and software-defined batteries (SDBs) [215], and then, is there a software-defined offloading?

B. Offloading as a Service

So far, we discuss computation offloading only from the technical point of view. However, if envisioning OaaS [216], a clear service model needs to be built. Offloading requests from mobile applications ask for quick response and may be infrequent. The execution is generally

the running of a set of stateless functions with input data forwarding and results sent back. In OaaS, mobile application development companies deploy back-end offloading services. Then, mobile users can use these services to speed up the execution of applications. It is not difficult to find a status quo approach to offer these services in MCC because the cloud is already mature for its business model of on-demand resource provisioning and pricing. As discussed in Section IV-C2, the serverless architecture is well-suited for computation offloading. Then, App development companies can use this architecture provided by public cloud services (e.g., AWS Lambda) to deploy their code.

However, to extend OaaS to edge computing, there is a pressing need to address the challenge of who provides and manages edge infrastructures. Potential providers are cloud service companies. Cloud computing leaders are also top players in edge computing, such as Amazon, Microsoft, and Google. These tech giants could deploy edge servers close to end users to provide better services. Unfortunately, although they are starting up to develop software platforms (e.g., AWS IoT Greengrass), building out edge infrastructures still lags.

Alternatively, the rise of the proposed decentralized edge computing paradigm is a promising solution. Prominent decentralized platforms have emerged, such as SONM [217], iExec [218], and Golem [219]. These platforms make up of any possible users' machines, including PCs, private servers, and entire DCs. To organize these devices to work together in a logic level, they employ the blockchain technology to address the challenges of distributed resource management and financial incentives. The incentive and the reward are vital parts of any decentralized platform. Zhu *et al.* [220] proposed EdgeChain, a blockchain-based architecture for edge computing. EdgeChain builds a cost and pricing model to optimize application placement across multiple mobile network operators. As a result, the mobile network operator who provides service will get paid, so it will continue to share its computing power. However, these blockchain-based decentralized cloud/edge platforms still have room for improvement, such as security/privacy [221] and QoS guarantees [222].

Besides, in OaaS, coping with user mobility is a challenge that cannot be ignored. Mobile devices are the nature of high mobility, leading to the frequent change of the service execution environment. It gives rise to the problem of seamless offloading service migration across edge nodes, i.e., migrating services from the current edge node to the nearest edge node based on the user's location. It is instinctive to adopt the VM handoff technique [132] (discussed in Section IV-C2) and recently lightweight container-based handoff [223] to handle with the service migration. However, this way involves cumbersome operations of provisioning and managing servers. Furthermore, the support of user mobility requires to find the optimal edge node according to the user's location and make decisions on whether to offload functions in light of the finding. It is difficult to find solutions in the traditional IP-based network. In contrast, newly proposed NFaaS [224] is a promising way. NFaaS extends the NDN architecture [225], which is a proof of concept of ICN [226]. NFaaS enables the dynamic execution of user code without knowing edge node provisioning. Moreover, NFaaS uses the serverless architecture and builds on unikernels. Therefore, it can quickly provide the offloading service by identifying interesting name data (e.g., user demands for the offloading) instead of IP-based packets.

C. Security and Privacy

Security and privacy are the most important topics in edge computing [227]. In general, the principles of security and privacy for cloud computing [228] are also fit for edge computing. However, in edge computing, it faces new challenges. On the one hand, edge infrastructures (e.g., base stations and IoT hubs) are with the capability of location and network context aware [22]. As a result, data collected by these infrastructures involve much private information. For example, in smart home applications, an IoT hub collects information about GPS, the running of the power meter (if no one at home, it runs slowly), lamps' turning on and off, doors' locking and unlocking, and so on. The exposure of such information is dangerous for personal safety [229]. Besides, computation offloading ships user data to edge infrastructures (possible untrusted servers), giving rise to the problem of privacy protection while taking advantages of offloading. On the other hand, edge infrastructures are characterized as distributed and heterogeneous without centralized deployments. Thus, it is difficult to manage a considerable amount of edge servers and build integrated security and privacy policies.

First, a key step to protecting data privacy in computation offloading is the application partitioning, whereby code that will incur privacy leak is identified. Program analysis technologies are often employed to perform automatic privacy detections [230], [231]. In general, they construct a call graph for each application, model components' lifecycle, and perform data-flow analysis across the call graph to find potential privacy leaks. For instance, the code reading the user's contact list is regarded as a danger to leak personal privacy. Then, developers can separate the safe part of code from the tainted part and only ship the secure code [232]. If private data indeed require to be transmitted, encryption policies should be adopted. In addition, because edge devices have limited computational power, the tradeoff between effectiveness and efficiency of security and privacy policies should be considered [233], given that these policies are usually compute-intensive [234].

Second, for centralized DCs, owners are "tech giants" (e.g., Amazon, Google, Microsoft, and so on) and servers in DCs are relatively trusted. Oppositely, decentralized edge infrastructures comprise servers from any person or organization who wants to share computing power, including

potential malicious nodes. It is very difficult to deploy standardized security and privacy policies in such distributed and "autonomic" infrastructures, driving the need of building high-level governance. Fortunately, technologies (e.g., blockchain) have already been used for security and privacy in edge computing. In the blockchain-based model, an edge server in the chain is supposed to be trusted and therefore safe for computation offloading. Nevertheless, a variety of factors have impacts on this model, such as QoE, user's preferences, system scalability, and reward strategies [235].

Third, in the scenario of computation offloading, the connection of an edge device and an edge server is generally stateless, and the task execution usually employs the serverless platform (as discussed in Section IV-C2). Unlike the traditional client–server model, there are no gatekeepers between edge devices and edge servers in serverless platforms to maintain the connection status. Therefore, it needs authorization policies to provide secure connections. As a prominent example, AWS Lambda [144] treats every request equally untrusted and asks for the authorization each for one request, also known as "request-level" authorization [236].

VII. CONCLUSION

Edge computing is a technology that computing takes place at the edge of the network, close to the data source (e.g., end devices). Intrinsically, edge computing can be viewed as a distributed computing paradigm associated with computation offloading, which is initially designed for MCC. Since this paradigm will reduce latency and distribute network traffic, there is a significant trend of computation offloading toward edge computing. However, edge computing happens in distributed and heterogeneous environments, leading to new challenges for computation offloading in terms of application partitioning, task allocation, resource management, and distributed execution. In this paper, we investigate the recent efforts made on these challenges and clarify their solutions. We envision computing flows smoothly across the hierarchical architecture of edge devices, the edge, and the cloud. To realize this vision, we explore potential technologies toward this trend, such as serverless computing, blockchain, NFaaS, and integrated domain-specific programming models. Furthermore, we illustrate typical application scenarios that demonstrate the benefits of edge computing and computation offloading. We hope all the insights provided will serve as a valuable guide of the academic community in edge computing.

Acknowledgments

The authors would like to thank Prof. W. Shi and the anonymous reviewers who provided many helpful suggestions. ■

REFERENCES

[1] M. Satyanarayanan, "Mobile computing: The next decade," in *Proc. 1st ACM Workshop Mobile Cloud Comput. Services, Social Netw. Beyond*, Jun. 2010, pp. 5:1–5:6.

[2] A. Fox *et al.*, "Above the clouds: A Berkeley view of cloud computing," Tech. Rep., Feb. 2010. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[3] P. Mell and T. Grance, "The NIST definition of cloud computing," Tech. Rep., Sep. 2011. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-145/final

[4] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[5] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services*, Mar. 2010, pp. 49–62.

[6] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, Apr. 2011, pp. 301–314.

[7] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 945–953.

[8] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen, "COMET: Code offload by migrating execution transparently," in *Proc. 10th Symp. Operating Syst. Design Implement.*, Oct. 2012, pp. 93–106.

[9] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing public cloud providers," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, Nov. 2010, pp. 1–14.

[10] G. Ananthanarayanan *et al.*, "Real-time video analytics: The killer app for Edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[11] 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.

[12] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[13] K. Panetta. (2018). *Gartner Top 10 Strategic Technology Trends for 2019*. [Online]. Available: https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2019/

[14] D. Newman and O. Blanchard, "Edge computing index: From edge to enterprise," Tech. Rep., May 2018. [Online]. Available: https://futurumresearch.com/edge-computing-from-edge-to-enterprise/

[15] 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.*, Aug. 2012, pp. 13–16.

[16] "Fog computing and the Internet of Things: Extend the cloud to where the things are," Cisco, San Jose, CA, USA, White Paper, Apr. 2015. [Online]. Available: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[17] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," Feb. 2015, *arXiv:1502.01815*. [Online]. Available: https://arxiv.org/abs/1502.01815

[18] M. Uehara, "Mist computing: Linking cloudlet to fogs," in *Proc. Int. Conf. Comput. Sci./Intell. Appl. Inform.*, Jul. 2017, pp. 201–213.

[19] 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.

[20] P. G. Lopez *et al.*, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, 2015.

[21] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li, "Edge-oriented computing paradigms: A survey on architecture design and system management," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 39:1–39:34, Apr. 2018.

[22] M. Patel *et al.*, "Mobile-edge computing," ETSI, Sophia Antipolis, France, White Paper, Sep. 2014. [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf

[23] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[24] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing a key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper, Sep. 2015. [Online]. Available: https://infotech.report/view-resource.aspx?id=966

[25] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

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

[27] CBINSIGHTS. (2018). *What is Edge Computing?* [Online]. Available: https://www.cbinsights.com/research/what-is-edge-computing/

[28] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi, "5G wireless network slicing for eMBB, URLLC, and mMTC: A communication-theoretic view," *IEEE Access*, vol. 6, pp. 55765–55779, Sep. 2018.

[29] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker, "Agile application-aware adaptation for mobility," in *Proc. 16th ACM Symp. Oper. Syst. Princ.*, Oct. 1997, pp. 276–287.

[30] J. Flinn and M. Satyanarayanan, "Energy-aware adaptation for mobile applications," in *Proc. 17th ACM Symp. Oper. Syst. Princ.*, Apr. 1999, pp. 48–63.

[31] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *IEEE Pers. Commun.*, vol. 8, no. 4, pp. 10–17, Aug. 2001.

[32] Amazon. (2006). *Announcing Amazon Elastic Compute Cloud (Amazon ec2)—Beta*. [Online]. Available: https://aws.amazon.com/cn/about-aws/whats-new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2—beta/

[33] Apple Inc. Cupertino, CA, USA. (2019). *Build Your Apps for iOS*. [Online]. Available: https://developer.apple.com/ios/

[34] Google. (2019). *Build Anything on Android*. [Online]. Available: https://developer.android.com/

[35] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generat. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.

[36] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Mobile Computing, Applications, and Services*, M. Gris and G. Yang, Eds. Oct. 2012, pp. 59–79.

[37] 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.

[38] M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: At the leading edge of mobile-cloud convergence," in *Proc. Int. Conf. Mobile Comput., Appl. Services (MobiCASE)*, Nov. 2014, pp. 1–9.

[39] V. Bahl, "Emergence of micro datacenter (cloudlets/edges) for mobile computing," in *Proc. Int. Conf. Comput., Netw. Commun.*, May 2015.

[40] Network World. (2016). *Who's Got the Best Cloud Latency?* [Online]. Available: https://www.networkworld.com/article/3095022/cloud-computing/who-s-got-the-best-cloud-latency.html

[41] J. Pilz, M. Mehlhose, T. Wirth, D. Wieruch, B. Holfeld, and T. Haustein, "A tactile Internet demonstration: 1 ms ultra low delay for wireless communications towards 5G," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Apr. 2016, pp. 862–863.

[42] M. Satyanarayanan *et al.*, "Edge analytics in the Internet of Things," *IEEE Pervasive Comput.*, vol. 14, no. 2, pp. 24–31, Feb. 2015.

[43] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 421–434.

[44] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, to be published. doi: 10.1109/JPROC.2019.2918951.

[45] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd IEEE Workshop Hot Topics Web Syst. Technol. (HotWeb)*, Nov. 2015, pp. 73–78.

[46] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018.

[47] B. Krzanich. (2016). *Data is the New Oil in the Future of Automated Driving*. [Online]. Available: https://newsroom.intel.com/editorials/krzanich-the-future-of-automated-driving/

[48] Irvine Sensors Corporation. (2019). *Irvine Sensors Alert*. [Online]. Available: https://www.irvine-sensors.com/products/alert

[49] D. Raychaudhuri and N. B. Mandayam, "Frontiers of wireless and mobile communications," *Proc. IEEE*, vol. 100, no. 4, pp. 824–840, Apr. 2012.

[50] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2016, pp. 20–26.

[51] M. Chiang and W. Shi, "NSF workshop report on grand challenges in edge computing," Tech. Rep., Feb. 2016. [Online]. Available: http://iot.eng.wayne.edu/edge/index.ph

[52] *Universal Mobile Telecommunications System (UMTS); LTE; General UMTS Architecture*, document G. T. 23.101, ETSI Technical Specification, Jan. 2009.

[53] Z. Feng, S. George, J. Harkes, P. Pillai, R. Klatzky, and M. Satyanarayanan, "Edge-based discovery of training data for machine learning," in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2018, pp. 145–158.

[54] T. Taleb and A. Ksentini, "Follow me cloud: Interworking federated clouds and distributed mobile networks," *IEEE Netw.*, vol. 27, no. 5, pp. 12–19, Sep./Oct. 2013.

[55] A. Vulimiri, C. Curino, P. B. Godfrey, T. Jungblut, J. Padhye, and G. Varghese, "Global analytics in the face of bandwidth and regulatory constraints," in *Proc. 12th Symp. Netw. Syst. Design Implement.*, May 2015, pp. 323–336.

[56] L. A. Tawalbeh, W. Bakheder, and H. Song, "A mobile cloud computing model using the cloudlet scheme for big data applications," in *Proc. IEEE 1st Int. Conf. Connected Health, Appl., Syst. Eng. Technol.*, Jun. 2016, pp. 73–77.

[57] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE Symp. Comput. Commun.*, Jul. 2012, pp. 59–66.

[58] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Jan. 2008.

[59] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: A platform for high-performance Internet applications," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, pp. 2–19, 2010.

[60] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally distributed content delivery," *IEEE Internet Comput.*, vol. 6, no. 5, pp. 50–58, Sep. 2002.

[61] S. Li, L. Da Xu, and S. Zhao, "5G Internet of Things: A survey," *J. Ind. Inf. Integr.*, vol. 10, pp. 1–9, Jun. 2018.

[62] F. Lobillo *et al.*, "An architecture for mobile computation offloading on cloud-enabled lte small cells," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops*, Apr. 2014, pp. 1–6.

[63] I. Giannoulakis *et al.*, "The emergence of operator-neutral small cells as a strong case for cloud computing at the mobile edge," *Trans. Emerg. Telecommun. Technol.*, vol. 27, no. 9, pp. 1152–1159, Jul. 2016.

[64] M. Chiosi *et al.*, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," ETSI, Sophia Antipolis, France, White Paper, Oct. 2012. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf

[65] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[66] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[67] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, "Sociablesense: Exploring the trade-offs of adaptive sampling and computation offloading for social sensing," in *Proc. 17th Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2011, pp. 73–84.

[68] Nokia. (2018). *Edge Cloud Takes Compute Capacity to Where the Traffic is*. [Online]. Available: https://networks.nokia.com/solutions/edge-cloud

[69] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proc. 13th ACM Conf. Embedded Netw. Sensor Syst.*, Nov. 2015, pp. 155–168.

[70] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Peer-to-peer live streaming," in *Proc. 25th IEEE Int. Conf. Comput. Commun.*, Apr. 2006, pp. 1–10.

[71] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *Proc. 8th Int. Conf. Cloud Comput.*, Jun. 2015, pp. 9–16.

[72] S. Kosta, V. C. Perta, J. Stefa, P. Hui, and A. Mei, "Clone2clone (C2C): Peer-to-peer networking of smartphones on the cloud," in *Proc. 5th Workshop Hot Topics Cloud Comput.*, Oct. 2013, pp. 1–5.

[73] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan, "Scalable crowd-sourcing of video from mobile devices," in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2013, pp. 139–152.

[74] P. E. Ross, "Cloud computing's killer app: Gaming," *IEEE Spectr.*, vol. 46, no. 3, p. 14, Mar. 2009.

[75] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[76] F. Yang *et al.*, "A unified platform for data driven Web applications with automatic client-server partitioning," in *Proc. 16th Int. Conf. World Wide Web*, May 2007, pp. 341–350.

[77] D. Wagner and D. Schmalstieg, "First steps towards handheld augmented reality," in *Proc. 7th IEEE Int. Symp. Wearable Comput.*, Oct. 2003, pp. 127–135.

[78] D. W. F. van Krevelen and R. Poelman, "A survey of augmented reality technologies, applications and limitations," *Int. J. Virtual Reality*, vol. 9, no. 2, pp. 1–20, Jun. 2010.

[79] Y.-W. Kwon and E. Tilevich, "Energy-efficient and fault-tolerant distributed mobile execution," in *Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2012, pp. 586–595.

[80] Z. Chen *et al.*, "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 14:1–14:14.

[81] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas.*, Nov. 2009, pp. 280–293.

[82] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services*, Jun. 2012, pp. 225–238.

[83] C. Wang and Z. Li, "Parametric analysis for adaptive computation offloading," in *Proc. ACM SIGPLAN Conf. Program. Lang. Design Implement.*, Jun. 2004, pp. 119–130.

[84] L. Wang and M. Franz, "Automatic partitioning of object-oriented programs for resource-constrained mobile devices with multiple distribution objectives," in *Proc. 14th IEEE Int. Conf. Parallel Distrib. Syst.*, Dec. 2008, pp. 369–376.

[85] J. Liu, E. Ahmed, M. Shiraz, A. Gani, R. Buyya, and A. Qureshi, "Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions," *J. Netw. Comput. Appl.*, vol. 48, pp. 99–117, Feb. 2015.

[86] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: A partition scheme," in *Proc. ACM Int. Conf. Compil., Archit., Synth. Embedded Syst.*, Oct. 2001, pp. 238–246.

[87] R. Newton, S. Toledo, L. Girod, H. Balakrishnan, and S. Madden, "Wishbone: Profile-based partitioning for sensornet applications," in *Proc. 6th USENIX Symp. Netw. Syst. Design Implement.*, Apr. 2009, pp. 395–408.

[88] R. K. Balan, M. Satyanarayanan, S. Y. Park, and T. Okoshi, "Tactics-based remote execution for

mobile computing," in *Proc. 1st Int. Conf. Mobile Syst., Appl. Services*, May 2003, pp. 273–286.

[89] E. Tilevich and Y. Smaragdakis, "J-orchestra: Automatic java application partitioning," in *Proc. Eur. Conf. Object-Oriented Program.*, Jun. 2002, pp. 178–204.

[90] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling interactive perception applications on mobile devices," in *Proc. 9th Int. Conf. Mobile Syst., Appl., Services*, Jun. 2011, pp. 43–56.

[91] G. C. Hunt *et al.*, "The coign automatic distributed partitioning system," in *Proc. 3rd Symp. Oper. Syst. Design Implement.*, Feb. 1999, pp. 187–200.

[92] L. Lin, P. Li, X. Liao, H. Jin, and Y. Zhang, "Echo: An edge-centric code offloading system with quality of service guarantee," *IEEE Access*, vol. 7, pp. 5905–5917, 2019.

[93] G. Kiczales *et al.*, "Aspect-oriented programming," in *Proc. 11th Eur. Conf. Object-Oriented Program.*, Jun. 1997, pp. 220–242.

[94] J. Liu *et al.*, "Energy efficient GPS sensing with cloud offloading," in *Proc. 10th ACM Conf. Embedded Netw. Sensor Syst.*, Toronto, ON, Canada, Nov. 2012, pp. 85–98.

[95] X. Jin, Y. Liu, W. Fan, F. Wu, and B. Tang, "Multisite computation offloading in dynamic mobile cloud environments," *Sci. China Inf.*, vol. 60, no. 8, p. 089301, Feb. 2017.

[96] K. Sinha and M. Kulkarni, "Techniques for fine-grained, multi-site computation offloading," in *Proc. 11th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2011, pp. 184–194.

[97] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, pp. 3317–3329, Dec. 2015.

[98] D. Parmar, A. S. Kumar, A. Nivangune, P. Joshi, and U. P. Rao, "Discovery and selection mechanism of cloudlets in a decentralized mcc environment," in *Proc. Int. Conf. Mobile Softw. Eng. Syst.*, May 2016, pp. 15–16.

[99] K. Ha, "System infrastructure for mobile-cloud convergence," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2016.

[100] S. Cheshire. (2019). *Zero Configuration Networking (Zeroconf)*. [Online]. Available: http://www.zeroconf.org/

[101] Avahi Org. (2019). *Welcome to Avahi*. [Online]. Available: https://www.avahi.org/

[102] Apple Inc. (2019). *Bonjour Support*. [Online]. Available: https://support.apple.com/bonjour

[103] H. H. Harvey, Y. Mao, Y. Hou, and B. Sheng, "EDOS: Edge assisted offloading system for mobile devices," in *Proc. 26th Int. Conf. Comput. Commun. Netw.*, Jul. 2017, pp. 1–9.

[104] 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.*, Apr. 2016, pp. 1–9.

[105] 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.

[106] J. Gedeon *et al.*, "From cell towers to smart street lamps: Placing cloudlets on existing urban infrastructures," in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2018, pp. 187–202.

[107] J. Flinn, "Cyber foraging: Bridging mobile and cloud computing," *Synth. Lectures Mobile Pervas. Comput.*, vol. 7, no. 2, pp. 1–103, Sep. 2012.

[108] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[109] A. Cidon, T. M. London, S. Katti, C. Kozyrakis, and M. Rosenblum, "MARS: Adaptive remote execution for multi-threaded mobile devices," in *Proc. 3rd ACM SOSP Workshop Netw., Syst., Appl. Mobile Handhelds*, Oct. 2011, pp. 1:1–1:6.

[110] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[111] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[112] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[113] M. Molina, O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Joint scheduling of communication and computation resources in multiuser wireless application offloading," in *Proc. 25th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun.*, Sep. 2014, pp. 1093–1098.

[114] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu, "Efficient resource allocation for on-demand mobile-edge cloud computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8769–8780, Sep. 2018.

[115] V. K. Vavilapalli *et al.*, "Apache hadoop YARN: Yet another resource negotiator," in *Proc. 4th Annu. Symp. Cloud Comput.*, Oct. 2013, pp. 5:1–5:16.

[116] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes, "Omega: Flexible, scalable schedulers for large compute clusters," in *Proc. 8th ACM Eur. Conf. Comput. Syst.*, Apr. 2013, pp. 351–364.

[117] L. Lin, P. Li, J. Xiong, and M. Lin, "Distributed and application-aware task scheduling in edge-clouds," in *Proc. 14th Int. Conf. Mobile Ad-Hoc Sensor Netw.*, Dec. 2018, pp. 165–170.

[118] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: Distributed, low latency scheduling," in *Proc. 24th ACM Symp. Oper. Syst. Principles*, Nov. 2013, pp. 69–84.

[119] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1094–1104, Oct. 2001.

[120] X. Ren, G. Ananthanarayanan, A. Wierman, and M. Yu, "Hopper: Decentralized speculation-aware cluster scheduling at scale," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 379–392.

[121] S. Rashidi and S. Sharifian, "Cloudlet dynamic server selection policy for mobile task off-loading in mobile cloud computing using soft computing techniques," *J. Supercomputing*, vol. 73, no. 9, pp. 3796–3820, Sep. 2017.

[122] Android-x86. (2019). *Android-x86 Run Android on your PC*. [Online]. Available: http://www.android-x86.org/

[123] J. Flinn, S. Park, and M. Satyanarayanan, "Balancing performance, energy, and quality in pervasive computing," in *Proc. 22nd Int. Conf. Distrib. Comput. Syst.*, Jul. 2002, pp. 217–226.

[124] P. Barham *et al.*, "Xen and the art of virtualization," in *Proc. 19th ACM Symp. Oper. Syst. Princ.*, Oct. 2003, pp. 164–177.

[125] Google. Android. (2019). *Android Interface Definition Language (AIDL)*. [Online]. Available: https://developer.android.com/guide/components/aidl

[126] Google. (2019). *Application Fundamentals*. [Online]. Available: https://developer.android.com/guide/components/fundamentals

[127] Google Android Source. (2019). *Art and Dalvik*. [Online]. Available: https://source.android.com/devices/tech/dalvik

[128] Microsoft. (2019). *Common Language Runtime (CLR) Overview*. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/clr

[129] G. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, Oct. 2014, pp. 1440–1446.

[130] 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*, Nov. 2014, pp. 116–124.

[131] 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*, Jun. 2013, pp. 153–166.

[132] K. Ha *et al.*, "You can teach elephants to dance: Agile VM handoff for edge computing," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 12:1–12:14.

[133] C. Clark *et al.*, "Live migration of virtual machines," in *Proc. 2nd Conf. Symp. Netw. Syst. Design Implement.*, May 2005, pp. 273–286.

[134] K. Ha and M. Satyanarayanan, "Openstack++ for cloudlet deployment," Tech. Rep., 2015. [Online]. Available: https://www.cs.cmu.edu/~satya/docdir/CMU-CS-15-123.pdf

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

[136] R. Bauer. (2018). *What's the Difference: VMS vs Containers*. [Online]. Available: https://www.backblaze.com/blog/vm-vs-containers/

[137] Linux Containers. (2019). *Infrastructure for Container Projects*. [Online]. Available: https://linuxcontainers.org/

[138] OpenVZ. (2019). *Open Source Container-Based Virtualization for Linux*. [Online]. Available: https://openvz.org/

[139] Docker. (2019). *Enterprise Container Platform for High-Velocity Innovation*. [Online]. Available: https://www.docker.com/

[140] C. Pahl and B. Lee, "Containers and clusters for edge cloud architectures—A technology review," in *Proc. 3rd Int. Conf. Future Internet Things Cloud*, Aug. 2015, pp. 379–386.

[141] S. Wu, C. Niu, J. Rao, H. Jin, and X. Dai, "Container-based cloud platform for mobile computation offloading," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, May 2017, pp. 123–132.

[142] S. Wu, C. Mei, H. Jin, and D. Wang, "Android unikernel: Gearing mobile code offloading towards edge computing," *Future Gener. Comput. Syst.*, vol. 86, pp. 694–703, Sep. 2018.

[143] R. Brown. (2017). *Running Containers on Bare Metal vs. VMs: Performance and Benefits*. [Online]. Available: https://www.stratoscale.com/blog/containers/running-containers-on-bare-metal/

[144] Amazon. (2019). *AWS Lambda*. [Online]. Available: https://aws.amazon.com/lambda/

[145] Google. (2019). *Google Cloud Functions*. [Online]. Available: https://cloud.google.com/functions/

[146] Microsoft. (2019). *Azure Functions*. [Online]. Available: https://functions.azure.com/

[147] (2019). *IBM Cloud Functions*. [Online]. Available: https://www.ibm.com/cloud/functions

[148] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Serverless computation with openlambda," in *Proc. 8th USENIX Workshop Hot Topics Cloud Comput.*, Jun. 2016, pp. 1–7.

[149] L. Baresi, D. F. Mendonça, and M. Garriga, "Empowering low-latency applications through a serverless edge computing architecture," in *Proc. Service-Oriented Cloud Comput.*, Sep. 2017, pp. 196–210.

[150] I. Baldini *et al.*, "Serverless computing: Current trends and open problems," in *Proc. Res. Adv. Cloud Comput.*, Dec. 2017, pp. 1–20.

[151] F. Loewenherz, V. Bahl, and Y. Wang, "Video analytics towards vision zero," *ITE J.*, vol. 87, no. 3, pp. 25–28, Mar. 2017.

[152] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, Dec. 2017.

[153] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. 12th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2014, pp. 68–81.

[154] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *Proc.*

21st Annu. Int. Conf. Mobile Comput. Netw., Sep. 2015, pp. 426–438.
[155] P. Jain, J. Manweiler, and R. R. Choudhury, "OverLay: Practical mobile augmented reality," in *Proc. 13th Annu. Int. Conf. Mobile Syst., Appl., Services*, May 2015, pp. 331–344.
[156] Y. Li and W. Gao, "MUVR: Supporting multi-user mobile virtual reality with resource constrained edge cloud," in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2018, pp. 1–16.
[157] R. Schmoll, S. Pandi, P. J. Braun, and F. H. P. Fitzek, "Demonstration of VR/AR offloading to mobile edge cloud for low latency 5G gaming application," in *Proc. 15th IEEE Annu. Consum. Commun. Netw. Conf.*, Jan. 2018, pp. 1–3.
[158] C. Hung *et al.*, "VideoEdge: Processing camera streams using hierarchical clusters," in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2018, pp. 115–131.
[159] J. Wang, B. Amos, A. Das, P. Pillai, N. Sadeh, and M. Satyanarayanan, "A scalable and privacy-aware IoT service for live video analytics," in *Proc. 8th ACM Multimedia Syst. Conf.*, Jun. 2017, pp. 38–49.
[160] J. Wang *et al.*, "Bandwidth-efficient live video analytics for drones via edge computing," in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2018, pp. 159–173.
[161] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "LAVEA: Latency-aware video analytics on edge computing platform," in *Proc. 37th Int. Conf. Distrib. Comput. Syst.*, Jun. 2017, pp. 2573–2574.
[162] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *Proc. 14th USENIX Conf. Netw. Syst. Design Implement.*, Mar. 2017, pp. 377–392.
[163] X. Wang, A. Chowdhery, and M. Chiang, "Networked drone cameras for sports streaming," in *Proc. 37th Int. Conf. Distrib. Comput. Syst.*, Jun. 2017, pp. 308–318.
[164] A. Chowdhery and M. Chiang, "Model predictive compression for drone video analytics," in *Proc. IEEE Int. Conf. Sens., Commun. Netw.*, Jun. 2018, pp. 1–5.
[165] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.
[166] G. Liang, J. Ca, X. Liu, and J. Liang, "Smart world: A better world," *Sci. China Inf.*, vol. 59, no. 4, p. 043401, Feb. 2016.
[167] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," *J. Cleaner Prod.*, vol. 140, no. 3, pp. 1454–1464, 2017.
[168] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid—The new improved power grid: A survey," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 944–980, Apr. 2012.
[169] 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.*, Oct. 2016, pp. 179–191.
[170] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Firework: Data processing and sharing for hybrid cloud-edge analytics," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 9, pp. 2004–2017, Sep. 2018.
[171] W. Bao *et al.*, "sFOG: Seamless fog computing environment for mobile IoT applications," in *Proc. 21st ACM Int. Conf. Model., Anal. Simul. Wireless Mobile Syst.*, Oct. 2018, pp. 127–136.
[172] J. Cao, L. Xu, R. Abdallah, and W. Shi, "EdgeOS_H: A home operating system for Internet of everything," in *Proc. 37th Int. Conf. Distrib. Comput. Syst.*, Jun. 2017, pp. 1756–1764.
[173] I. Zavalyshyn, N. O. Duarte, and N. Santos, "HomePad: A privacy-aware smart hub for home environments," in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2018, pp. 58–73.
[174] D. Vasisht *et al.*, "FarmBeats: An IoT platform for data-driven agriculture," in *Proc. 14th USENIX Conf. Netw. Syst. Design Implement.*, Mar. 2017, pp. 515–528.
[175] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.
[176] Q. Zhang *et al.*, "OpenVDAP: An open vehicular data analytics platform for CAVs," in *Proc. 38th Int. Conf. Distrib. Comput. Syst.*, Jul. 2018, pp. 1310–1320.
[177] Y. Wang, S. Liu, X. Wu, and W. Shi, "CAVBench: A Benchmark Suite for Connected and Autonomous Vehicles," in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2018, pp. 30–42.
[178] D. Grewe, M. Wagner, M. Arumaithurai, I. Psaras, and D. Kutscher, "Information-centric mobile edge computing for connected vehicle environments: Challenges and research directions," in *Proc. Workshop Mobile Edge Commun.*, Aug. 2017, pp. 7–12.
[179] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, Dec. 2017.
[180] K. Sasaki, N. Suzuki, S. Makido, and A. Nakao, "Vehicle control system coordinated between cloud and mobile edge computing," in *Proc. 55th Annu. Conf. Soc. Instrum. Control Eng. Jpn.*, Sep. 2016, pp. 1122–1127.
[181] G. Kar, S. Jain, M. Gruteser, J. Chen, F. Bai, and R. Govindan, "PredriveID: Pre-trip driver identification from in-vehicle data," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 2:1–2:12.
[182] K. Lee, J. Flinn, and B. D. Noble, "Gremlin: Scheduling interactions in vehicular computing," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 4:1–4:13.
[183] G. Grassi, K. Jamieson, P. Bahl, and G. Pau, "Parkmaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 16:1–16:14.
[184] Y.-C. Liang and A. E. Smith, "An ant colony optimization algorithm for the redundancy allocation problem," *IEEE Trans. Rel.*, vol. 53, no. 3, pp. 417–423, Sep. 2004.
[185] S. Chen *et al.*, "Vehicle-to-everything (V2X) services supported by LTE-based systems and 5G," *IEEE Commun. Standards Mag.*, vol. 1, no. 2, pp. 70–76, Jun. 2017.
[186] W. Cai *et al.*, "The future of cloud gaming," *Proc. IEEE*, vol. 104, no. 4, pp. 687–691, Apr. 2016.
[187] P. E. Ross, "Cloud computing's killer app: Gaming," *IEEE Spectr.*, vol. 46, no. 3, p. 14, Mar. 2009.
[188] Sony. (2019). *Playstation Now*. [Online]. Available: https://www.playstation.com/en-us/explore/playstation-now/
[189] Nvidia. (2019). *Geforce Now*. [Online]. Available: https://www.nvidia.com/en-us/geforce/products/geforce-now/
[190] Paperspace. (2019). *Paperspace Gaming*. [Online]. Available: https://www.paperspace.com/gaming
[191] L. Lin *et al.*, "LiveRender: A cloud gaming system based on compressed graphics streaming," in *Proc. 22nd ACM Int. Conf. Multimedia*, Nov. 2014, pp. 347–356.
[192] W. Zhang, X. Liao, P. Li, H. Jin, and L. Lin, "ShareRender: Bypassing GPU virtualization to enable fine-grained resource sharing for cloud gaming," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 324–332.
[193] 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.*, Oct. 2017, pp. 8:1–8:14.
[194] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Proc. 11th Annu. Workshop Netw. Syst. Support Games*, Nov. 2012, pp. 2:1–2:6.
[195] M. Claypool and D. Finkel, "The effects of latency on player performance in cloud-based games," in *Proc. 13th Annu. Workshop Netw. Syst. Support Games*, Dec. 2014, pp. 1–6.
[196] K. Chen, Y. Chang, H. Hsu, D. Chen, C. Huang, and C. Hsu, "On the quality of service of cloud gaming systems," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 480–495, Feb. 2014.
[197] K. Lee *et al.*, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proc. 13th Annu. Int. Conf. Mobile Syst., Appl., Services*, May 2015, pp. 151–165.
[198] R. Shea, D. Fu, and J. Liu, "Rhizome: Utilizing the public cloud to provide 3D gaming infrastructure," in *Proc. 6th ACM Multimedia Syst. Conf.*, Mar. 2015, pp. 97–100.
[199] H. Tian, D. Wu, J. He, Y. Xu, and M. Chen, "On achieving cost-effective adaptive cloud gaming in geo-distributed data centers," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 2064–2077, Dec. 2015.
[200] W. Cai, V. C. M. Leung, and L. Hu, "A cloudlet-assisted multiplayer cloud gaming system," *Mobile Netw. Appl.*, vol. 19, no. 2, pp. 144–152, 2014.
[201] Y. Lin and H. Shen, "CloudFog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, pp. 431–445, Feb. 2017.
[202] I. Zhang *et al.*, "Customizable and extensible deployment for mobile/cloud applications," in *Proc. 11th USENIX Symp. Operat. Syst. Design Implement.*, Oct. 2014, pp. 97–112.
[203] S. H. Mortazavi, M. Salehe, C. S. Gomes, C. Phillips, and E. de Lara, "CloudPath: A multi-tier cloud computing framework," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 20:1–20:13.
[204] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful Web services vs. 'big' Web services: Making the right architectural decision," in *Proc. 17th Int. Conf. World Wide Web*, Apr. 2008, pp. 805–814.
[205] Microsoft. (2019). *Azure Iot Edge*. [Online]. Available: https://azure.microsoft.com/en-us/services/iot-edge/
[206] Amazon. (2019). *AWS Iot Greengrass*. [Online]. Available: https://aws.amazon.com/greengrass/
[207] Cisco. (2019). *Cisco IOx*. [Online]. Available: https://www.cisco.com/c/en/us/products/cloud-systems-management/iox/index.html
[208] H. P. Sajjad, K. Danniswara, A. Al-Shishtawy, and V. Vlassov, "SpanEdge: Towards unifying stream processing over central and near-the-edge data centers," in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2016, pp. 168–178.
[209] Z. Li, X. Peng, L. Chao, and Z. Xu, "EveryLite: A lightweight scripting language for micro tasks in IoT systems," in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2018, pp. 381–386.
[210] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. 37th Int. Conf. Distrib. Comput. Syst.*, Jun. 2017, pp. 328–339.
[211] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.
[212] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11008–11021, Nov. 2018.
[213] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[214] E. Thereska *et al.*, "IOFlow: A software-defined storage architecture," in *Proc. 24th ACM Symp. Oper. Syst. Princ.*, Nov. 2013, pp. 182–196.
[215] A. Badam *et al.*, "Software defined batteries," in *Proc. 25th Symp. Oper. Syst. Princ.*, Oct. 2015, pp. 215–229.
[216] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "COSMOS: Computation offloading as a service for mobile devices," in *Proc. 15th ACM Int. Symp. Mobile Ad Hoc Netw.*

Comput., Aug. 2014, pp. 287–296.
[217] SONM. (2019). *Decentralized Fog Computing Platform*. [Online]. Available: https://sonm.com/
[218] iExec. (2019). *Blockchain-Based Decentralized Cloud Computing*. [Online]. Available: https://iex.ec/
[219] Golem. 2019. *Golem Network*. [Online]. Available: https://golem.network/
[220] H. Zhu, C. Huang, and J. Zhou, "EdgeChain: Blockchain-based multi-vendor mobile edge application placement," in *Proc. 4th IEEE Conf. Netw. Softwarization Workshops*, Jun. 2018, pp. 222–226.
[221] I. Psaras, "Decentralised edge-computing and IoT through distributed trust," in *Proc. 16th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2018, pp. 505–507.
[222] R. B. Uriarte and R. De Nicola, "Blockchain-based decentralized cloud/fog solutions: Challenges, opportunities, and standards," *IEEE Commun. Standards Mag.*, vol. 2, no. 3, pp. 22–28, Sep. 2018.
[223] 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, pp. 11:1–11:13.
[224] M. Król and I. Psaras, "NFaaS: Named function as a service," in *Proc. 4th ACM Conf. Inf.-Centric Netw.*, Sep. 2017, pp. 134–144.
[225] L. Zhang *et al.*, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014.
[226] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.
[227] R. Roman *et al.*, "Mobile edge computing, fog: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.
[228] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 843–859, 5th Quart., 2013.
[229] R. Trimananda, A. Younis, B. Wang, B. Xu, B. Demsky, and G. Xu, "Vigilia: Securing smart home edge computing," in *Proc. IEEE/ACM Symp. Edge Comput.*, Oct. 2018, pp. 74–89.
[230] S. Arzt *et al.*, "FlowDroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," in *Proc. 35th ACM SIGPLAN Conf. Program. Lang. Des. Implement.*, Jun. 2014, pp. 259–269.
[231] W. Enck *et al.*, "TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Trans. Comput. Syst.*, vol. 32, no. 2, p. 5, Jun. 2014.
[232] M. Al-Mutawa and S. Mishra, "Data partitioning: An approach to preserving data privacy in computation offload in pervasive computing systems," in *Proc. 10th ACM Symp. QoS Secur. Wireless Mobile Netw.*, Oct. 2014, pp. 51–60.
[233] J. Liu, K. Kumar, and Y. Lu, "Tradeoff between energy savings and privacy protection in computation offloading," in *Proc. ACM/IEEE Int. Symp. Low-Power Electron. Design*, Aug. 2010, pp. 213–218.
[234] J. Xiong *et al.*, "Enhancing privacy and availability for data clustering in intelligent electrical service of IoT," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1530–1540, Apr. 2019.
[235] A. G. Tasiopoulos, O. Ascigil, I. Psaras, and G. Pavlou, "Edge-map: Auction markets for edge resource provisioning," in *Proc. 19th Int. Symp. World Wireless, Mobile Multimedia Netw.*, Jun. 2018, pp. 14–22.
[236] G. Adzic and R. Chatley, "Serverless computing: Economic and architectural impact," in *Proc. 11th Joint Meeting Found. Softw. Eng.*, Sep. 2017, pp. 884–889.

ABOUT THE AUTHORS

Li Lin (Member, IEEE) received the B.E. and M.S. degrees in computer science and technology from Sichuan University, Chengdu, China, in 2005 and 2008, respectively. He is currently working toward the Ph.D. degree in computer science and engineering at the Huazhong University of Science and Technology (HUST), Wuhan, China.

He is also a Lecturer with Fujian Normal University, Fuzhou, China. His current research interests include the areas of mobile cloud computing and edge computing.

Xiaofei Liao (Member, IEEE) received the Ph.D. degree in computer science and engineering from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2005.

He is currently a Professor with the School of Computer Science and Engineering, HUST. He has published papers in important conferences, including ASPLOS, ICS, USENIX ATC, CGO, and PACT. His current research interests include the areas of system virtualization, system software, and cloud computing.

Hai Jin (Fellow, IEEE) received the Ph.D. degree in computer engineering from Huazhong University of Science and Technology (HUST), Wuhan, China, in 1994.

He was with The University of Hong Kong, Hong Kong, from 1998 to 2000. He was a Visiting Scholar with the University of Southern California, Los Angeles, CA, USA, from 1999 to 2000. He is currently a Cheung Kung Scholars Chair Professor of computer science and engineering with HUST. He is also the Chief Scientist of ChinaGrid, the largest grid computing project in China, and the National 973 Basic Research Program Project of Virtualization Technology of Computing System and Cloud Security. He has coauthored 22 books and published over 800 research papers. His current research interests include computer architecture, virtualization technology, cluster computing and cloud computing, peer-to-peer computing, network storage, and network security.

Dr. Jin is also a Fellow of the China Computer Federation and a member of the Association for Computing Machinery. In 1996, he received the German Academic Exchange Service Fellowship to visit the Technical University of Chemnitz, Chemnitz, Germany. He also received the Excellent Youth Award from the National Science Foundation of China in 2001.

Peng Li (Member, IEEE) received the B.S. degree from Huazhong University of Science and Technology, Wuhan, China, in 2007, and the M.S. and Ph.D. degrees from The University of Aizu, Aizuwakamatsu, Japan, in 2009 and 2012, respectively.

He is currently an Associate Professor with The University of Aizu. His current research interests include cloud computing, Internet-of-Things, big data systems, and the related wired and wireless networking problems.

Dr. Li was a recipient of the Young Author Award of the IEEE Computer Society Japan Chapter in 2014 and the Best Paper Award of IEEE TrustCom 2016. He has supervised students to win the First Prize of the IEEE ComSoc Student Competition in 2016.