Edge-Oriented Computing Paradigms: A Survey on Architecture Design and System Management

CHAO LI, Shanghai Jiao Tong University and Beijing Advanced Innovation Center for Imaging Technology
YUSHU XUE, National University of Singapore and Shanghai Jiao Tong University
JING WANG, Beijing Advanced Innovation Center for Imaging Technology, Capital Normal University
WEIGONG ZHANG, College of Information Engineering, Capital Normal University
TAO LI, University of Florida

While cloud computing has brought paradigm shifts to computing services, researchers and developers have also found some problems inherent to its nature such as bandwidth bottleneck, communication overhead, and location blindness. The concept of fog/edge computing is therefore coined to extend the services from the core in cloud data centers to the edge of the network. In recent years, many systems are proposed to better serve ubiquitous smart devices closer to the user. This paper provides a complete and up-to-date review of edge-oriented computing systems by encapsulating relevant proposals on their architecture features, management approaches, and design objectives.

Categories and Subject Descriptors: **• General and reference ~ Surveys and overviews • Computer systems organization ~ Distributed architectures**

General Terms: Definition, Architecture, Management, Performance, Efficiency

Additional Key Words and Phrases: distributed cloud, fog computing, edge computing, ubiquitous data processing, architecture design, resource management.

1. INTRODUCTION

We are in a world where smart devices and appliances are starting to overwhelm people's daily lives. With the increasing intelligence in all sorts of machine instances such as watches, phones, cameras, drones, cars, set-top boxes and radio access gateways, it is becoming increasingly rewarding to utilize these ubiquitous and massive resources of smart devices. In light of that, *fog computing*, an emerging idea is bred to deliver wider scope of services by extending cloud computing to devices that produce and act on local data [Cisco, 2015]. At the edge of the network, *edge computing* refers to computation on a variety of systems that are closer to end users [ETSI, 2015]. It is expected that the concepts of fog and edge computing will open up new possibilities for the future of internet and smart living environment.

This work is supported in part by the National Basic Research Program of China (973 Program, No. 2015CB352403), the NSFC (No. 61502302, 61472260, and 61402302), The Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges under Beijing Municipality (No. IDHT 20150507).

Author's addresses: C. Li, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China; Y. Xue, Department of Electrical and Computer Engineering, National University of Singapore, Singapore; J. Wang and W. Zhang, College of Information Engineering, Capital Normal University, Beijing, China; T. Li, Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA.

https://doi.org/10.1145/3154815

The emergence of edge-oriented computing paradigms such as *fog/edge computing* is of no coincidence. Existing *cloud computing* technology has altered our ways of utilizing computing services in the last decade. It thrives upon the massive fusion of computation and storage capabilities. However, as much as cloud architecture excels in driving forward our ease of using computing service, we also confront many obstacles that come along. The bandwidth limit, global centralization of data processing and storage have brought many issues such as undesirable latencies, less mobility support, inability of context awareness and data transmission overhead. This is especially true for smart sensing, content delivery, augmented reality, online gaming, and many other applications that require real-time analysis and control. With people's explosive demands and growing reliance on quick information access and efficient data processing, edge-oriented computing paradigms have become an evolving necessity for the realm of IoT (Internet of Things).

Despite its popularity, putting edge-oriented computing into practice is still challenging [Shi et al. 2016]. Both fog and edge computing systems are only on the verge of maturity. In recent years, researchers continue to provide new and different perspectives on how these systems should be built. It is without doubt that more efforts are needed to realize the vision of *fog/edge computing*. A thorough survey on the basic system architectures and management strategies is therefore of heightening demand for both academics and practitioners.

1.1 Related Surveys

Several integrative reviews have concisely summarized the key features of *fog computing*. Vaquero and Rodero-Merino [2014] tried to define fog computing by discussing a set of technologies that drive it. Yi et al. [2015a] briefly examined *fog computing* platform and pointed out representative applications that are well suited for computing near the network edge. Chiang and Zhang [2016] compared cloud and fog in detail and showed several case studies of *fog computing*. Kitanov et al. [2016] introduced the emerging fifth-generation (5G) network and its potential impact on fog. While the above reviews present a good overall picture of the fog, they only cover a limited number of papers and do not compare the methodology and technologies in detail. Recent years have witnessed a growing number of papers in this rapidly changing field. It is crucial to synthesize new findings from them.

There have been a few papers that tried to provide a fairly comprehensive survey of the *fog computing* paradigm. Yi et al. [2015b] identified various problem domains of *fog computing*, including networking complexity, quality of service (QoS), programming model, resource provisioning and etc. Mahmud and Buyya [2016] proposed a taxonomy of *fog computing* and gave a broad review on component types, general metrics and design objectives. Nevertheless, these works do not provide a classification of architecture and topology. Detailed explorations on resource management and system optimization approaches are absent as well.

A few recent studies have been concentrated on certain specific design issues of edge-oriented computing solutions. Wang et al. [2017] gave a detailed survey on mobile edge computing, with an emphasis on wireless communication and edge networking. Driven by economic and environmental concerns, Buzzi et al. [2017] provided a detailed survey on energy-aware designs for 5G networks. Recently, Osanaiye et al. [2017] discussed the security and privacy issue of fog computing in a virtualized environment. These prior works do not provide a detailed comparison of different architecture designs in the edge-oriented computing era. Their discussions are also limited to just one or two resource management strategies.

Another group of surveys have been focused on the security and privacy issue of fog. Stojmenovic and Wen [2014] examined fog computing with an emphasis on authentication and intrusion detection. Wang et al. [2015] explored potential issues in fog security and fog forensics. Recently, Roman et al. [2016] gave a through threat analysis on mobile cloud, mobile edge as well as fog. In light of this, we have excluded security-related topics from our discussion to reduce the scope of this survey.

This survey does not intend to provide an exhaustive list of studies covering all the fog/edge related challenges. We focus our attention on architecture design and system management with a rich and carefully selected set of papers to provide a very detailed landscape in this field. There have been prior surveys that summarize the architecture design and resource management of distributed systems [Orgerie et al. 2014] and mobile systems [Bellavista, 2012]. However, to the best of our knowledge, *fog/edge computing* has not yet been surveyed in detail from such a perspective.

1.2 Our Contribution

The main contributions of this work can be summarized as follows.

— We characterize fog and edge computing by comparing a complete list of closely-related concepts. We highlight the major differences among existing fog systems through a set of key dimensions that best summarize prior works.

— We present a detailed taxonomy of the architecture in the fog and edge computing paradigms. Specifically, we discuss the layered architecture of the cloud-fog-edge ecosystem from the physical as well as the logical view.

— We detail the state-of-the-art resource management approaches that envelop three different computing layers including back-end cloud infrastructure, near-edge fog system, and front-end edge devices.

— We classify research works based on key design objectives including latency, energy, availability and scalability. We describe major system optimization solutions and the heuristics used by them.

The remainder of this survey is organized as follows. Section 2 provides a thorough review of existing computing paradigms. Section 3 classifies the architecture of *fog computing* systems and compares representative fog components. Section 4 introduces our taxonomy of resource management approaches. Section 5 discusses major design objectives of *fog computing* systems. Section 6 outlines open problems in this field and finally Section 7 concludes this article.

2. ARGUMENTS ON DEFINITIONS

Perspectives aside, it is logical for the community to have various definitions on the emerging idea until the techniques grow mature, before which all contributions carry reasonable value to be discussed. This section provides a clear glance of ten closely related computing paradigms/concepts.

2.1 Edge-Oriented Computing: An Overview

In recent years, edge-oriented computing has emerged along with the dashing deployment and application of Internet of Things (IoT). Figure 1 depicts a reference model of computing systems organized as a five-layered structure. As a compliment piece to *cloud computing*, (*mobile*) *edge computing* [PNNL, 2013; ESTI, 2014] builds itself from the enormous scattered edge systems. Since these systems are of proximity to event instances and data sources, *edge computing* has its advantages of

shorter response time, better geographical awareness, more isolation from the outside world, optimized content distribution and etc.

As much as edge-oriented computing solutions aid *cloud computing* architectures in terms of executing geography or latency sensitive applications, it is not an add-on patch to *cloud computing*. Oftentimes, edge-oriented computing marks itself for on-spot data processing exploiting local devices, but neither is it just a simple blind data migration or computation offloading from the cloud. For example, *fog computing* [CISCO, 2015] was conceived to address four types of applications and services that do not fit the traditional *cloud computing* architecture very well: 1) latency-sensitive applications, 2) geo-distributed applications, 3) fast mobile applications, and 4) distributed control systems [Bonomi et al. 2015].

The concept of *fog computing* has great similarity to *edge computing*. Both of the paradigms construct themselves on the edges of the network near data sources. Currently there is no universally accepted open standard on what defines the "network edge". In fact, another concept, which is termed mist computing, pushes data processing even further to the extreme edge of the network [Cisco, 2015].

In general, *fog computing* can be treated as a special form of edge computing. A major difference is that *fog computing* pushes elastic IT services further to the end user devices [Manzalini and Crespi, 2016]. In addition, devices within the fog are often coordinated globally in a software-defined manner [Vaquero and Rodero-Merino, 2014]. In most scenarios identifiable, *fog computing* is often used when the task is service-oriented while *edge computing* pops up more if it is on an analytical ground. Prior works [Chang et al. 2014; Bonomi et al. 2014; Orsini et al. 2015] where both terms were touched upon suggest an interchangeable nature of them. This paper would therefore survey the related papers on both phrases, and summarize them as equals to cover the bigger picture.

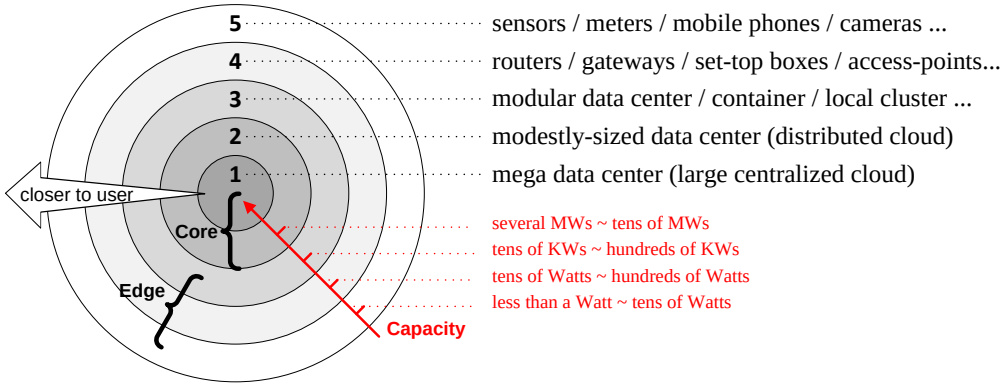

Fig. 1. From the core cloud to the edge devices: computing systems of different scales.

2.2 Synonyms and Related Concepts

The idea of computing near the edge is not new. It has been used for file sharing in early days. A well-known example is *peer-to-peer* (P2P) system [Androutsellis-Theotokis et al. 2004] which is a distributed application architecture that partitions tasks or workloads among peers. *EdgeComputing* [Davis et al. 2004] is a term first coined by Akamai, describing the architecture of content delivery network (CDN). *EdgeComputing* delivers content from cache servers located at the network edge,

closer to customers. Doing so ensures faster download time and better response time for interactive tasks, particularly when the cache servers are well-chosen.

In the meantime, *mobile grid computing* [Guan et al. 2005] mainly focused on the provision of grid services for mobile applications. Grid services can save device energy and storage space, hence reducing the size, weight, and cost of mobile devices. As mobile device users continue to grow, *mobile cloud computing* [Dinh et al. 2013] starts to attract increasing attention. It shifts end-user applications to a new era by jointly providing elastic storage and computation resources that augment the services. Both *mobile grid computing* and *mobile cloud computing* can be traced to the concept of "*cyber foraging*" [Satyanarayanan, 2001]. The purpose of *cyber foraging* is to augment wireless mobile system on demand by exploiting various resource-rich hardware infrastructures through ubiquitous networking.

There have been recent proposals trying to leverage human elements in addition to physical and digital artifacts. For example, *mobile crowd computing* [Fernando et al. 2016] is a special form of *mobile cloud computing* that also treat mobile device users as a kind of resource. Similarly, *edge-centric computing* [Lopez et al. 2015] intends to bring data and computation to the network edge, with a focus on human-driven applications.

Another related concept is *in-situ data processing*. [Stanley-Marbell, 2011; Li et al. 2015]. It is proposed for pre-processing of data in the field. It provides local data analytics in environmentally sensitive areas, or remote places that lack established utility infrastructure. Differently, the idea of *in-situ analysis* used in the scientific computing community and data management community normally refers to bringing computing to the data (in the memory or disk). This concept, also known as *near-data processing*, allows for reducing I/O overhead for compute-intensive workloads [Zhang et al. 2012; Lakshminarasimhan et al. 2013; Babarinsa and Idreos, 2016].

2.3 A Comparison of Edge-Oriented Computing Paradigms/Models

We compare different concepts by presenting fourteen design features. As shown in Table I, the fourteen features are divided in to three categories: 1) underlying infrastructures that look at the required types of hardware, 2) operation mechanisms that are related with the management and control of the platform, and 3) service types that highlight the results and performance of the applications.

2.3.1. Infrastructure. EdgeComputing [Davis et al. 2004] processes are designed to run on Akamai's virtualized edge server environment. Load balancing is required to prevent overloading of any particular computer or application. Limited computing resource is a major issue faced by *mobile edge computing* [Orsini et al. 2015], *mobile crowds computing* [Dinh et al. 2013], and *mist computing* [Martin et al. 2015]. A common denominator of these concepts is that they rely on endpoint devices that have limited computing capability. For *fog computing* and *in-situ processing* [Li et al. 2015], resource-rich server clusters might be used to process data locally. Like *mobile cloud computing*, *edge-centric computing* [Lopez et al. 2015] uses a hybrid system architecture that combines cloud resources with mobile terminals.

Energy and power constraints are not big concerns for some early proposals (e.g., *peer-to-peer computing* [Androutsellis-Theotokis et al. 2004] and *mobile grid computing* [Guan et al. 2005]) utilizing stable resources. Since mobile phones and sensors typically have battery life issues, power management is crucial for computing paradigms such as *mobile crowds computing* [Dinh et al. 2013] and *mobile edge computing* [Orsini et al. 2015]. Differently, *in-situ processing* systems [Li et al. 2015] incur a power availability issue as intermittent energy sources are used.

Table I. Comparison of Different Concepts for Data Processing Near the Edge

Terminology \ Features	Underlying Infrastructure					Operation Mechanism					Service Type			
	Rich in compute and/or storage resource?	Rich in energy and/or power resource?	Computation is mainly at the network edge?	Data storage is mainly at the network edge?	Exhibit continuous system availability?	Interact with remote infrastructure (cloud)?	Control logic is mainly deployed at the edge?	Context (location, activity, etc.) awareness?	Human factor (intelligence) involved?	Well-designed for user mobility?	Support real-time control and interactive service?	Support throughput applications?	Augment cloud data center service?	Augment mobile device performance?
Peer-to-Peer Computing [Androutsellis-Theotokis et al. 2004]	●	●	●	●	●	○	●	○	○	○	○	●	○	○
EdgeComputing [Davis et al. 2004]	●	●	●	●	●	●	●	●	○	○	●	?	●	○
Mobile Grid Computing [Guan et al. 2005]	●	●	●	●	●	○	●	○	○	●	●	○	○	●
Mobile Cloud Computing [Dinh et al. 2013]	●	●	○	●	?	●	●	○	○	●	●	○	○	●
(Mobile) Edge Computing [Orsini et al. 2015]	●	○	●	●	?	○	●	●	○	●	●	○	●	●
(Mobile) Fog Computing [Bonomi et al. 2012]	●	●	●	●	?	○	●	●	○	●	●	○	●	●
Edge-Centric Computing [Lopez et al. 2015]	●	○	●	●	?	●	●	●	●	●	●	●	●	●
In-Situ Processing [Li et al. 2015]	●	○	●	●	○	○	●	●	○	●	○	●	●	●
Mist Computing [Martin et al. 2015]	○	○	●	●	○	○	●	●	○	●	●	○	○	●
Mobile Crowds Computing [Fernando et al. 2016]	○	○	●	●	○	○	●	●	●	●	●	○	●	●

● – Yes ○ – No ? – Not mentioned

Almost all the above concepts focus on local data processing, with the exception of *mobile cloud computing*. It emphasizes computation offloading, e.g., moving certain part of code/states to the cloud. In Table I, the location of data files vary greatly for different computing paradigms, but still at the edge. For example, the data can be spread over local PCs, cache servers, grid computers, and nearby mobile devices.

The availability of underlying resources can be affected by different factors. For example, *mobile cloud computing* requires stable connection to specific cloud services, while *in-situ processing* requires sufficient renewable power budget. Managing frequent node disconnections and randomly joined new devices must be considered by *mobile crowd computing* [Fernando et al. 2016]. In a *peer-to-peer computing* environment, node dynamics (failure or departure) may cause service disruptions to the downstream peers.

2.3.2. Operation Feature. It is not unusual for edge-oriented designs to interact with a central platform such as the cloud. This can be seen in *EdgeComputing* [Davis et al. 2004], *mobile cloud computing* [Dinh et al. 2013], as well as *edge-centric computing* [Lopez et al. 2015]. Oftentimes, the cloud is positioned to play a significant role in the ingestion and processing of ubiquitous data. Other than relying on the cloud data center for system coordination, all the surveyed computing paradigms choose to enable certain degree of autonomy by placing the major control logic on edge machines. As a result, edge computing systems can assign computation, perform synchronization, control storage, or adapt to local environment responsively.

Context-awareness is a key advantage of many edge-oriented computing systems. Human factors (social interaction, personal activity, etc.) and physical environment (location, resource availability, etc.) are two major aspects of the context. Context-aware computers could both sense and react based on their current environment. For example, the cache server of *EdgeComputing* can be used to enable personalization or customization of web services. Both *mobile edge computing* and *mobile fog computing* are able to infer their own location and make informed decisions based on the user mobility. To take advantage of renewable energy, *in-situ processing* systems consider the variability of local power budget. One thing that distinguishes *edge-centric computing* and *mobile crowds computing* from other concepts is the emphasis on human elements. Both proposals leverage human intelligence to solve issues (e.g. qualitative classification, human validation) that involve human-beavering sensing.

2.3.3. Service Type. Two representative types of application that may benefit from near-data processing: interactive jobs and throughput jobs. Faced with constant influx of medium-sized data created by multiple machines, many edge-oriented computing solutions support real-time analysis and control. There have been prior proposals that concentrate on optimizing throughput tasks. For example, both *peer-to-peer computing* and *in-situ processing* aim to maximize the throughput of data-driven streaming/processing.

From the perspective of service augmentation, existing proposals can be roughly grouped into 1) complementing cloud service, and 2) augmenting mobile service. For example, *peer-to-peer computing* and *EdgeComputing* are not designed to enhance the user experience of mobile users. Alleviating data center burden is not a major intention of *mobile grid computing*, *mobile cloud computing*, and *mist computing*.

2.4 A Comparison of Existing System Units for Edge-Oriented Computing

Over the years, various modular computing systems have been proposed along with different edge-oriented computing models. These systems are the basic building block for edge computing. Table II summarizes the main features of some representative system model and outlines their differences.

2.4.1 Hardware Components. Prior works have exploited three types of hardware resources for computing on the edge: end devices, smart gateways, and local servers. At the extreme-edge of the network, mobile devices and sensor networks are able to work together to form a device cloud [Shi et al. 2015]. It is also feasible to tap into smart gateways (access point, set-top boxes) that have certain computing/storage capability [Su and Flinn et al. 2005; Savolainen et al. 2013, Alam et al. 2016]. As computer hardware continues to become more affordable, dedicated computers and servers would be placed locally to augment edge devices that have severe resource constraints [Satya et al. 2013; Li et al. 2015].

2.4.2 Virtualized Environment. Like the cloud data center, fog computing nodes typically require virtualized environment to enable their functionalities. The virtualization approach varies from system to system. Hardware virtualization has been used to support multi-tenancy, elastic scaling, and tight resource management [Satya et al. 2009, Li et al. 2015]. Oftentimes, wireless sensor network (WSN) virtualization is the crux of building a collaborative and efficient sensing application [Abdelwahab et al. 2015]. In some cases, sandbox mechanisms can be used to provide a highly controlled environment. For example, Savolainen et al. [2013] uses JavaScript sandbox to manage downloaded programs and separate running services.

Table II. Summary of System Units Proposed for Data Processing Near the Edge

System Units \ Categories	**Underlying Hardware:** - D: Sensors, mobile devices - G: gateway system - S: servers	**Virtualized Environment:** - V: HW virtualization - L: LXC / Docker - O: other methods	**Typical Scale:** - S: local / neighborhood - L: community area	**Compute Resource:** - H: Heterogeneous - O: Homogeneous	**Operation Mode:** - S: standalone (w/o cloud) - C: cloud-assisted - O: cooperative (with others)	**Workload Type:** - S: Time-series / Stream - B: Bulk data / batch	**Role / Function:** - D: data caching / delivery - P: processing / analytics - I: integration / management	**Usage Scenario:** - G: generic / general - S: specific / specialized	**Evaluation:** - N: numerical analysis - S: event-driven simulation - P: prototyping / Measuring
Edge Cache (Grid) [Ramaswamy, et al. 2007]	S	-	L	O	CO	SB	D	S	S
Nano Data Center [Laoutaris et al. 2008]	G	-	S	O	CO	B	D	S	P
Cloudlet [Satya et al. 2009]	S	V	S	O	SC	SB	P	G	P
Spaceify Node [Savolainen et al. 2013]	G	O	S	H	C	S	PI	G	P
Edge Node [Chang et al. 2014]	S	L	L	-	C	S	DP	G	P
Fog Node [Dsouza et al. 2014]	-	-	S	H	S	T	P	G	P
Device Cloud [Shi et al. 2015]	D	-	S	H	SO	S	P	G	P
Fog Computer [Dubey et al. 2015]	G	-	S	O	C	S	P	S	P
In-Situ Server System [Li et al, 2015]	S	V	S	O	S	SB	P	S	P
Mobile Micro-Cloud [Wang et al. 2015a]	G	-	S	H	SC	S	P	G	NS
Mobile Fog [Alam et al. 2016]	G	V	L	H	CO	S	P	G	S
Mobile Edge-Clouds [Fernando et al. 2016]	M	-	L	H	SC	S	P	G	P

Many recent works chose light-weight OS-level virtualization. Chang et al. [2014] adopted Linux container (LXC) to run services on edge systems which have limited storage and processing capabilities. ParaDrop [Willis et al. 2014] is a framework that enables developers to design virtually isolated compute containers on various gateways. Unlike virtual machines, multiple containers share the same Linux kernel and therefore exhibit very small overhead. Emerging solutions like Docker allow dynamic packaging of an application and all its dependencies in a software container. Ismail et al. [2015] introduced Docker container based edge computing and qualitatively evaluated its performance. Pahl et al. [2014; 2015] have demonstrated the potential of Docker in improving the efficiency and interoperability of platform-as-a-service (PaaS) on distributed edge clouds. Saurez et al. [2016] stored binaries for their fog application components on a Docker registry server and run task process in a Docker container. Yangui et al. [2016] leveraged containers to facilitate the test, deployment, and execution of their Cloud/Fog hybrid services. Machen et al. [2016] considered live migration using container in the edge computing environment.

2.4.3 System Scale. Depending on the services provided and the system topology, fog systems scale from sensors that cover a person or neighborhood area to a regional platform that is responsible for a large community. Most of the fog nodes are localized, such as the Cloudlet for mobile services [Satya et al. 2009], In-Situ Server Systems for bulk data analytics in the field [Li et al. 2015] and Fog Computer for smart body sensing [Dubey et al. 2015]. In some cases, a Combine Cloud/Fog system such as Mobile Edge-Clouds [Fernando et al. 2016] may cover a broader area.

2.4.4 Resource Type. Fog computing nodes may come in different form factors, and they can be deployed in various environments. Consequently, many dynamically-established edge computing systems such as the Device Cloud [Shi et al. 2015] are

normally a pool of heterogeneous resources. Another example is the architecture of Mobile Fog [Alam et al. 2016], which is considered a mix of access point (AP) and the access point controller (APC) units. Note that fixed devices or stationary resources may manifest themselves as homogeneous (or less heterogeneous) systems. For example, Edge Cache [Ramaswamy et al. 2007] servers are standard system implemented using pre-fabricated machines; Fog Computer [Dubey et al. 2015] and In-Situ Server System [Li et al. 2015] are small or medium scale systems designed for preprocessing near the data acquisition site. There is not a strong motivation for integrating heterogeneous hardware resource in them.

2.4.5 Operation Mode. Some cloud surrogate can provide continuous service even if the cloud is temporarily unavailable. Since a Cloudlet only contains soft state (cached data/code) that is available elsewhere, it is highly resilient. In addition, the synthesis process of Cloudlet is independent of the cloud datacenter, and therefore it can serve mobile devices in case of WAN failures. In other cases, such as Spaceify nodes, Internet connection is necessary. Sometimes, nodes just need to cooperate with each other to share data and improve performance [Ramaswamy, et al. 2007].

2.4.6 Workload Type. In the IoT era, time-series data is an important type of workload. Sensors dynamically collect data and send them to the cloud via various fog systems [Dubey et al. 2015]. To ensure low-overhead, it is crucial to preprocess time-series data. In addition, bulk files or batch jobs can be seen in the fog era. For example, InSURE [Li et al. 2015] is able to pre-process bulk data (Hadoop jobs) onsite.

2.4.7 Major Function. The task performed by a fog system generally falls into three roughly divided categories. The first is to provide application caching and content delivery. Representative examples include Edge Cache [Ramaswamy, et al. 2007] and Nano Data Center [Laoutaris et al. 2008]. The second is to provide a computational presence in the proximity of the end-user. In this case, the fog systems can be leveraged to augment nearby devices (e.g., Cloudlet) or preprocess data to reduce data movement overhead (e.g., In-Situ Servers). Finally, in some cases, the fog computing platform is mainly designed to efficiently integrate and coordinate ubiquitous devices [Savolainen et al. 2013; Dsouza et al. 2014].

2.4.8 Usage Scenario. Researchers have explored building both a general model for fog computing or a specialized fog system. Systems such as Edge Node and Cloudlet provide a more generic interface for users. In contrast, several system designs are proposed to solve a specific usage scenario. For example, Nano Data Center [Laoutaris et al. 2008], is primarily designed for facilitating content delivery on the edge. They lack the capability of managing various IoT devices. Combining local computing systems with standalone renewable energy systems, in-situ servers mainly focusing on processing raw data generated in the remote area.

2.4.9 Evaluation Approach. Both real systems and simulators are indispensable evaluation platforms for fog studies. Most of the prior works demonstrate their design via scaled-down prototype or real measurement on a testbed. Trace-based simulation has been used to simulate various techniques for edge systems that are economically impractical to build in a lab [Ramaswamy, et al. 2007]. In some cases, numerical analysis and event-driven simulation are used to validate the effectiveness of optimization algorithms [Wang et al. 2015a, Wang et al. 2015b].

3. ARCHITECTURAL EXPLORATION

Research on edge-oriented computing is still at its beginning. Currently there is no universally accepted open standard available. Most of the prior articles refer to edge computing as a platform with layered architecture. Based on the "4+1 Architectural View Model" [Kruchten, 1995], we classify existing proposals into two groups: physical architecture and logical architecture.

3.1 Physical View of the Hierarchical Design

The physical architecture depicts edge computing infrastructure mainly from a system engineer's point view. It is concerned with the topology of edge components.

3.1.1 Three-Tier Design. It is typical to treat fog as an additional layer between the core cloud and the edge devices today. Such a Cloud-Fog-Edge three-tiered architecture has been recognized in many prior works [Stojmenovic and Wen, 2014; Papageorgiou et al. 2015; Dastjerdi and Buyya, 2016]. In the remainder of this article, we will use this topology when discussing resource management approaches.

The fog layer offers great opportunity for alleviating edge devices from severe resource constraints. For example, Cloudlet [Satyanarayanan et al. 2009] is a classic fog system sitting between remote cloud and mobile devices. Cloudlets are widely dispersed, discoverable cloud surrogates that are one hop away from mobile devices. It usually resembles a cluster of virtualized high-performance computers with high-bandwidth network. In contrast to Cloudlet, Spaceify [Savolainen et al. 2013] focuses on edge services enabled by lightweight web technologies. The concept of Spaceify aims at digitalizing a richer portfolio of computing resources that may be available in the physical space. In its client-edge-server model, the edge node is generally a smart gateway or access point that integrates devices in the physical space. Importantly, the edge node is also able to offer its own computing and storage resources to support users in proximity. Similarly, NECtar [Papageorgiou et al. 2015] also treats smart gateway as the key bridge between the core cloud and the fog. NECtar agents run on a gateway close to the data source. Data are collected at the gateway and propagated through the network core to the backend system (e.g., cloud database). The gateway provides computing capabilities that can be leveraged to preprocess raw data.

By adding a fog layer, one can greatly improve the scalability of cloud. Xu et al. [2011] explored the scalability issue of cloud-fog system in smart cities. Cloud data centers can be easily overwhelmed by massive numbers of sensors that request services. A three-tier architecture called CEB (cloud, edge, and beneath) is proposed to manage the scale of the cloud-sensor system. Cloud of Things [Abdelwahab et al. 2015a] is another highly scalable architecture for sensing as a service (SaaS). It consists of three main elements: IoT devices, first-tier clouds, and cloud agents. First-tier clouds are conventional cloud data centers and cloud agents are trusted and resource-rich components near the network edge. Depending on the types of devices they serve, cloud agents can be either powerful servers or flexible smartphones.

The fog components are often characterized by their non-functional properties such as real-time behavior and availability. Many fog services have been designed with the non-functional properties in mind [Steiner et al. 2016]. For example, real-time behavior is especially important in the Industrial Internet of Things (IIOT) domain, in which information exchange must be fulfilled within milliseconds or even microseconds. To ensure the quality of service in the telehealth domain, Dubey et al. [2015] proposed a three-layer fog-based sensing system that is composed of a body sensor network (BSN) for data acquisition, a fog gateway computer for onsite

analytics, and a cloud platform for back-end support. Its fog gateway computer can be implemented using low-power microprocessors such as Intel Edison.

As an intermediate system layer between cloud resources and edge devices, fog components also bring cloud more context awareness [Aazam and Huh, 2015]. With localized fog system, it becomes much easier to capture some crucial context information like resource availability, mobility pattern, intermittent connectivity, etc. [Abdelwahab et al. 2015a]. This information allows one to make informed decisions towards a more adaptive system [Orsini et al. 2015].

3.1.2 Four-Tier Design. In the four-tiered architecture, the fog is generally divided into two sub-layers based on servers' capacity and vicinity to users. For instance, Bonomi et al. [2012] first gave a high-level architectural view of the fog distributed infrastructure for IoT. This work describes the fog computing system mainly from a network perspective. The four architecture layers are cloud data centers, network core, network edge, and embedded systems (endpoints).

Tang et al. [2015] highlighted a 4-layer fog computing architecture in emerging smart cities. The proposed architecture places the cloud data center at the top (Layer 1) and treat fog as a three-layer system with end user devices included. Layer 2 consists of a number of intermediate computing nodes that target community-level services. Layer 3 has a group of edge computing nodes for neighborhood-wide area. Layer 4 at the bottom is sensing networks on critical infrastructures.

Chang et al. [2014] introduced the architecture of Edge Cloud with edge nodes as basic building blocks. Edge nodes interact with mobile devices and provide low-latency, bandwidth-efficient, and resilient services to end users. Edge nodes in the same network are grouped together to form an Edge Zone. The Edge Cloud is the federation of cloud data centers with edge zones.

In a recent work by Souza et al. [2016], researchers have investigated the architecture of Combined Fog-Cloud (CFC). This topology model also vertically distributes computing servers in four layers. Between the top (cloud) and the bottom (end-user devices) layers are two fog layers. The first fog layer is composed of low-latency nodes that are one hop (wireless access connection) away. In the second fog layer, computing nodes of medium capacity and latency are deployed to provide data aggregation service in a neighborhood area network (NAN).

3.2 Logical View of the Hierarchical Design

The logical view is concerned with the functionality that edge computing would provide to its users. In other words, the logical architecture describes the functional requirements taken from the problem domain.

3.2.1 Three-Tier Architecture. Bonomi et al. [2014] described three key factors for fog software architecture. The proposed design comprises the following layers: IoT/IoE verticals layer, fog orchestration layer, and fog abstraction layer. This architecture aims at coordinating heterogeneous resources that span core service, edge services, and endpoints. The orchestration layer supports data aggregation, decision making, service migration, etc. The abstraction layer is mainly designed to expose a uniform interface to the clients and hide the heterogeneity of the platform. Dsouza et al. [2014] extended this logical architecture by adding a policy management module at the orchestration layer to enhance the capability of security management.

Xu et al. [2014] describes a service-oriented architecture that can automate the integration of sensors. The system is mainly composed of a cloud middleware for cloud sensing service, an edge middleware for edge sensing service, and a sensor

platform for data acquisition. Servers in the edge layer are used to cache selected applications to slow down the growth of cloud dimension.

3.2.2 Many-Tier Architecture. Researchers have proposed various ways of sub-dividing an edge computing system into many smaller layers. Each layer is a collection of similar functions that provide data/services to the layer above it and received data/services from the layer below it. For example, a three-layered IoT architecture can be extended to be composed of five layers: *business*, *application*, *processing, transport*, and *perception* [Wu et al. 2012, and Khan et al, 2012].

Based on the basic data processing tasks that edge computing system can offer, Aazam and Huh [2015] showed a more elaborated 6-tier architecture. At the *physical and virtualization layer*, various physical/virtual devices are managed and maintained. At the *monitoring layer*, the system collects crucial statistics (performance, energy, etc.) of the underlying hardware. At the *preprocessing layer*, the fog refines data, analyzes data, and distills key knowledge. The obtained information is then stored in the *temporary storage layer*. Once the data has been delivered to the cloud and there is no need to keep a copy in local storage media, one can delete the data. Since edge devices frequently generate some private data, a *security layer* is added to handle sensitive data. Finally, the topmost *transport layer* is responsible for efficiently moving data to the cloud.

Another six-layered general reference edge computing architecture has been proposed by Dastjerdi and Buyya [2016]. The architecture mainly consists of the application layer, the programming layer, the service layer, the management layer, the resource layer, and the IoT sensor/actuator layer. This architecture is more focused on the key functions that are required for delivering services to users.

3.3 Architecture Level Optimization Strategies

There have been many recent proposals of optimized architectures. These architectures normally feature finely-divided hierarchies or functionalities. We classify these optimization strategies into four groups based upon the number of physical layers and logical layers they have (Figure 2). The physical layer refers to the implementation of the edge computing infrastructure. Multiple physical layers mean the system is composed of physically different structures. The logical layer refers to system function and operating modes of the hardware. Multiple logical layers mean that the edge computing node may exhibit different runtime status.

3.3.1 Single Physical Layer, Single Logical Layer (SPSL). Oftentimes these works use a traditional architecture with optimized software framework support. For example, Savolainen et al. [2013] described the open-source software architecture of their edge nodes, which are decomposed into three layers: the operating system layers (GNU/Linux), the application layer, and a JavaScript sandbox layer. Each node runs downloaded services called Spacelet, which allows external service providers to run software on demand on the edge nodes. Papageorgiou et al. [2015] introduced a streamification method for IoT data reduction at the network edge. The edge devices dynamically select appropriate data handlers as different IoT data sources require different data reduction techniques. Yangui et al. [2016] proposed a software framework based on the extension of existing platform-as-a-service (PaaS) to automate application provisioning in a hybrid cloud/fog environment. Manzalini and Crespi [2016] introduced the edge operating system which integrates software defined network (SDN) and network function virtualization (NFV). Liang et al. [2017] discussed a hierarchical control strategy that combines SDN controller and local

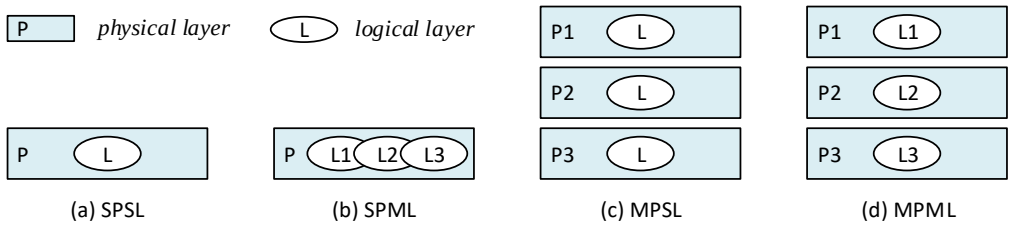

Fig. 2. Diagrams comparing different types of architecture-level exploration

controllers. The SDN controller manages the whole network and local controllers responsible for handling latency-sensitive requests.

3.3.2 Single Physical Layer, Multiple Logical Layers (SPML). This type of optimization aims to enhance the efficiency of a specific layer in the edge-oriented computing systems. For example, Willis et al. [2014] proposed to dynamically leverage gateways in a multi-tenant fashion. They propose to share CPU cycles, unused memory capacity and extra disk space from the gateway devices for fog acceleration. The pull and push modes of edge systems have been investigated for better energy efficiency [Xu et al. 2011]. Abdelwahab et al [2016] designed a pull-based memory replication protocol for minimizing cloud-edge communication overhead. Li et al. [2016] leverages different system and performance states of servers to improve the energy flow in an in-situ server cluster.

3.3.3 Multiple Physical Layers, Single Logical Layers (MPSL). Instead of severing users using a flat collection of fog servers, Tong et al. [2016] further proposed to deploy hierarchical edge cloud. In other words, the fog layer is composed of a tree hierarchy of server nodes. The system tends to assign tasks to lower tiers of servers first. During peak hours, it opportunistically place loads at higher tiers in the hierarchy. With appropriate task placement, this architecture is shown to be effective in improving resource utilization and reduce execution delay. This type of architecture is an extension of the four-tier edge node [Chang et al, 2014; Tang et al. 2015]. The major difference is that Tong et al. [2016] maintained one uniform fog layer with finely-divided hierarchies.

3.3.4 Multiple Physical Layers, Multiple Logical Layers (MPML). There have been several works exploring heterogeneous hierarchical fog nodes. Giang et al [2015] proposed three generic abstractions for fog nodes: 1) edge nodes that produces data, 2) IO nodes that broker communications with edge nodes, and 3) compute nodes that are responsible for processing. Different from IO and compute nodes, the edge nodes are typically not reconfigurable and programmable. A distributed dataflow programming model is proposed to ease the development process for such three-layered fog environment. Similarly, volunteer nodes can choose to participate as a data node or a compute node [Ryden et al. 2014]. Faruque et al. [2016] further spatially divided edge devices into multiple zones. By dividing sensor devices into access-point, base station, and end devices, one can increase the range of the sensor network.

4. MANAGEMENT APPROACHES

This section presents a detailed taxonomy of fog computing with respect to three different resource management levels. As shown in Figure 3, they are: *back-end*

cloud infrastructure, *edge systems*, and *front-end smart devices*. The edge system includes both server-based edge and local edge systems.

Back-end infrastructure can be either a few centralized large-scale cloud data centers [Chandramouli et al. 2012] or many geographically distributed cloud data centers [Fricker et al. 2016]. A common resource management challenge is to smartly distribute tasks of varying characteristics in the Cloud-Edge environment.

On-premise computing systems play a key role in the post-cloud era as well. They are local resource-rich facilities that could be leveraged to support data- and compute- intensive tasks created by ubiquitous edge devices. These systems are deemed as local surrogates that operate on behalf of the remote back-end systems.

Finally, as much as the edge devices are considered as enormous and ubiquitous, edge computing is only worth implementing with a legitimate resource allocation method, which would otherwise turn data processing into chaos. Unlike cloud centers, edge devices are more diversified and therefore more complicated.

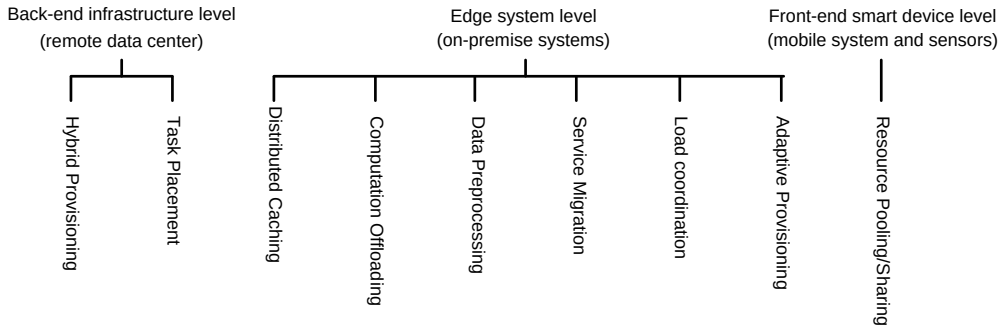

Fig. 3. A taxonomy of edge-oriented resource management approaches

4.1 Hybrid Provisioning

Hybrid provisioning recognizes that centralized cloud resources are unsuited for processing ubiquitous data due to inefficient and expensive data movement. Hybrid Provisioning provides a unified view of cloud resource and edge resource. With hybrid provisioning, cloud and edge resources can complement and augment each other.

An important attempt is to enable leverage distributed systems sitting at the network edge to enhance the performance of cloud infrastructure. Chandra et al. [2009] describes several classes of services for which cloud computing may not fit well. Dispersed-data-intensive services, experimental cloud service, as well as non-commercial shared services all have problems with the pay-as-you-go cloud model. The authors proposed to enrich centralized cloud with volunteer dispersed computing resources to satisfy the requirements of such services. Sundarrajan et al. [2011] devised a new model called Nebula, which features dispersed computing resources, context-aware scheduling, and cost-effective management. An enhanced version of Nebula [Ryden et al. 2014] further supports data-intensive computing framework (e.g., MapReduce) on edge resources.

There have been other projects that exploit the efficiency presented by hybrid resource provisioning to enhance end user experiences. Cloud4Home [Kannan et al. 2011] aims at providing seamless data storage services on local/private and remote/public systems. It uses a distributed key-value store to build a uniform interface for data access, resource monitoring, and request routing. Personal Cloud (PCloud) [Jang et al. 2013] focuses on gracefully combines remote cloud resources

and local machines to improve mobile device capabilities. It gracefully maintains a pool of network-reachable resources for each end user.

To enable efficient execution of data-intensive tasks on such a cloud-edge hybrid system, resource management decisions must take into account job characteristics and system dynamics. For example, Cloud4Home queries node resource information and determines the most suitable node that supports a request. The coordinator of PCloud rates each candidate resource according to the network connectivity conditions and a specific metric called performance index (PI). Nebula uses a scoring function to rank data nodes based on network bandwidth, service latency or proximity to the requesting user to reduce performance overhead.

4.2 Task Placement

Recent years have witness a shift in resource management focus from central cloud data center to distributed cloud systems [Hao et al. 2014]. Importantly, there is a growing interest in resource selection and service allocation problem considering different properties of in-cloud (i.e., within central/distributed cloud data center) and in-situ (i.e., near the dispersed data source) resources.

Several studies aim to find a desirable design tradeoff in the so called Cloud-Edge topology [Chandramouli et al. 2012], Combined Fog-Cloud (CFC) system [Souza et al. 2016], or Fog-to-Cloud (F2C) environment [Masip-Bruin et al. 2016]. The basic observation is that in-cloud compute nodes typically have the capacity and capability of high-performance data processing, whereas in-situ compute nodes exhibit less network communication cost and better responsiveness to local request.

Chandramouli et al. [2012] considered cloud as an important communication hub for massive edge systems and focused on a class of applications that need to join/correlate edge data in real time. This work uses a cost-based optimizer to find the optimal task placement such that the global communication cost can be minimized. Souza et al. [2016] classified edge computing services into two distinct types: 1) a big amount of light-weight services and 2) a small number of resource-hungry services. In this work, resources are represented using slots and the latency due to edge device interaction is considered. To optimize the Quality of Service (QoS), the authors formulated the service allocation as an ILP problem. Differently, Deng et al. [2016] proposed to solve the workload allocation problem through convex optimization techniques. Specifically, the authors decomposed the problem into three sub-problems: a convex optimization problem with linear constraints (power-delay tradeoff for fog), a mixed-integer nonlinear programming problem (power-delay tradeoff for cloud), and a job assignment problem that can be solved in polynomial time (communication latency minimization for traffic dispatch). Kang et al. [2017] showed that cloud-edge hybrid systems for machine learning must consider the load variation of each layer within a deep neural networks (DNNs) workload. The authors propose a fine-grained layer-level scheduling mechanism that can improve end-to-end latency while greatly saving mobile energy consumption.

In addition to the cloud-fog co-optimization, load placement among distributed micro data centers or has been explored. In the context of fog computing, some small-scale data centers could be easily overwhelmed if not scheduled appropriately. Steiner et al. [2012] showed how a network-aware data center scheduling can improve service for edge applications (e.g., virtual desktop). Their algorithm uses the live measurements of the network conditions and data center load to calculate an overall score for each data center. The score can be used to guide service placement and migration. Other than optimizing load placement using simple heuristics,

Fricker et al. [2016] qualitatively evaluated the gain achieved by the collaboration of data centers in the context of fog computing. This paper models a simple scenario: when one request cannot be accommodated by a data center, it may be forwarded to another one that has adequate capacity.

4.3 Distributed Caching

Edge caching grows out of the concept of content delivery networks (CDNs). Caching often-requested data on the logical edges of the Internet is an important way to better deliver dynamic Web content [Davis et al. 2004].

Considering the trends towards more cost-efficient server hardware, companies are becoming willing to deploy computer resources at the edge level. Caching user content from remote cloud to a collection of edge nodes near the users help alleviate the burden of today's over-subscribed data center. In addition, it can minimize communication overhead across the wide area network and reduces service latency. Gao et al. [2003] showed that smart data replication at edge server system could provide 5X performance improvement over conventional centralized architecture. Lin et al. [2007] aimed to provide full consistency for edge content and proposed a replica control protocols that execute all the transactions locally. Malandrino et al. [2016] studied various caching schemes from core-level switches to individual base stations. Using realistic large-scale data traces, they show that vehicular networks make a strong case for the adoption of mobile-edge caching.

In addition to the performance issue, Xu et al. [2014] further looked at cloud scalability in the smart city era. Caching data at the distributed edge servers contributes to the growth slowdown of cloud, thereby alleviating the scalability issue. They proposed an Application Fragment Caching Algorithm which selectively puts cacheable parts of an application on to the edge server according to the anticipated benefit of the caching activity. This mechanism aims at maximizing the reduction of the resource usage in cloud components.

As a tremendous amount of datasets are coming from end users, application caching becomes increasingly important. For example, in the fifth generation (5G) wireless networking era, Telco datacenters demands decentralized and flexible architectures where predictive resource management plays a crucial role. Caching strategic contents locally at base stations (BS) allows for higher user satisfaction and backhaul offloading. Zeydan et al [2016] investigated proactive content caching in 5G wireless networks. A general big data processing framework for analyzing users' data traffic is discussed. The purpose of this platform is to store users' data traffic and extract useful information for proactive caching decisions.

In the fifth generation (5G) era, cloud radio access network deploys a centralized baseband signal processing unit (BBU). BBU is responsible for the baseband processing of a set of remote radio heads (RRHs). RRHs are radio transceivers that have become the crux of distributed base stations today. In the fog era, RRHs are equipped with storage and signal processing functionalities to form enhanced RRHs. Park et al. [2016] studied fog radio access network as a cache-aided system. RRH can be equipped with storage and signal processing functionalities to form an enhanced RRH. Edge caching can be performed to prefetch the most frequently requested files to the eRRHs' local caches during off-peak traffic period. By coordinating processing at the BBU and at the eRRHs level, one can maximize the delivery rate while satisfying fronthaul capacity and RRH power constraints.

In recent years, another line of works is about cloud gaming, in which games are stored and run on the cloud. To ensure a desirable online gaming experience, the

transmission delay of the game videos should be no more than 100ms [Jarschel et al. 2011]. Lin and Shen [2015] proposed to build edge server (called supernode) to render game videos and stream them to nearby players. The supernode can adaptively adjust the encoding rate of different game videos based on their degrees of tolerance for lower quality. An improved version of the supernode design further considers the impact of social network [Lin and Shen, 2016]. The insight is that assigning social friends to the same supernode can lower data communication overhead.

To reduce transaction latency, Romano and Quaglia [2014] proposed to route transactional requests to the origin sites through multiple edge cache servers. A protocol has been devised to identify the fastest data delivery path through a self-exclusion process of less responsive edge cache servers. The proposed server selection scheme is lightweight since it does not require very complex coordination among geographically dispersed edge servers.

Cooperation among edge caches is a key technique that can be used to enhance the capabilities of edge cache networks serving highly dynamic Web content. If the requested document is not available locally, the edge server can first contact nearby nodes to retrieve the document rather than turn to remote servers. Ramaswamy et al. [2007] demonstrated that cache cooperation can provide significant benefits to edge cache networks. Some challenges include cache placement problems, how to setup and maintain (cooperate and build groups), how to handle dynamism and adaptivity, and manage utilization. The authors have incorporated several strategies for improving the efficiency and effectiveness of cooperation.

4.4 Computation Offloading

Offloading is the leverage of external resource-rich system for extending the capabilities of mobile systems. In the case of computation offloading, the end user application can be partitioned so that the most computation-intensive operations can be moved to a resource-rich cloud surrogate such as Cloudlet [Satyanarayanan et al. 2009]. Offloading is often preferable if the application requires high amount of computational processing and low amount of data communication. Most research efforts focus on offloading computation rather than data. For instance, CloneCloud leverages the great computation capability of the cloud to accelerate the execution of a selected portion of its mobile applications [Chun et al. 2011].

Flores et al. [2014] compared different mobile cloud computing models, namely, offloading and delegation. Computation offloading allows users to take advantage of external computing resources and storage capacity. Task delegation is a different concept which aims at enriching the functionality of the device instead of alleviating its component from the burden of performing resource-/energy- intensive operations.

Many previous works have investigated the offloading strategy in terms of where, when, and what to offload. Rudenko et al. [1998] demonstrated that a great amount of energy can be saved through computation offloading. Huertacanepa et al [2008] leveraged both historical execution data and current system status to perform adaptive offloading. Xian et al. [2007] proposed and efficient timeout scheme for computation offloading to improve the efficiency on mobile devices.

The performance of mobile offloading can be greatly affected by the wireless network connectivity [Barbera et al. 2013]. Hu et al [2016] quantified how cloudlets impact different metrics such as response time and energy consumption relative to cloud offload. The paper shows that offloading to a distant cloud can result in even lower performance and higher energy costs than running locally on the mobile device. For highly interactive applications, even offloading to nearby clouds can be

detrimental to performance. Even for offloading over LTE, edge computing continues to provide superior results. Their results also show that offloading computation blindly to the cloud can be a losing strategy.

Recent effort has focused on more complex computational offloading problem. For example, Oueis et al. [2015] did not treat the edge cloud as a pre-established system, but as a dynamically formed cluster. The composition of the cluster varies according to the service request. Orsini et al. [2015] further considered the context information of various mobile edge applications. With connectivity prediction, the proposed framework is able to foresee the effectiveness of offloading decisions. Instead of pushing memory replicas from the device to the cloud, Abdelwahab et al [2016] showed that pulling the replicas from the IoT devices can be more efficient. They proposed a protocol for computation offloading in the LTE environment with support of massive IoT devices. Gu et al. [2015] integrated fog computing and medical cyber-physical systems to host virtual medical applications. Data must be uploaded to the associated base stations (BSs) before being routed to a service instance. The authors jointly investigate BS association, tasks distribution and VM placement.

While most works have addressed the single-user computation offloading, Sardellitti et al. [2014] considered a much more general context: multiple mobile device users requesting computation offloading of their applications to the cloud. They formulated the problem as the joint optimization of the mobile radio resources and the computational resources. Alam et al. [2016] proposed a multi-agent based code offloading mechanism. To handle the dynamicity of resource requests, the method uses distributed reinforcement learning algorithm. Chen et al. [2016] also looked at the way to efficiently coordinate multiple wireless mobile devices for computation offloading. Differently, they proposed a game theory based approach that considers users interests during computation offloading.

4.5 Content Preprocessing

The desire to minimize the data movement overhead drives research on examining and pre-processing raw data at the edge. What is of particular interest is data in the form of time series, which are the primary kind of data in the IoT era. Surveillance camera, mobile phones, metering sensors all attempt to forward huge amount of time-series data (video, speech, text, etc.) continuously and in real time. Data reduction and data prioritization are two major approaches for content preprocessing.

General data reduction/compression technique includes sampling, summarization, and approximation. Oftentimes, more than one data reduction techniques are used to preprocess data at the edge. For example, in addition to general loss-less compression such as GNU zip, Fog Data [Dubey et al. 2015] uses lossy data reduction methods (feature extraction and pattern mining) to reduce overhead. Song et al. [2016] focused on structured files and proposed a content-oriented method that decomposes a file into objects according to its structure. To facilitate the switching among different data reduction algorithms at the edge, Papageorgiou et al. [2015] proposed NECtar, a unified data handling interface. NECtar agent is based on identifying PIP (Perceptually Important Points) and is able to make data reduction decisions per incoming data item (e.g. stream data).

Due to limited network bandwidth and energy capacity, data prioritization is very important for wireless multimedia big data. A common approach is to first determine the relative importance of different video frames, and then efficiently deliver the multimedia content to the cloud. Vigil [Zhang et al. 2015] aims at

optimizing the number of specified objects uploaded to remote data center while minimizing the bandwidth requirements. Vigil gives priority to more critical frames, especially when the wireless capacity is very limited. Wang et al. [2016] introduced the idea of multimedia sensing as a service (MSaaS). The resource allocation of MSaaS can be formulated as a video quality-aware (i.e., premium data vs. regular data) optimization problem with energy as major constraints. It has been further broken down into three sub-problems with regard to the frequency domain, the spatial domain, and the temporal domain.

4.6 Service Migration

One important aspect of edge resource management is to determine the location (in terms of which edge node) of running the service. Edge nodes are generally responsible for a relatively small geographical area. They are required to be deployed in close proximity to users to provide low-latency service access. A shared storage with ultra-high speed network connection is generally not available. Therefore, as a user moves across different locations, the service applications often need to be migrated to follow the user so that the benefit of edge computing is maintained. Taleb and Ksentini [2013] investigated the performance of service migration following user mobility and confirmed the advantages of service migration.

To determine whether or not the service should be migrated out of the original host edge system, researchers have considered the distance between the user and the service as a key parameter. Service migration incurs a cost that is considered to be a non-decreasing function of the distance. If we choose not to migrate, the migration cost is zero. However, the user still incurs a transmission cost due to the need to get connected to a remote service provider. Researchers have analyzed the above problem using Markov Decision Process (MDP) [Urgaonkar et al. 2015]. Ksentini et al. [2014] proposed mobility-driven service migration in the one-dimensional (1-D) scenario based on MDPs, while Wang et al. [2015a] considered a more realistic two-dimensional (2-D) mobility. Wang et al. [2015b] further pushed the theoretical results on step closer to practice. They evaluated the above scheme using real-world data and experimented with an emulated environment that contains one centralized cloud data center and multiple distributed micro data centers.

Service migration techniques are mainly used to handle streaming applications from widely distributed devices such as mobile sensor and camera [Ottenwalder et al. 2014, Saurez et al. 2016]. The assignment of a computing node to perform the stream processing has a significant impact on latency and bandwidth. Ottenwalder et al. [2014] showed that the placement of the stream processing has to be constantly adapted through migrations to new computing node. The idea is to alleviate performance degradation and reduce network utilization by predicting future migration targets and planning migration before they are required. The Foglet [Saurez et al. 2016] implementation focuses on meeting quality of service (QoS) expectations and a balanced resource usage among fog nodes. It can automatically mange the migration of application from one node to another.

Minimizing the impact of migration is also a crucial design consideration. It is often time consuming to copy program state during migration. It also takes time to actually offloading the data once a new edge node is discovered. Su and Flinn et al. [2005] proposed to dynamically make replicas of stateful applications to optimize the performance of mobile systems that have resource constraints. This allows the user to hide the perceived cost of migration. Other than virtual machine based migration, Rusta [Valvag, et al. 2013] choose to enable light-weight process migration between

clines of a group. It aims at facilitating using resources both on the client machine and in the cloud environment. Teka et al. [2016] proposed to use multipath TCP to ensure constant TCP connection when performing live migration. This mechanism greatly reduces migration latency between neighboring cloudlets. Machen et al. [2016] proposed a generic layered migration framework. A copy of the base layer is stored on every edge node, so that it does not need to be transferred in every migration.

4.7 Load Coordination

In the context of edge-level resource management, another line of work aims to coordinate various edge nodes.

Manjunath et al. [2004] introduced a client-server based multimedia management framework that allows edge users to adapt content, organize files, upload content, and share data with other machines. Grieco et al. [2005] proposed distributed dispatchers to dynamically balance the load (through round-robin strategy, least loaded node, etc.) on a cluster of heterogeneous edge machines.

Sheikh et al. [2015] considered a nano data center consists of a group of edge servers (also referred as nano servers). The first heuristics is to find a server which can fulfill the user request in the minimum expected time. The second heuristic is to find nano servers that are least recently used. Such server is likely to be light loaded and the request can be finished soon as time to serve a request contributes to the total power consumption. Note that sending request to the least loaded machine may be problematic if load information is stale [Dahlin, 1999]. Machines that appear to be underutilized may quickly become overloaded. Grieco et al. [2005] evaluated a modified algorithm on edge-computing services: sending requests to the least loaded server from a randomly selected server subset.

Recent works have started to adopt a software-defined approach. For example, Truong et al. [2015] proposed to combine fog computing and software defined networking (SDN) to coordinate tasks and resource for Vehicular Ad-hoc Networks (VANETs). Cellular base stations (BSs) and Road-Side-Unit Controller (RSUCs) in the VANETs are crucial virtualized equipment for enabling fog computing services. These BSs and RSUCs are coordinated by a SDN controller. With data profiling, the controller is able to maximize the utilities of vehicles, RSUCs, and RSUs.

The software-defined approach has been applied to edge embedded systems as well. Zeng et al. [2016] introduced fog-based software-defined embedded system (FC-SDES). The network edges (cellular base stations) are equipped with certain storage or computation resources. The authors investigated the interaction among embedded clients, a set of storage servers, and a set of computation servers. Consequently, the task completion time consists of three basic components, i.e., computation time, I/O time and transmission time. One can balance the workload on both client and edge side by jointly managing task scheduling and image placement.

Lai et al. [2016] explored QoS-aware multimedia streaming service over fog computing infrastructures. There are three kinds of nodes: an index node, a center node, and a group of edge nodes. The center node sends multimedia content to edge nodes according to the path determined by an index node. A dynamic edge selection mechanism is proposed to balance the loading of overall cloud network based on the loading of edge nodes. Destounis et al. [2016] also studied the dynamic resource allocation that arises in network computation. They aim at designing a universal methodology and algorithm to adaptively allocate resources in order to support maximum query rate. Their algorithm could orchestrate utilization of computation and bandwidth resources by performing dynamic load balancing, data routing, etc.

Cardellini [2015] extended Apache's Storm architecture to run a distributed QoS-aware scheduler. The proposed new architecture is suitable for managing streams of fog data. It has a monitoring module for estimating intra-node and inter-node network information. Based on the monitored information, a scheduler automatically performs operator reassignment when unexpected environmental changes occur.

4.8 Adaptive Provisioning

Considering the dynamism and uncertainty of edge requests, intelligently adjusting the capability/capacity of compute nodes at the fog layer is often necessary. Fog components should be orchestrated as adaptive systems that are able to react to internal and external changes of various operating conditions automatically.

A common way is to adjust the system by tuning operational parameter based on monitored information. Fog servers are believed to have the distinct advantage of obtaining the operating conditions (network congestion, device loading, etc.) local to an end user [Zhu et al. 2013]. One can therefore take advantage of some fog-specific knowledge of the environment to fine tune the system at real-time. Through a series of fog-based load scaling methods (reducing HPPT requests, adjusting web object sizes, and re-organizing webpage), researchers form Cisco [Zhu et al. 2013] showed that one can automate user-aware website performance optimization.

Xu et al. [2011] invested adaptation of data acquisition method in a cloud-fog-sensor environment. A push operation allows fog server to subscribe to a particular data source to receive continuous reading. When facing sporadic request, it is believed to be less economical due to excessive data transmission. In contrast, a pull operation represents on-demand data query that incurs a round-trip data communication overhead for each query. By properly mixing the two operations and fine-tuning data delivery, one can greatly improve performance and reduce energy.

On the other hand, research efforts have been made in building elastic fog resource capacity. For example, upon dynamic capacity scaling, fog servers can be designed to dynamically split/merge tasks [Hong et al. 2013]. This allows one to create on-demand fog compute instances that may follow system status. ThinkAir [Kosta et al. 2012] is a mobile cloud computing framework with on-demand resource allocation capability. Considering the high variance of workload, ThinkAir puts more emphasis on the dynamic creation, reusing, and recycling of resources in the cloud.

Aazam and Huh [2015] studied the model for resource prediction, reservation, and pricing in the fog. It allows for better capacity planning, service creation, and resource provisioning. To ensure that users can be served by edge servers with sufficient capacity, Yin et al [2016] explored how online service provider (OSP) cloud provision fog infrastructure with flexible server placements. They consider various pragmatic requirements such as cost budgets, performance requirements, traffic constraints and existing server deployments.

While computing and networking dynamics are the primary reasons for system adaptation, there are more factors that need to be considered. An example is building fog server system running on local renewable energy sources. InSURE [Li et al. 2015] is a rack-scale green energy powered fog infrastructure for in-field processing. Since InSURE is more likely to be deployed in rural areas, batteries are necessary to store green energy and serve as energy buffer. Adapting server loads (through frequency scaling, VM checkpointing, etc.) to renewable energy availability and battery discharging behavior is necessary in this scenario. Otherwise the system may incur increased service disruption, reduced data throughput and increased energy loss.

4.9 Resource Pooling/Sharing

Nowadays the edge layer is becoming highly dense. Various mobile devices and smart devices could form a fairly resource-rich computing environment that can be leveraged to support service [Miluzzo et al. 2012; Abedin et al. 2015; Flores et al. 2016]. It can be treated as a special form of cyber foraging that offloading is performed primarily among resource-constrained mobile/fixed gargets [Shi et al. 2012]. Undoubtedly, this approach enables a range of applications that need to leverage external resources in proximity to process data timely. It is especially important in scenarios that fixed cloud/fog infrastructure is unavailable, highly unreliable, or prohibitively expensive (e.g., military, disaster, and rural area).

The dynamically formed resource pool is known to be device cloud [Fahim et al 2013]. If needed, different device clouds can further be connected to various sensor networks. They can share their resources/services in a seamless manner though appropriate design patterns (e.g. REST) and protocols (e.g. CoAP) [Shi et al. 2015]. On the other hand, a device may not have to share resource with distant nodes. For some high-end devices, cooperative device pairing based on short-range device-to-device (D2D) communication can be enough [Abedin et al. 2015].

Efficient edge services require fine-grained computation/data offloading at the device level. Recently, Wang al. [2017] proposed a graph partition based functional cell distribution scheme between the sensor node and data aggregator. Reiter et al. [2017] proposed a general model that uses edge resources to solve the energy and performance issue of mobile devices. The benefits of near-edge offloading have been evaluated in prior work [Fahim et al 2013]. Having larger portion of computation provides more energy conservation and time saving opportunities. For larger data sets, this gain reduces due to the overhead introduced by data communication.

The resource management problem of device cloud can be even trickier. There is an important trade-off between reduced energy consumption and the gains from the cooperation with distant nodes. Increasing the number of cooperating devices allows for a larger pool of resources, while it requires more energy for data transmission to obtain the increased communication rage [Nishio et al. 2013].

In addition to energy consumption, responsiveness is another important concern. For device cloud users, service delay is a straightforward measure of the decrease of quality of experience (QoE). To quantify the satisfaction perceived by mobile user, many prior works have used a utility-based model [Miluzzo et al. 2012, Abedin et al. 2015]. The utility of nodes in the device cloud can be modeled as a function of latency [Nishio et al. 2013]. For many IoT applications, enabling real-time control is critical. Maestro [Viswanathan et al. 2016] is designed for concurrent applications on the device cloud. Here a mobile application is represented as a workflow (Directed Acyclic Graph) composed of multiple parallelizable tasks along with dependencies. Maestro proactively replicates critical tasks to avoid deadline violation. It also performs task deduplication to simplify workflow and improve resource utilization.

Many other issues need to be taken into consideration when sharing resources. First, the computing capability of other devices is largely unknown. Oftentimes, the device cloud exhibits a high degree of node heterogeneity [Nishio et al. 2013]. Second, the device cloud space is somewhat fragmented: resources are usually measured in disparate ways (energy budget, network latency, available bandwidth, etc.). A possible solution is to design a unified framework that uses “time” as the universal metric for different resources [Nishio et al. 2013]. Third, intermittent connectivity among devices could be normal and there are randomly joined nodes. To tackle these issues, Fernando et al. [2016] introduced Honeybee, a new resourcing sharing

Table III. A Classification of Works on Resource and Workload Management Approaches

Approaches	Related Works
Hybrid Provisioning	[Chandra et al. 2009], [Sundarrajan et al. 2011], [Kannan et al. 2011], [Ryden et al. 2014], [Jang et al. 2014], [Chang et al. 2014]
Task Placement	[Steiner, et al. 2012], [Chandramouli et al. 2012], [Hao et al. 2014], [Deng et al. 2016], [Fricker et al. 2016], [Souza et al. 2016], [Masip-Bruin et al. 2016], [Kang et al. 2017]
Distributed Caching	[Gao et al. 2003], [Davis et al. 2004], [Ramaswamy et al. 2007], [Lin et al. 2007], [Xu et al. 2014], [Romano and Quaglia 2014], [Lin et al. 2015], [Zeydan et al. 2016], [Lin et al. 2016], [Park 2016], [Malandrino et al. 2016]
Computation Offloading	[Sardellitti et al. 2014], [Orsini et al. 2015], [Oueis et al. 2015], [Gu et al. 2015], [Abdelwahab et al. 2016], [Hu et al. 2016], [Chen et al. 2016], [Tong et al. 2016], [Alam et al. 2016]
Data Preprocessing	[Rooney et al. 2005], [Shi et al. 2014], [Papageorgiou et al. 2015], [Zhang et al. 2015], [Dubey et al. 2015], [Wang et al. 2016], [Song et al. 2016]
Service Migration	[Su and Flinn 2005], [Valvag et al. 2013], [Ottenwalder et al. 2014], [Urgaonkar et al. 2015], [Wang et al. 2015a], [Wang et al. 2015b], [Machen et al. 2016], [Saurez et al. 2016], [Teka et al. 2016], [Willis et al. 2014]
Load Coordination	[Manjunath et al. 2004], [Grieco et al. 2005], [Truong et al. 2015], [Sheikh et al. 2015], [Cardellini et al. 2015], [Lai et al. 2016], [Destounis et al. 2016], [Zeng et al. 2016]
Adaptive Provisioning	[Xu et al. 2011], [Kosta et al. 2012], [Zhu, et al. 2013], [Aazam and Huh 2015], [Li et al. 2015], [Yin et al. 2016], [Hong et al. 2016]
Resource Sharing	[Newton et al. 2009], [Miluzzo et al. 2012], [Nishio et al. 2013], [Fahim et al. 2013], [Shi et al. 2015], [Abedin et al. 2015], [Viswanathan et al. 2016], [Fernando et al. 2016], [Pamboris and Pietzuch 2016], [Flores et al. 2016], [Wang et al. 2017], [Reiter et al. 2017]

strategy that leverages work stealing to balance loads among devices and periodically discovery resources to manage the dynamism.

Prior proposals have explored optimizing the ability to partition application code. For example, Wishbone [Newton et al. 2009] looks at sensing application configured as a stream-oriented collection of operators (dataflow graph). It partitions stream operators between sensors and servers to minimize resource. Wishbone determines how each operator in the dataflow graph will actually perform based on profiling information. C-RAM [Pamboris and Pietzuch, 2016] is a partition strategy that can split program state between a mobile device and a remote computing node without developer intervention. It automatically partitions application source code based on fine-grained profiling. Once remote node fails, C-RAM can switch to offline execution with the support of a virtual memory mechanism designed for iOS.

4.10 Classification

Finally, Table III maps the taxonomy of resource management approaches to prior works that have been surveyed in this section.

5. OPTIMIZATION OBJECTIVES

In this section we describe prior works based on their optimization objectives and the heuristics/strategies used by them.

5.1 Delay and Execution Time

Fog systems normally need to handle time-sensitive requests under agreed Service Level Agreements (SLAs). The presence of system/workload dynamics has a significant impact on the latency of fog computing.

5.1.1. Transmission Time. Many works chose to reduce transmission time via smart network adaptation. Lai et al. [2016] proposed to adapt video data to the current network conditions. The edge node receives the multimedia content from a specific center node according to the path given by the index node, and adjusts content according to the end device's bandwidth. Romano et al. [2014] considered network path diversity and proposed a selection logic that allows the identification of the edge server that might provide better performance guarantee. Yin et al. [2016] developed a novel method to take advantage of the latency ranking, which is much less sensitive to delay prediction errors to search the ideal edge location.

With appropriated management, a multi-layer fog may further reduce delay. Souza et al. [2016] investigated a dual-layer topology aiming to avoid the high-latency access on upper fog layers. The computation of total delay for a service considers each fog layer with different delay. Tong et al. [2016] enabled execution of a mobile application on hierarchical edge cloud servers. Placing mobile programs at high-tier servers reduces resource contention, but also spends a longer time on transmitting the program states to these servers via the network. Thus, a joint optimization of workload optimization and node capacity provisioning is necessary.

Reducing transmission time is also critical for data/service migration. The transmission time (between the access point of a mobile sensor and the computing node that performs processing) changes dynamically [Ottenwalder et al. 2014]. Su and Flinn [2005] showed that replication allows for seamless service migration between surrogates with negligible latency. Teka et al. [2016] argued that low-latency VM migration can be achieved with multipath TCP (MPTCP).

Reducing data size is an effective way to reduce transmission time. Fog computing nodes with data deduplication capabilities are ideal for time-sensitive tasks [Dubey et al. 2015]. In addition, one can also trade off data quality for lower latency. To reduce transmission delay under unfavorable network condition, Lin et al. [2014b] adjusted encoding bitrates to lower video quality level on their fog nodes. Many workloads include both delay-tolerant and delay-intolerant data. Yassine et al. [2016] investigated dynamic scheduling of multimedia edge data. Delay-tolerant requests include data replications and backup contents that typically have very large deadlines that span few hours to days. Delay-intolerant data include video processing and massively interactive video games that must be transferred between data centers at a specific rate and time.

5.1.2. Execution Time. The problem of minimizing execution time requires optimal task allocation. Nebula [Ryden et al. 2014] intends to select the best performing server nodes for computation to reduce job execution time. Viswanathan et al. [2016] showed that task deduplication leads to efficient real-time, in-situ processing of simplified workflows (with fewer tasks than before) as well as to better utilization of computing resources. Alam [2016] explored a machine learning based mobile agent who can reduce execution time.

5.1.3. Round-Trip Time. Viswannathan et al. [2013] argued that ubiquitous mobile applications require real-time analysis of raw data generated in the field. Both the task property and resource selection have an effect on the timeliness. They proposed a stage-wise threshold-based heuristic for assigning tasks to resource providers. To select the right data center, Steiner et al. [2012] collected live measurements of network and compute performance. An overall score is calculated that takes account of various resources. CloudAware's optimization solver [Orsini et al. 2015] is able to decide which compute instances on what edge nodes to use for offloading. The solver

calculates the cost that is derived from a specific optimization target such as computation acceleration, bandwidth saving, latency minimization, etc.

Oueis et al. [2015] proposed a two-step resource allocation process to satisfy as much user request as possible. In the first step local resources of fog components are allocated based on a specific scheduling rule. In the second step, unsatisfied requests (due to resource scarcity) are processed by a dynamically formed fog node cluster, following certain optimization objective. Priority is given to tasks with the lowest latency tolerance. This imposes an earliest deadline first (EDF) scheduling.

5.1.4. Real-time Support. Providing real-time support mechanism for streaming data has been explored from different aspects. Chandramouli et al. [2012] looked at continuous queries running over streaming data from a large number of edge-devices and the Cloud. Nishio et al. [2013] modeled the utility of nodes as a function of the round-trip latency. Abdelwahab et al. [2015a] proposed to perform sensing resource discovery through a gossip based algorithm that requires a time complexity of $O(r^{-1}\log n)$. Their design objective is to construct the virtual domains, as fast as possible and with minimal communication overhead between the agent and the devices as well as between the devices. Papageorgiou et al [2015a] proposed to "streamify" data reduction techniques that are otherwise not used in real-time scenarios, but rather on the complete data set. Their technique avoids delays in pre-processing and forwarding data items. Their basic ideas are: 1) being applicable in real-time and per item; 2) not delaying the data item forwarding.

5.2 Energy and Power Demand

Energy is consumed when fog components performs computation or send/receive data. It is crucial to account for power behaviors while designing fog systems.

5.2.1. Communication Energy. In the wireless communication domain, communication energy often refers to radio energy consumption. Due to the proximity to users, fog systems have lower radio energy consumption [Alam et al. 2016]. A more detailed transmission expenditure model may consider the energy consumption due to electric circuit, data aggregation, and signal amplification [Abedin et al. 2015]. In the fog radio access network, some communication hubs such as remote radio heads (eRRH) can be subject to transmit power constraint [Park et al. 2016].

The communication energy typically depends on the size of the transmitted data and the network bandwidth. Transmission of a large snapshot can considerably increase device energy consumption [Pamboris et al. 2016]. For a Cloudlet, VM synthesis typically consumes significant amount of power due to the large payload. Lewis et al, [2014] showed by realistic measurements that the energy consumption of mobile device that perform offloading is directly proportional to payload size.

Nishio et al. [2012] discussed a trade-off between performance gain and energy saving: it often requires more transmission energy to obtain larger communication rage. Fahim et al. [2013] developed an experimental platform to show insights into the energy behaviors of device cloud. Barbera et al [2013] studied the data communication overhead under different levels of synchronization and report the costs in terms of energy. Hu et al. [2016] qualitatively compared different offloading scenarios (cloudlet vs. remote cloud). Real measurements showed that offloading to a remote data center can result in even higher energy costs than running locally.

Similar to latency optimization, an important approach to communication power reduction is minimizing data transmission size. Data reduction technique has reduced the power needed for data transmission (from Fog to the cloud) for the Fog

Data architecture [Dubey et al. 2015]. Given wireless communication energy budget, Wang et al. [2016] divided an image into crucial premium information and unimportant regular information to ensure optimal multimedia transmission quality.

Another line of work has been focused on the communication pattern in the cloud and fog environment. If too many users choose the same wireless channel to offload simultaneously, they may cause undesirable interference to each other. This hence can lead to low data rates, long data transmission time and low energy efficiency [Chen et al. 2015]. Current LTE network incurs such an inefficiency issue facing large simultaneous access from ubiquitous devices. Sardellitti et al. [2014] looked for the best way to connect users to a nearby base station and to a remote cloud so that the overall communication energy is minimized. Xu et al. [2011] studied different data acquisition methods, i.e., push and pull mechanism. By dynamically adjusting the base push and supplementary pull rates for each sensor, the total energy cost is reduced. Abdelwahab et al. [2015b] also looked at the pull and push operation of memory replicas. They took advantage of the sparsity at both the network and memory levels to reduce delay and energy consumption in a LTE environment.

5.2.2. Computation Energy. Computation power has long been a problem in servers and data centers. Laoutaris et al. [2008] summarized the power and energy issue of conventional data center and show that nano data center could help address it. Jalali et al. [2014] further described the model for estimating the energy consumption of accessing data from nano datacenters. The model allows one to identify applications that are best suitable for running on nano data centers. Lebre et al. [2016] further provided a detailed model for estimating the computation and cooling power for distributed cloud infrastructure in the fog era.

Execution time is a major factor that affects computing energy in the fog. This has been proved on both fog system and edge devices. For example, on-demand VM provisioning on cloudlet incurs high energy consumption due to the relatively longer application-ready time [Lewis et al., 2014]. The partitioned execution scheme of mobile applications [Pamboris et al. 2016] has shown to be more energy-efficient due to significantly reduced execution time on mobile devices. Sheikh et al. [2015] propose to select a server that is the least recently used – such a server are likely to be lightly loaded and therefore may result in a faster execution time.

To reduce computation power, various low-power designs and control techniques are used. Fog data [Dubey et al. 2015] uses low power processor node, namely, Intel Edison to process data locally. Stanley-Marbell [2011] designed and implemented an embedded multiprocessor system for energy-efficient in-situ data processing. Wen et al. [2012] propose to dynamically configure the clock frequency in the mobile device to reduce energy consumption. Manjunath et al. [2004] enhanced the Quality of Service (QoS) while maximizing the battery lifetime of the client devices. The system has to communicate to the client ahead of time about the size of the data burst to receive before going into a low power mode and when to wake up for the next burst.

Running large tasks remotely can save significant amounts of power, and is often orthogonal to other system-level power saving techniques [Rudenko et al. 1998]. For battery-powered systems, a convenient way is to set a timeout and offload program if the task is not finished after a given timeout [Xian et al. 2007]. When performing offloading, both the energy consumption of remote nodes and the residual energy available to these nodes should be considered. Shi et al. [2012] proposed an energy-aware scheme that can allocate more tasks to devices with enough residual energy.

Considering the ubiquitous service requests, Do et al. [2015] argued that data center carbon footprint would increase without appropriate resource allocation. To

optimize sustainability, Li et al. [2015] proposed to power fog servers with green energy sources (such as solar panel) and use battery as energy buffer/backup. Dynamic control of the energy flow from the solar panel to fog server has been shown to be crucial for maintaining high throughput and minimize service disruption.

5.3 Scalability and Availability

The demands for service availability and scalability have increased substantially over the years. Xie et al. [2002] argued that distributing jobs to an overlay network of edge servers improves both service availability and scalability. Ramaswamy et al. [2007] devised a scalable content delivery network that emphasizes the cooperation among edge caches near the network edge. Romano et al. [2014] presented a protocol which exploits the node diversity and redundancy to improve the end-to-end latency. This protocol is inherently scalable since it requires the slightest coordination among edge nodes. Recently, Pahl et al. [2016b] proposed to use a container-based system to deal with scalability, flexibility, and multi-tenancy concerns.

To improve the availability as well as latency, Gao et al. [2003] proposed to decouple edge server's local request processing from the store-and-forward processing of orders to the backend server. At the distributed cloud level, Hancock [2015] devised availability policies that leverage multiple service providers to provide video files – a failure of any provider will not affect the reachability of the files. For those surrogates of the cloud, they have also been designed to rapidly capture dynamics of the devices includes connectivity, and availability [Abdelwahab et al. 2015].

Another way to improve resiliency is to create checkpoints for critical states. Network failures greatly affect techniques that offload part of the application to remote nodes [Pamboris, et al. 2016]. By periodically retrieving a snapshot of the program state on remote server, local devices can easily recover from a failure.

To ensure that the edge computing system is running properly, heartbeat signals are widely used. For example, Ramaswamy et al. [2007] proposed a failure resilient edge cache network. It employs heartbeat messages to detect failures of caches. To ensure failure resiliency, each document's lookup and update information is replicated at all the caches in the document's beacon ring. Nebular [Ryden et al. 2014] is a fog platform resilient to a variety of failures in a volunteer computing environment. If the heartbeats are missed, the system will try to re-execute tasks to achieve fault tolerance. Similarly, Honeybee [Fernando et al. 2016] also uses a periodic signal as an indication that the worker is alive and the network is accessible.

6. CHALLENGES AND OPPORTUNITIES

Although many prior works have explored the design and optimization of an edge-oriented computing system, several key problems remain open. From the studies surveyed above, we have identified the following opportunities that are expected to drive innovation within this domain:

- ***New Hardware Technologies***. Very few fog/edge computing designs today intend to incorporate emerging hardware technologies. This includes, but are not limited to, non-volatile storage technologies, low-power reconfigurable devices such as FPGAs, 3D-stacking that integrate DRAM memories on top of cores, photonics for communication, etc. Significant architectural advancement at the edge level is required to exploit these promising technologies.
- ***Application-Driven Processing***. It is crucial to exploit parallelism on ubiquitous fog/edge machines, i.e., rethinking system (architecture, operating systems, virtual machines, compiler tools, and programming languages) design

for enabling fine-grained fog device level parallelism. Importantly, edge data processing should be able to adapt to the emerging applications (e.g., training and inference workloads in the era of AI) and environment.

— ***Heterogeneity & Dynamism.*** Edge computing is a double-edged sword. In the fog environment, heterogeneity is both larger in scale and greater in depth. How to combine edge devices into a talent pool is considered the foremost challenge any edge-oriented computing systems need to take care of today and tomorrow. The system should be able to orchestrate the functionalities of an unbalanced network with nodes of various cluster sizes, different storage space, mismatched computation power and unpredictable dynamics.

— ***Cross-Layer Co-Exploration***. Current fog/edge system designs are largely decoupled from data center level management. Most data centers today heavily rely on aggressive resource management to improve system utilization while maintaining QoS. Leveraging the data processing capabilities at different layers of the computing stack can contribute to enabling data centers to stay on track with their performance requirement. Importantly, the resulting system is expect to exhibit a synergism in which the Cloud-Fog-Edge system can yield a cost-effectiveness far better than that can be offered today.

— ***Energy and Sustainability***. Power and energy management are still open to research due to the ever-increasing compute load, evolving workload patterns, and heightening demand for cost-effectiveness and sustainability. In recent years, many computing systems even start to tap into non-conventional power provisioning solutions such as renewable power supply, energy storage systems (battery, supercapacitor) and aggressive power capping techniques. Harvesting distributed energy (solar, wind) can be an interesting step ahead for power management research in the fog and edge computing era.

— ***Human Activity Awareness***. A possible future work addressing fog/edge performance and efficiency includes bringing human factors to edge computer design and management. For example, given user feedback, the fog system may leverage approximate computing to reduce energy overhead. The system can also provide task acceleration based on user habit information. Many more opportunities might arise, provided a strong support in the technology itself.

— ***Security, Privacy, and Resiliency***. Some of the other future works include security measures and resiliency supports in the paradigm. Although not discussed in this survey, security and privacy are important design considerations. IoT systems today are still highly vulnerable and are likely to cause widespread disruption of services. It is more difficult to address because unlike protection for cloud centers, protection for edge devices are more diversified and therefore more complicated.

7. CONCLUSION

Although it would still take time for edge-oriented computing systems to evolve from prototypes to our daily common platforms, the concepts and the technologies themselves are without doubt promising and exciting. Edge computer systems of various forms are projected to become crucial links for an efficient information processing architecture in the near future. Recent years have witnessed an increasing number of system studies in this area. On this foundation, many parties have also proposed resource management and optimization strategies. In this article,

we have surveyed a wide range of edge-oriented computing systems. It is hoped that this article will provide useful insights for both academics and practitioners.

ACKNOWLEDGMENTS

We thank all the anonymous reviewers for their valuable feedbacks. Chao Li is also generously supported by the CCF-Tencent Open Fund, CCF-Intel Young Faculty Research Program Award, and a SJTU-MSRA Faculty Award. Corresponding authors are Chao Li at Shanghai Jiao Tong University and Weigong Zhang at Capital Normal University.

REFERENCES

Mohammad Aazam and Eui-Nam Huh. 2015. Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. In *Proceedings of the 29th Int. Conf. on Advanced Information Networking and Applications*. IEEE, Piscataway, NJ, 687-694. DOI: 10.1109/AINA.2015.254

Sarder Fakhrul Abedin; Md. Golam Rabiul Alam; Nguyen H. Tran; Choong Seon Hong. 2015. A Fog based system model for cooperative IoT node pairing using matching theory. In *Proceedings of the Asia-Pacific Network Operations and Management Symposium*. IEEE, Piscataway, NJ, 309-314. DOI: 10.1109/APNOMS.2015.7275445

Sherif Abdelwahab, Bechir Hamdaoui, Mohsen Guizani, and Taieb Znati. 2015. Cloud of things for sensing as a service: Sensing resource discovery and virtualization. In *Proceedings of the 2015 IEEE Global Communications Conference*. IEEE, Piscataway, NJ, 1-7. DOI: 10.1109/GLOCOM.2015.7417252

Sherif Abdelwahab, Bechir Hamdaoui, Mohsen Guizani, and Taieb Znati. 2016. REPLISOM: Disciplined tiny memory replication for massive IoT devices in LTE edge cloud. *IEEE Internet of Things Journal* 3, 3 (June 2016), 327 – 338. DOI: 10.1109/JIOT.2015.2497263

Md. G.R. Alam, Yan K. Tun, and Choong S. Hong. 2016. Multi-agent and reinforcement learning based code offloading in mobile fog. In *Proceedings of the 2016 Int. Conference on Information Networking*. IEEE, Piscataway, NJ, 285-290. DOI: 10.1109/ICOIN.2016.7427078

Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys* 36, 4 (Dec. 2004), 335–371. DOI: 10.1145/1041680.1041681

Oreoluwatomiwa Babarinsa and Stratos Idreos. 2015. JAFAR: Near-data processing for data bases. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, New York, NY, 2069-2070. DOI: 10.1145/2723372.2764942

Marco V. Barbera, Sokol Kosta, Alessandro Mei, and Julinda Stefa. 2015. To offload or not to offload? The bandwidth and energy costs of mobile cloud computing. In *Proceedings of the 2015 IEEE Conference on Computer Communications*. IEEE, Piscataway, NJ, 1285-1293. DOI: 10.1109/INFCOM.2013.6566921

Paolo Bellavista, Antonio Corradi, Mario Fanelli, and Luca Foschini. 2012. A survey of context data distribution for mobile ubiquitous systems. *ACM Computing Surveys*: Vol 44, Issue 4. ACM, New York, NY, 69-80. DOI: 10.1145/2333112.2333119

Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, Jiang Zhu. 2014. Fog computing: A platform for Internet of Things and analytics. *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer International Publishing, 169-186. DOI: 10.1007/978-3-319-05029-4_7

Flavio Bonomi, Rodolfo Milito, Jiang Zhu, Sateesh Addepalli. 2012. Fog computing and its role in the Internet of Things. In *Proceedings of the first edition of the MCC workshop on Mobile Cloud Computing*. ACM, New York, NY, 13-16. DOI: 10.1145/2342509.2342513

Stefano Buzzi, Chih-Lin I, Thierry E. Klein, H. Vincent Poor, Chenyang Yang, Alessio Zappone. 2016. A survey of energy-efficient techniques for 5G networks and challenges ahead. *IEEE Journal on Selected Areas in Communications,* Vol. 34, No. 4, (2016). IEEE, Piscataway, NJ, 697-709. DOI: 10.1109/JSAC.2016.2550338

Cisco. 2015. Fog computing and the Internet of Things: Extend the cloud to where the things are. http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

Valeria Cardellini, Vincenzo Grassi, Francesco Lo Presti, Matteo Nardelli. 2015. On QoS-aware scheduling of data stream applications over fog computing infrastructures. In *Proceedings of the 2015 IEEE Symp. on Computers and Communication*. IEEE, Piscataway, NJ, 271-276. DOI: 10.1109/ISCC.2015.7405527

Abhishek Chandra and Jon Weissman. 2009. Nebulas: Using distributed voluntary resources to build clouds. In *Proceedings of Hot Topics in Cloud Computing*. USENIX Association, Berkley, CA.

Badrish Chandramouli, Joris Claessens, Suman Nath, Ivo Santos, Wenchao Zhou. 2012. RACE: real-time applications over cloud-edge. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, New York, NY, 625-628. DOI: 10.1145/2213836.2213916

Hyunseok Chang, Adiseshu Hari, Sarit Mukherjee, and T.V. Lakshman. Bringing the cloud to the edge. In *Proceedings of the 2014 IEEE INFOCOM Workshop on Mobile Cloud Computing*. IEEE, Piscataway, NJ, 346-351. DOI: 10.1109/INFCOMW.2014.6849256

Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. 2016. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. on Networking 24, 5 (Oct. 2016), 2795-2808. DOI: 10.1109/TNET.2015.2487344

Mung Chiang, and Tao Zhang. 2016. Fog and IoT: An overview of research opportunities. *IEEE Internet of Things*

Journal, in press. DOI: 10.1109/JIOT.2016.2584538

Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, Ashwin Patti. 2011. CloneCloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth European Conference on Computer Systems (EuroSys)*. ACM, New York, NY, 301-314. DOI: 10.1145/1966445.1966473

M. Dahlin. 2000. Interpreting stale load information. *IEEE Trans. on Parallel and Distributed Systems* 11, 10 (Oct. 2000), 1033-1047. DOI: 10.1109/71.888643

Amir V. Dastjerdi and Rajkumar Buyya. 2016. Fog computing: Helping the Internet of Things realize its potential. *Computer* 49, 8 (August 2016), 112-116. DOI: 10.1109/MC.2016.245

Andy Davis, Jay Parikh, and William E. Weihl. 2004. EdgeComputing: Extending enterprise applications to the edge of the Internet. In *Proceedings of the 13th International World Wide Web Conference*. ACM, New York, NY, 180–187. DOI: 10.1145/1013367.1013397

Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H. Luan, and Hao Liang. 2016. Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption. *IEEE Internet of Things Journal*, in press. DOI: 10.1109/JIOT.2016.2565516

Apostolos Destounis, Georgios S. Paschos, Iordanis Koutsopoulos. 2016. Streaming big data meets backpressure in distributed network computation. In *Proceedings of the 35th IEEE International Conference on Computer Communications*. IEEE, Piscataway, NJ, 1-9. DOI: 10.1109/INFOCOM.2016.7524388

Hoang T. Dinh, Chonho Lee, Dusit Niyato, Ping Wang. 2011. A survey of mobile cloud computing: Architecture, applications, and approaches. *Wireless Communications and Mobile Computing* 13, 18, 1587–1611. DOI: 10.1002/wcm.1203

Cuong T. Do, Nguyen H. Tran, Chuan Pham, Md. G. R. Alam, Jae H. Son, and Choong S. Hong. 2015. In *Proceedings of the International Conference on Information Networking*. IEEE, Piscataway, NJ, 324-329. DOI: 10.1109/ICOIN.2015.7057905

Clinton Dsouza, Gail-Joon Ahn, and Marthony Taguinod. 2014. Policy-driven security management for fog computing: preliminary framework and a case study. In *Proceedings of the IEEE 15th Int. Conf. on Information Reuse and Integration*. IEEE, Piscataway, NJ, 16-23. DOI: 10.1109/IRI.2014.7051866

Harishchandra Dubey, Jing Yang, Nick Constant, Amir Mohammad Amiri, Qing Yang, Kunal Makodiya. 2015. Fog Data: Enhancing telehealth big data through fog computing. In *Proceedings of the ASE Big Data & Social Informatics*, Article No. 14. DOI: 10.1145/2818869.2818889

ETSI. 2014. Mobile-edge computing. https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf

Mohammad Abdullah Al Faruque and Korosh Vatanparvar. 2016. Energy management-as-a-service over fog computing platform. *IEEE Internet of Things Journal*. Vol 3 NO. 2. IEEE, Piscataway, NJ, 161–169. DOI: 10.1109/JIOT.2015.2471260

Afnan Fahim, Abderrahmen Mtibaa, and Khaled A. Harras. 2013. Making the case for computational offloading in mobile device clouds. In *Proceedings of the 19th International Conference on Mobile Computing and Networking*. ACM, New York, NY, 203-205. DOI: 10.1145/2500423.2504576

Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu. 2016. Computing with nearby mobile devices: A work sharing algorithm for mobile edge-clouds. *IEEE Transactions on Cloud Computing*, in press. DOI: 10.1109/TCC.2016.2560163

Huber Flores, Satish Narayana Srirama, and Rajkumar Buyya. 2014. Computational offloading or data binding? Bridging the cloud infrastructure to the proximity of the mobile user. In *Proceedings of the 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. IEEE, Piscataway, NJ, 10–18. DOI: 10.1109/MobileCloud.2014.15

Huber Flores, Rajesh Sharma, Denzil Ferreira, Chu Luo, Vassilis Kostakos, Sasu Tarkoma, Pan Hui, Yong Li. 2016. Social-aware device-to-device communication: A contribution for edge and fog computing? In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, New York, NY, 1466-1471. DOI: 10.1145/2968219.2968589

Christine Fricker, Fabrice Guillemin, Philippe Robert, and Guilherme Thompson. 2016. Analysis of an offloading scheme for data centers in the framework of fog computing. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 1, 4 (Sep. 2016), Article 16. DOI: 10.1145/2950047

Lei Gao, Mike Dahlin, Amol Nayate, Jiandan Zheng, Arun Iyengar. 2003. Application specific data replication for edge services. In *Proceedings of the 13th International World Wide Web Conference*. ACM, New York, NY, 449–460. DOI: 10.1145/775152.775217

Nam Giang, Michael Blackstock, Rodger Lea, and Victor C.M. Leung. 2015. Developing IoT applications in the fog: A distributed dataflow approach. In *Proceedings of the 5th International Conference on the Internet of Things*. IEEE, Piscataway, NJ, 155-162. DOI: 10.1109/IOT.2015.7356560

Raffaella Grieco, Delfina Malandrino, and Vittorio Scarano. 2005. SEcS: Scalable edge-computing services. In *Proceedings of the 2005 ACM symposium on Applied computing*. ACM, New York, NY, 1709–1713. DOI: 10.1145/1066677.1067063

Lin Gu, Deze Zeng, Song Guo, Ahmed Barnawi, and Yong Xiang. 2016. Cost-efficient resource management in fog computing supported medical CPS. IEEE Trans. on Emerging Topics in Computing, in press. DOI: 10.1109/TETC.2015.2508382

Tao Guan, Ed Zaluska, and David D. Roure. 2005. A grid service infrastructure for mobile devices. 2005. In *Proceedings of the 1st International Conference on Semantics, Knowledge and Grid*. IEEE, Piscataway, NJ, 42-42. DOI: 10.1109/SKG.2005.10

Matthew B. Hancock and Carlos A. Varela. 2015. Augmenting performance for distributed cloud storage. In *Proceedings of the 15th International Symposium on Cluster, Cloud and Grid Computing*. IEEE, Piscataway, NJ, 1189–1192. DOI: 10.1109/CCGrid.2015.124

Fang Hao, Murali Kodialam, T.V. Lakshman, and Sarit Mukherjee. 2014. Online allocation of virtual machines in a distributed cloud. *IEEE/ACM Transactions on Networking*, in press. DOI:10.1109/TNET.2016.2575779

Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, Boris Koldehofe. 2013. Mobile fog: A programming model for large-scale application on the Internet of Things. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing*. ACM, New York, NY, 15–20. DOI: 10.1145/2491266.2491270

Wenlu Hu, Ying Gao, Kiryong Ha, Junjue Wang, Brandon Amos, Zhuo Chen, Padmanabhan Pillai†, Mahadev Satyanarayanan. Quantifying the impact of edge computing on mobile applications. In *Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems*. ACM, New York, NY, Article No. 5. DOI: 10.1145/2967360.2967369

Gonzalo Huerta-Canepa and Dongman Lee. 2008. An adaptable application offloading scheme based on application behavior. In *Proceedings of the 22nd Int. Conf. on Advanced Information Networking and Applications - Workshops*. IEEE, Piscataway, NJ, 387–392. DOI: 10.1109/WAINA.2008.148

Bukhary Ikhwan Ismail, Ehsan Mostajeran Goortani, Mohd Bazli Ab Karim, Wong Ming Tat, Sharipah Setapa, Jing Yuan Luke, Ong Hong Hoe. 2015. In *Proceedings of the 2015 IEEE Conference on Open Systems*. IEEE, Piscataway, NJ, 130–135. DOI: 10.1109/ICOS.2015.7377291

Fatemeh Jalali, Rob Ayre, Arun Vishwanath, Kerry Hinton, Tansu Alpcan, Rod Tucker. 2014. Energy consumption of content distribution from nano data centers versus centralized data centers. *ACM SIGMETRICS Performance Evaluation Review* 42, 3 (Dec 2014). DOI: 10.1145/2695533.2695555

Michael Jarschel, Daniel Schloser, Sven Scheuring, Tobias Hoßfeld. 2011. An evaluation of QoE in cloud gaming based on subjective tests. In *Proceedings of Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, Piscataway, NJ, 330-335. DOI: 10.1109/IMIS.2011.92

Minsung Jang, Karsten Schwan, Ketan Bhardwaj, Ada Gavrilovska, and Adhyas Avasthi. 2014. Personal clouds: Sharing and integrating networked resources to enhance end user experiences. In *Proceedings of 2014 IEEE Conference on Computer Communications*. IEEE, Piscataway, NJ, 2220-2228. DOI: 10.1109/INFOCOM.2014.6848165

Sudarsun Kannan, Ada Gavrilovska, and Karsten Schwan. 2011. Cloud4Home — Enhancing data services with @Home clouds. In *Proceedings of the 31st International Conference on Distributed Computing Systems*. IEEE, Piscataway, NJ, 539-548. DOI: 10.1109/ICDCS.2011.74

Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, Lingjia Tang. 2017. Neurosurgeon: Collaborative intelligence between cloud and the mobile edge, In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, New York, NY, 615-629. DOI: 10.1145/3093336.3037698

Rafiullah Khan, Sarmad Ullah Khan, Rifaqat Zaheer, and Shahid Khan. 2012. Future Internet: the Internet of Things architecture, possible applications and key challenges. In *Proceedings of 10th International Conference on Frontiers of Information Technology*. IEEE, Piscataway, NJ, 257-260. DOI: 10.1109/FIT.2012.53

Stojan Kitanov, Edmundo Monteiro, and Toni Janevski. 2016. 5G and the fog - survey of related technologies and research directions. In *Proceedings of the 18th Mediterranean Electrotechnical Conference*. IEEE, Piscataway, NJ, 1-6. DOI: 10.1109/MELCON.2016.7495388

Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, Xinwen Zhang. 2012. ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *Proceedings of 2012 IEEE Conference on Computer Communications*. IEEE, Piscataway, NJ, 945-953. DOI: 10.1109/INFCOM.2012.6195845

Philippe Kruchten. 1995. Architectural blueprints — the "4+1" view model of software architecture. IEEE Software 12, 6 (Nov 1995), 42-50.

Ablen Ksentini, Tarik Taleb, and Min Chen. 2014. A Markov decision process-based service migration procedure for follow me cloud. In *Proceedings of the IEEE International Conference on Communications*. IEEE, Piscataway, NJ, 1350-1354. DOI: 10.1109/ICC.2014.6883509

Chin-Feng Lai, Dong-Yu Song, Ren-Hung Hwang, and Ying-Xun Lai. 2016. A QoS-aware streaming service over fog computing infrastructures. In *Proceedings of 2016 Digital Media Industry & Academic Forum*. IEEE, Piscataway, NJ, 94-98. DOI:10.1109/DMIAF.2016.7574909

Sriram Lakshminarasimhan, David A. Boyuka, Saurabh V. Pendse, Xiaocheng Zou, John Jenkins, Venkatram Vishwanath, Michael E. Papka, and Nagiza F. Samatova. 2013. Scalable in-situ scientific data encoding for analytical query processing. In *Proceedings of the ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*. ACM, New York, NY, 1-12. DOI: 10.1145/2462902.2465527

Nikolaos Laoutaris, Pablo Rodriguez, and Laurent Massoulie. 2008. ECHOS: Edge capacity hosting overlays of nano data centers. *ACM SIGCOMM Computer Communication Review* 38, 1 (Jan. 2008), 51-54. DOI: 10.1145/1341431.1341442

Adrien Lebre, Anthony Simonet, Anne-Cécile Orgerie. 2016. Deploying distributed cloud infrastructures: who and at what cost. In *Proceedings of 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*. IEEE, Piscataway, NJ, 178-183. DOI: 10.1109/IC2EW.2016.48

Grace Lewis, Sebastián Echeverría, Soumya Simanta, Ben Bradshaw, James Root. 2014. Tactical cloudlets: Moving cloud computing to the edge. In *Proceedings of the 2014 IEEE Military Communications Conference*.

IEEE, Piscataway, NJ, 1440-1446. DOI: 10.1109/MILCOM.2014.238

Grace Lewis, Sebastian Echeverría, Soumya Simanta, Ben Bradshaw, James Root. 2014. Cloudlet-based cyber-foraging for mobile systems in resource-constrained edge environments. In *ICSE Companion 2014: Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, New York, NY, 412-415. DOI: 10.1145/2591062.2591119

Chao Li, Yang Hu, Longjun Liu, Juncheng Gu, Mingcong Song, Xiaoyao Liang, Jingling Yuan, and Tao Li. 2015. Towards sustainable in-situ server systems in the big data era. In *Proceedings of the 42nd Int. Symposium on Computer Architecture*. ACM, New York, NY, 14-26. DOI: *10.1145/2749469.2750381*

Kai Liang, Liqiang Zhao, Xiaoli Chu, and Hsiao-Hwa Chen. 2017. An integrated architecture for software defined and virtualized radio access networks with fog computing. *IEEE Network*, January/February (2017). IEEE, Piscataway, NJ, 80-87 DOI: 10.1109/MNET.2017.1600027NM

Yi Lin, Bettina Kemme, Marta Patino-Martinez, Ricardo Jimenez-Peris. 2007. Enhancing edge computing with database replication. In *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems*. IEEE, Piscataway, NJ, 45-54. DOI: 10.1109/SRDS.2007.10

Yuhua Lin and Haiying Shen. 2015. CloudFog: Towards high quality of experience in cloud gaming. In *Proceedings of the 44th International Conference on Parallel Processing*. IEEE, Piscataway, NJ, 500-509. DOI: 10.1109/ICPP.2015.59

Yuhua Lin and Haiying Shen. 2016. CloudFog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service. *IEEE Transactions on Parallel and Distributed Systems*, in press. DOI: 10.1109/TPDS.2016.2563428

Pedro G. Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, Etienne Riviere. 2015. Edge-centric computing: Vision and Challenges. *ACM SIGCOMM Computer Communication Review* 45, 5 (Oct. 2015), 37-45. DOI: 10.1145/2831347.2831354

Andrew Machen, Shiqing Wang, Kin K. Leung, Bong Jun Ko, Theodoros Salonidis. 2016. Migrating running applications across mobile edge clouds. In *Proceedings of the 22nd International Conference on Mobile Computing and Networking*. ACM, New York, NY, 435 - 436. DOI:10.1145/2973750.2985265

Redowan Mahmud and Rajkumar Buyya. 2016. Fog computing: A taxonomy, survey and future directions. arXiv:1611.05539v3 [cs.DC], 24 Nov 2016

Francesco Malandrino, Carla Chiasserini, Scott Kirkpatrick. 2016. The price of fog: A data-driven study on caching architectures in vehicular networks. In *Proceedings of the 1st Int. Workshop on Internet of Vehicles and Vehicles of Internet*. ACM, New York, NY, 37 - 42. DOI: 10.1145/2938681.2938682

G. Manjunath, T. Simunic, V. Krishnan. 2004. Smart edge server: Beyond a wireless access point. In *Proceedings of the 2nd International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*. ACM, New York, NY, 41 - 50. DOI: 10.1145/1024733.1024739

Xavi Masip-Bruin, Eva Marin-Tordera, Ghazal Tashakor, Admela Jukan, and Guang-Jie Ren. 2016. Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems. *IEEE Wireless Communications* 23, 5 (Nov. 2016), 120-128. DOI: 10.1109/MWC.2016.7721750

Emiliano Miluzzo, Ramón Cáceres, Yih-Farn Chen. 2012. Vision: mClouds – computing on clouds of mobile devices. In *Proceedings of the third ACM Workshop on Mobile Cloud Computing and Services*. ACM, New York, NY, 9-14. DOI: 10.1145/2307849.2307854

Ryan Newton, Sivan Toledo, Lewis Girod, Hari Balakrishnan, and Samuel Madden. 2009. Wishbone: Profile-based partitioning for sensornet applications. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, Berkley, CA, 395-408

Takayuki Nishio, Ryoichi Shinkuma, Tatsuro Takahashi, Narayan B. Mandayam. 2013. Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud. In *Proceedings of the 1st International Workshop on Mobile Cloud Computing and Networking*. ACM, New York, NY, 19-26. DOI: 10.1145/2492348.2492354

Anne-Cecile Orgerie, Marcos Dias De Assuncao, and Laurent Lefevre. 2014. A survey on techniques for improving the energy efficiency of large-scale distributed systems. ACM Computing Surveys: Vol 46, Issue 4. ACM, New York, NY, 69-80. DOI: 10.1145/2532637

Gabriel Orsini, Dirk Bade, and Winfried Lamersdorf. 2015. Computing at the mobile edge: Designing elastic android applications for computation offloading. In *Proceedings of the 8th IFIP Wireless and Mobile Networking Conference*. IEEE, Piscataway, NJ, 112-119. DOI: 10.1109/WMNC.2015.10

Opeyemi Osanaiye, Shuo Chen, Zheng Yan, Rongxing Lu, Kim-Kwang Raymond Choo, and Mqhele Dlodlo. 2017. From cloud to fog computing: A review and a conceptual live VM migration framework. *IEEE Access*, Vol 5 (2017). IEEE, Piscataway, NJ, 8284-8300. DOI: 10.1109/ACCESS.2017.2692960

Beate Ottenwälder, Ruben Mayer, Boris Koldehofe. 2014. Distributed complex event processing for mobile large-scale video applications. In *Middleware Posters and Demos*. ACM, New York, NY, 5-6. DOI:10.1145/2678508.2678511

Jessica Oueis, Emilio C. Strinati, and Sergio Barbarossa. 2015. The fog balancing: Load distribution for small cell cloud computing. In *Proceedings of IEEE 81st Vehicular Technology Conference*. IEEE, Piscataway, NJ, 1 – 6. DOI: 10.1109/VTCSpring.2015.7146129

Claus Pahl and Brian Lee. 2015. Containers and clusters for edge cloud architectures – A technology review. In *Proceedings of the 3rd International Conference on Future Internet of Things and Cloud*. IEEE, Piscataway, NJ, 379-386. DOI:10.1109/FiCloud.2015.35

Claus Pahl, Sven Helmer, Lorenzo Miori, Julian Sanin, and Brian Lee. 2016. A container-based edge cloud PaaS

architecture based on Raspberry Pi clusters. In *Proceedings of the 4th Int. Conf. on Future Internet of Things and Cloud*. IEEE, Piscataway, NJ, 117-124. DOI: 10.1109/W-FiCloud.2016.36

Andreas Pamboris and Peter Pietzuch. 2016. C-RAM: Breaking mobile device memory barriers using the cloud. *IEEE Transactions on Mobile Computing* 15, 11 (Nov. 2016), 2693-2705. DOI:10.1109/TMC.2015.2513040

Apostolos Papageorgiou, Bin Cheng, and Ernö Kovacs. 2015. Real-time data reduction at the network edge of Internet-of-Things Systems. In *Proceedings of the 11th Int. Conference on Network and Service Management*. IEEE, Piscataway, NJ, 284-291. DOI: 10.1109/CNSM.2015.7367373

Seok-Hwan Park, Osvaldo Simeone, and Shlomo Shamai Shitz. 2016. Joint optimization of cloud and edge processing for fog radio access networks. *IEEE Trans. on Wireless Communications* 15, 11 (2016), 7621-7632. DOI: 0.1109/TWC.2016.2605104

PNNL. 2013. Edge Computing. http://vis.pnnl.gov/pdf/fliers/EdgeComputing.pdf

Lakshmish Ramaswamy, Ling Liu, and Arun Iyengar. 2007. Scalable delivery of dynamic content using a cooperative edge cache grid. *IEEE Trans. On Knowledge and Data Engineering* 19, 5 (May 2007), 614-630. DOI: 10.1109/TKDE.2007.1019.

Andreas Reiter, Bernd Prunster, and Thomas Zefferer. 2017. Hybrid mobile edge computing: Unleashing the full potential of edge computing in mobile device use cases. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. ACM, New York, NY, 935-94. DOI: 10.1109/CCGRID.2017.125

Rodrigo Roman, Javier Lopez, Masahiro Mambo. 2016. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, in press. DOI: 10.1016/j.future.2016.11.009

Paolo Romano and Francesco Quaglia. 2014. Design and evaluation of a parallel invocation protocol for transactional applications over the Web. *IEEE Trans. on Computers* 63, 2 (Feb. 2014), 317-334. DOI: 10.1109/TC.2013.214.

Sean Rooney, Daniel Bauer, and Paolo Scotton. 2015. Edge server software architecture for sensor applications. In *Proceedings of the 2005 Symposium on Applications and the Internet*. IEEE, Piscataway, NJ, 64-71. DOI:10.1109/SAINT.2005.24

Mathew Ryden, Kwangsung Oh, Abhishek Chandra, and Jon Weissman. 2014. Nebula: Distributed edge cloud for data intensive computing. In *Proceedings of 2014 IEEE International Conference on Cloud Engineering*. IEEE, Piscataway, NJ, 57-66. DOI:10.1109/IC2E.2014.34

Alexey Rudenko, Peter Reiher, Gerald J. Popek. 1998. Saving portable computer battery power through remote process execution. *ACM SIGMOBILE Mobile Computing and Communications Review* 2, 1 (Jan. 1998), 19-26. DOI: 10.1145/584007.584008

S. Sardellitti, S. Barbarossa, G. Scutari. 2014. Distributed mobile cloud computing: Joint optimization of radio and computational resources. In *Proceedings of the Globecom Workshops*. IEEE, Piscataway, NJ, 1505-1510. DOI: 10.1109/GLOCOMW.2014.7063647

Mahadev Satyanarayanan. 2001. Pervasive computing: Vision and challenges. *IEEE Personal Communications* 8, 4 (Aug. 2011), 10-17. DOI:10.1109/98.943998

Mahadev Satyanarayanan, Paramvir Bahl, Ramón Cáceres, Nigel Davies. 2009. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing* 9, 8 (Oct-Dec. 2009), 14-23. DOI:10.1109/MPRV.2009.82

Enrique Saurez, Kirak Hong, Dave Lillethun, Umakishore Ramachandran, and Beate Ottenwälder. 2016. Incremental deployment and migration of geo-distributed situation awareness application in the fog. In *Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems*. ACM, New York, NY, 258-269. DOI:10.1145/2933267.2933317

Petri Savolainen, Sumi Helal, Jukka Reitmaa, Kai Kuikkaniemi, Giulio Jacucci, Mikko Rinne, Marko Turpeinen, and Sasu Tarkoma. 2013. Spaceify – a client-edge-server ecosystem for mobile computing in smart spaces. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking*. ACM, New York, NY, 211- 213. DOI:10.1145/2500423.2504578.

Fareha Sheikh, Habiba Fazal, Fatima Taqvi, and Jawwad Shamsi. 2015. Power-aware server selection in nano data center. In *Proceedings of the 40th Local Computer Networks Conference Workshops*. IEEE, Piscataway, NJ, 776-782. DOI: 10.1109/LCNW.2015.7365927

Cong Shi, Vasileios Lakafosis, Mostafa H. Ammar, Ellen W. Zegura. 2012. Serendipity: enabling remote computing among intermittently connected mobile devices. In *Proceedings of the 13th ACM International Symposium on Mobile Ad Hoc Netwkroing and Computing*. ACM, New York, NY, 211- 213. DOI: 10.1145/2248371.2248394

Heng Shi, Nan Chen, and Ralph Deters. 2015. Combining mobile & fog computing. In *Proceedings of the 2015 IEEE International Conference on Data Science and Data Intensive Systems*. IEEE, Piscataway, NJ, 564-571. DOI: 10.1109/DSDIS.2015.115

Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet of Things J.* 3, 5 (Oct 2016), 637-646. DOI: 10.1109/JIOT.2016.2579198

Sejun Song, Baek-Young Choi, and Daehee Kim. 2016. Selective encryption and component-oriented deduplication for mobile cloud data computing. In *Proceedings of the 2016 Int. Conf. on Computing, Networking and Communications*. IEEE, Piscataway, NJ, 1-5. DOI: 10.1109/ICCNC.2016.7440636

V.B.C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, G. Tashakor. 2016. Handling service allocation in combined fog-cloud scenarios. In *Proceedings of the 2016 IEEE International Conference on Communications*. IEEE, Piscataway, NJ, 1-5. DOI:10.1109/ICC.2016.7511465

Ivan Stojmenovic and Sheng Wen. 2014. The fog computing paradigm: Scenarios and security issues. In *Proceedings of the 2014 Federated Conf. on Computer Science and Information Systems*. IEEE, Piscataway, NJ, 1-8. DOI: 10.15439/2014F503

Phillip Stanley-Marbell. 2011. Parallelism, performance, and energy-efficiency tradeoffs for in-situ sensor data processing. *IEEE Embedded Sys. Lett. 3, 1* (Mar. 2011), 16-19. DOI:10.1109/LES.2010.2092412

Moritz Steiner, Bob Gaglianello, Vijay Gurbani, Volker Hilt, W.D. Roome, Michael Scharf, and Thomas Voith. 2012. Network-aware service placement in a distributed cloud environment. *ACM SIGCOMM Computer Communication Review* 42, 4 (Oct. 2012), 73-74. DOI: 10.1145/2377677.2377687

W. Steiner and S. Poledna. 2016. Fog computing as enabler for the industrial Internet of Things. Elektrotechnik und Informationstechnik 133, 7 (Nov. 2016), 310–314. DOI 10.1007/s00502-016-0438-2

Pradeep Sundarrajan, Abhishek Gupta, Matthew Ryden, Rohit Nair, Abhishek Chandra, and Jon Weissman. 2011. Early experience with the distributed nebula cloud. In *Proceedings of the fourth International Workshop on Data-Intensive Distributed Computing*. ACM, New York, NY, 17-26. DOI: 10.1145/1996014.1996019

Ya-Sunn Su and Jason Flinn. 2015. Slingshot: Deploying stateful services in wireless hotspots. In *Proceedings of the third International conference on Mobile Systems, Applications, and Services*. USENIX Association, Berkley, CA, 79-92

Bo Tang, Zhen Chen, Gerald Hefferman, Tao Wei, Haibo He, Qing Yang. 2015. A hierarchical distributed fog computing architecture for big data analysis in smart cities. In *Proceedings of the ASE Big Data & Social Informatics*, Article No. 28. DOI: 10.1145/2818869.2818898

Fikirte Teka, Chung-Horng Lung, and Samuel A. Ajila. 2016. Nearby live virtual machine migration using cloudlets and multipath TCP. *Journal of Cloud Computing: Advances, Systems and Applications* 5, 12, 21 pages. Springer Open. DOI: 10.1186/s13677-016-0061-0

Liang Tong, Yong Li, and Wei Gao. 2016. A hierarchical edge cloud architecture for mobile computing. In *Proceedings of the 35th IEEE International Conference on Computer Communications*. IEEE, Piscataway, NJ, 1-9. DOI: 10.1109/INFOCOM.2016.7524340

Nguyen B. Truong, Gyu Myoung Lee, Yacine Ghamri-Doudane. Software defined networking-based vehicular adhoc network with fog computing. In *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, Piscataway, NJ, 1202-1207. DOI: 10.1109/INM.2015.7140467

Rahul Urgaonkar, Shiqiang Wang, Ting He, Murtaza Zafer, Kevin Chan, and Kin K. Leung. 2015. Dynamic service migration and workload scheduling in edge-clouds. *Performance Evaluation*, 91 (July 2015). DOI: 10.1016/j.peva.2015.06.013

Steffen Viken Valvåg, Dag Johansen, Åge Kvalnes. 2013. Position paper: elastic processing and storage at the edge of the cloud. In *Proceedings of the 2013 International Workshop on Hot Topics in Cloud Services*. ACM, New York, NY, 43-50. DOI: 10.1145/2462307.2462317

Luis M. Vaquero and Luis Rodero-Merino. 2014. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review* 44, 5 (Oct. 2014), 205-228. DOI: 10.1145/2677046.2677052

Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. 2012. Cloudlets: Bringing the cloud to the mobile user. In *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*. ACM, New York, NY, 257-262. DOI: 10.1145/2307849.2307858

Hariharasudhan Viswanathan, Eun Kyung Lee, and Dario Pompili. 2013. Enabling real-time in-situ processing of ubiquitous mobile-application workflows. In *Proceedings of the 10th International Conference on Mobile Ad-Hoc and Sensor Systems*. IEEE, Piscataway, NJ, 324 – 332. DOI: 10.1109/MASS.2013.86

Hariharasudhan Viswanathan, Parul Pandey, and Dario Pompili. 2016. Maestro: Orchestrating concurrent application workflows in mobile device clouds. In *Proceedings of the 2016 IEEE Int. Conf. on Autonomic Computing*. IEEE, Piscataway, NJ, 257 – 262. DOI: 10.1109/ICAC.2016.44

Aosen Wang, Lizhong Chen, and Wenyao Xu. 2017. XPro: A cross-end processing architecture for data analytics in wearables. In *Proceedings of the 44th International Symposium on Computer Architecture*. ACM, New York, NY, 69-80. DOI: 10.1145/3079856.3080219

Chuo Wang, Xing Zhang, Yan Zhang, Lin Wang, Juwo Yang, and Wenbo Wang. 2017. A survey on mobile edge networks: convergence of computing, caching and communications. *IEEE Access*, Vol 5 (2017). IEEE, Piscataway, NJ, 6757-6779. DOI: 10.1109/ACCESS.2017.2685434

Shiqiang Wang, Rahul Urgaonkar, Murtaza Zafer, Ting He, Kevin Chan, and Kin K. Leung. 2015a. Dynamic service migration in mobile edge-clouds. In *Proceedings of the 2015 IFIP Networking Conf*. IEEE, Piscataway, NJ, 1-9. DOI:10.1109/IFIPNetworking.2015.7145316

Shiqiang Wang, Kevin Chan, Rahul Urgaonkar, Ting He, and Kin K. Leung. 2015b. Emulation-based study of dynamic service placement in mobile micro-clouds. In *Proceedings of the 2015 IEEE Military Communications Conference*. IEEE, Piscataway, NJ, 1046-1051. DOI: 10.1109/MILCOM.2015.7357583

Wei Wang, Qin Wang, and Kazem Sohraby. 2016. Multimedia sensing as a service (MSaaS): Exploring resource saving potentials of at cloud-edge IoTs and Fogs. *IEEE Internet of Things Journal*, in press. DOI: 10.1109/JIOT.2016.2578722

Yifan Wang, Tetsutaro Uehara, and Ryoichi Sasaki. 2015. Fog computing: Issues and challenges in security and forensics. In *Proceedings of the 39th Annual Int. Computers, Software & Applications Conf*. IEEE, Piscataway, NJ, 53-59. DOI: 10.1109/COMPSAC.2015.173

Yonggang Wen, Weiwen Zhang, Haiyun Luo. 2012. Energy-optimal mobile application execution: Taming resources-poor mobile devices with cloud clones. In *Proceedings of the 31st IEEE Int. Conf. on Computer*

Communications. IEEE, Piscataway, NJ, 2716-2720. DOI: 10.1109/INFCOM.2012.6195685

Dale Willis, Arkodeb Dasgupta, Suman Banerjee. 2014. ParaDrop: A multi-tenant platform to dynamically install third party services on wireless gateways. In *Proceedings of the 9th ACM Workshop on Mobility in the Evolving Internet Architecture*. ACM, New York, NY, 43-48. DOI: 10.1145/2645892.2645901

Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, and Hui-Ying Du. 2010. Research on the architecture of Internet of Things. In *Proceedings of the 3rd Int. Conf. on Advanced Computer Theory and Engineering*. IEEE, Piscataway, NJ, 484-487. DOI: 10.1109/ICACTE.2010.5579493

Changjiu Xian, Yung-Hsiang Lu, Zhiyuan Li. 2007. Adaptive computation offloading for energy conservation on battery-powered systems. In *Proceedings of the 2007 International Conference on Parallel and Distributed Systems*. IEEE, Piscataway, NJ, 1-8. DOI: 10.1109/ICPADS.2007.4447724

Xing Xie, Hua-Jun Zeng, Wei-Ying Ma. 2002. Enabling personalization services on the edge. In *Proceedings of the Tenth ACM International Conference on Multimedia*. ACM, New York, NY, 263-266. DOI: 10.1145/641007.641060

Yi Xu, Sumi Helal, My T. Thai, Mark Schmalz. 2011. Optimizing push/pull envelopes for energy-efficient cloud-sensor systems. In *Proceedings of the 14th ACM Int. Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, New York, NY, 17-26. DOI: 10.1145/2068897.2068904

Yi Xu and Sumi Helal. 2014. Application caching for cloud-sensor systems. In *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, New York, NY, 303-306. DOI: 10.1145/2641798.2641814

Sami Yangui, Pradeep Ravindran, Ons Bibani, Roch H. Glitho, Nejib Ben Hadj-Alouane, Monique J. Morrow, and Paul A. Polakos. 2016. A platform as-a-service for hybrid cloud/fog environments. In *Proceedings of the International Symposium on Local and Metropolitan Area Networks*. IEEE, Piscataway, NJ, 1-7. DOI: 10.1109/LANMAN.2016.7548853

Abdulsalam Yassine, Ali Asghar Nazari Shirehjini, and Shervin Shirmohammadi. 2016. Bandwidth on-demand for multimedia big data transfer across geo-distributed cloud data centers. *IEEE Trans. on Cloud Computing*, in press. DOI: 10.1109/TCC.2016.2617369

Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. 2015a. Fog computing: Platform and Applications. In *Proceedings of the Third IEEE Workshop on Hot Topics in Web Systems and Techniques*. IEEE, Piscataway, NJ, 73-78. DOI: 10.1109/HotWeb.2015.22

Shanhe Yi, Cheng Li, and Qun Li. 2015b. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*. ACM, New York, NY, 37-42. DOI: 10.1145/2757384.2757397

Hao Yin, Xu Zhang, Hongqiang Harry Liu, Yan Luo, Chen Tian, Shuoyao Zhao, Feng Li. Edge provisioning with flexible server placement. *IEEE Trans. on Parallel and Distributed Systems*, in press. DOI: 10.1109/TPDS.2016.2604803

Deze Zeng, Lin Gu, Song Guo, Zixue Cheng, and Shui Yu. 2013. *IEEE Trans. on Computers* 65, 12 (Feb. 2016). IEEE, Piscataway, NJ, 3702 – 3712. DOI: 10.1109/TC.2016.2536019

Engin Zeydan, Ejder Bastug, Mehdi Bennis, Manhal Abdel Kader, Ilyas Alper Karatepe, Ahmet Salih Er, and Mérouane Debbah. 2016. Big data caching for networking: Moving form cloud to edge. *IEEE Communications Magazine* 54, 9 (Sep. 2016), 36-42. DOI: 10.1109/MCOM.2016.7565185

Fan Zhang, Solomon Lasluisa, Tong Jin, Ivan Rodero, Hoang Bui, and Manish Parashar. 2012. In-situ feature-based objects tracking for large-scale scientific simulations. In *Proceedings of the SC Companion: High Performance Computing, Networking Storage and Analysis*. IEEE, Piscataway, NJ, 736-740. DOI: 10.1109/SC.Companion.2012.100

Tan Zhang, Aakanksha Chowdhery, Paramvir Bahl, Kyle Jamieson, Suman Banerjee. 2015. The design and implementation of a wireless video surveillance system. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, New York, NY, 426-438. DOI: 10.1145/2789168.2790123

Jiang Zhu, Douglas S. Chan, Mythili Suryanarayana Prabhu, Preethi Natarajan, Hao Hu and Flavio Bonomi. 2013. In *Proceedings of the Seventh International Symposium on Service-Oriented System Engineering*. IEEE, Piscataway, NJ, 320-323. DOI: 10.1109/SOSE.2013.73