A Survey on Edge Performance Benchmarking

BLESSON VARGHESE, Queen's University Belfast, UK
NAN WANG, Durham University, UK
DAVID BERMBACH, TU Berlin and ECDF, Germany
CHEOL-HO HONG, Chung-Ang University, South Korea
EYAL DE LARA, University of Toronto, Canada
WEISONG SHI, Wayne State University, USA
CHRISTOPHER STEWART, Ohio State University, USA

Edge computing is the next Internet frontier that will leverage computing resources located near users, sensors, and data stores to provide more responsive services. Therefore, it is envisioned that a large-scale, geographically dispersed, and resource-rich distributed system will emerge and play a key role in the future Internet. However, given the loosely coupled nature of such complex systems, their operational conditions are expected to change significantly over time. In this context, the performance characteristics of such systems will need to be captured rapidly, which is referred to as performance benchmarking, for application deployment, resource orchestration, and adaptive decision-making. *Edge performance benchmarking* is a nascent research avenue that has started gaining momentum over the past five years. This article first reviews articles published over the past three decades to trace the history of performance benchmarking from tightly coupled to loosely coupled systems. It then systematically classifies previous research to identify the system under test, techniques analyzed, and benchmark runtime in edge performance benchmarking.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Computer systems organization** → **Cloud computing**;

Additional Key Words and Phrases: Edge computing, edge performance benchmarking, system under test, techniques analyzed, benchmark runtime

The first author is supported by funds from Rakuten Mobile, Japan and by a Royal Society Short Industry Fellowship. The corresponding author is supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2019R1C1C1011068).
Authors' addresses: B. Varghese, Electrical Engineering and Computer Science, School of Electronics, Queen's University Belfast, University Road, Belfast, BT7 1NN, UK; email: b.varghese@qub.ac.uk; N. Wang, Department of Computer Science, Durham University, Stockton Road, Durham, DH1 3LE, UK; email: nan.wang@durham.ac.uk; D. Bermbach, Mobile Cloud Computing Research Group, TU Berlin and ECDF, Straße des 17. Juni 135, 10623, Berlin, Germany; email: david.bermbach@tu-berlin.de; C.-H. Hong (corresponding author), School of Electrical and Electronics Engineering, Chung-Ang University, 84 Heukseok-ro, Dongjak-gu, Seoul, 06974, South Kore; email: cheolhohong@cau.ac.kr; E. de Lara, Department of Computer Science, University of Toronto, 27 King's College Circle, Toronto, Ontario M5S 1A1, Canada; email: delara@cs.toronto.edu; W. Shi, Department of Computer Science, Wayne State University, 42 W. Warren Ave. Detroit, MI 48202, USA; email: weisong@wayne.edu; C. Stewart, Department of Computer Science and Engineering, Ohio State University, 281 W Lane Ave, Columbus, OH 43210, USA; email: cstewart@cse.ohio-state.edu.

0360-0300/2021/04-ART66 $15.00
https://doi.org/10.1145/3444692

ACM Reference format:
Blesson Varghese, Nan Wang, David Bermbach, Cheol-Ho Hong, Eyal de Lara, Weisong Shi, and Christopher Stewart. 2021. A Survey on Edge Performance Benchmarking. *ACM Comput. Surv.* 54, 3, Article 66 (April 2021), 33 pages.
https://doi.org/10.1145/3444692

1 INTRODUCTION

The computing landscape has changed significantly over the past three decades. Loosely coupled and geographically dispersed systems have begun replacing tightly coupled monolithic systems [146]. One example is from two decades ago, when computing resources that were distributed across numerous organizations and continents were connected under the umbrella of grid computing. Grids offered the unique capability of processing large datasets near data sources without requiring the transfer of data from a distributed workflow to a central system [49]. Subsequently, computing became a utility offered remotely through the cloud [147].

Although the cloud is the main computing model adopted for many Internet-based applications, it has been recognized as an untenable model for the future. This is because billions of devices and sensors are connected to the Internet, and the data generated by these sources cannot be transferred and processed in geographically distant cloud data centers without incurring considerable communication delays [150]. Therefore, the next disruption in the computing landscape is to distribute infrastructure resources and application services further, to bring computing closer to the edge of the network and data sources [114, 132]. In this article, we use the term "edge computing" to refer to the use of resources located at the edge of a network, such as routers and gateways or dedicated micro data centers, to either provide applications with acceleration by co-hosting services in cooperation with the cloud or by hosting them natively or entirely on edge resources [125].

The inclusion of edge resources for computing creates a large-scale, geographically dispersed, resource-rich distributed system that spans multiple technological domains and ownership boundaries. Such a complex system will be transient, meaning that resources, their availability, and characteristics will change over time. For example, an edge resource previously available for an application may become unavailable based on a recent fault or because the operating system of a target resource may change during maintenance [154, 155]. In this context, it is essential to address the challenge of understanding the relative performance of applications by comparing diverse target hardware platforms from different vendors and their impact on performance when system software changes or new networking protocols are introduced [102]. This has motivated the development of *edge performance benchmarking*[1] [13, 67].

Performance benchmarking is the process of inducing stress on a system while closely observing its responses using a wide range of quality metrics. Typically, synthetic or application-driven workloads are executed on a system under test, such as a virtual machine, a storage system, a stream processing system, or a specific application component, while measuring quality characteristics, such as I/O throughput, end-to-end communication, or computation latency. Unlike alternative approaches such as predictive methods or simulation, insights into real system behaviors can be obtained by replicating the conditions of a production environment [19].

To the best of our knowledge to date, a survey on edge performance benchmarking is not available in the literature. Hence, this article focuses on the following aspects: (i) Tracing the history of the development of performance benchmarking over the past three decades for

[1]This article will use the terms "performance benchmarking" and "benchmarking" interchangeably. However, the focus of this article is on edge performance benchmarking.

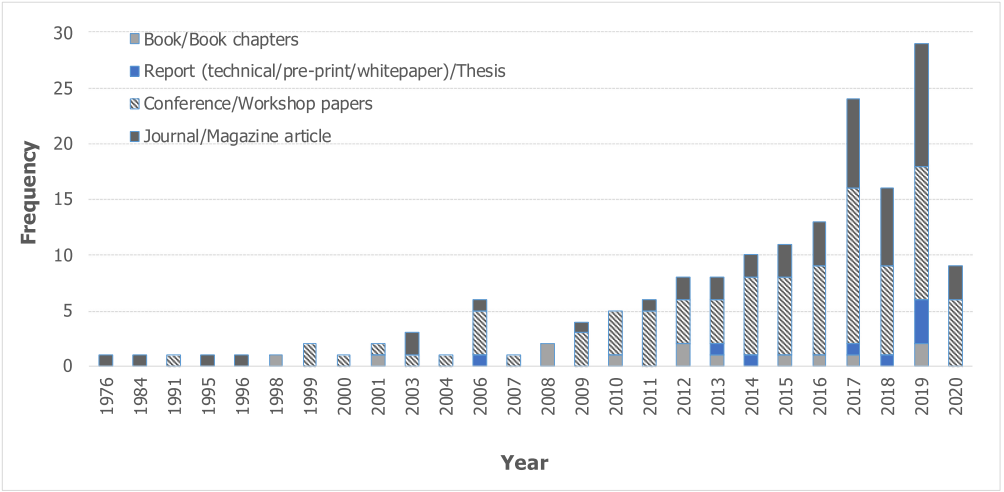

Fig. 1. Histogram of the research publications reviewed in this article.

high-performance computing (HPC), grid, and cloud systems; (ii) cataloging and examining different edge performance benchmarks; and (iii) reviewing the system under test, techniques analyzed, and benchmark runtime that facilitate edge performance benchmarking.

Figure 1 presents a histogram of the total number of research publications reviewed in this article from 1976 and 2020 in the following categories: (i) books and book chapters; (ii) reports, including preprint articles or white papers and doctoral research theses; (iii) conference or workshop papers; and (iv) journal or magazine articles. More than 83% of the articles reviewed were published after 2010 and more than 61% of the articles were published after 2015.

1.1 Survey Method

The survey method adopted for preparing this article is based on an approach presented in a previous survey article [46]. This method includes (1) defining the objectives of the survey, (2) defining research questions, (3) selecting keywords for searching, and (4) identifying criteria for including or excluding research. These aspects are defined below.

(1) The *objectives* of this survey are defined as follows:

O1 Provide the research community with a catalog of research related to edge performance benchmarking.
O2 Trace the development timeline of performance benchmarking for edge systems.
O3 Understand the key dimensions of existing research on edge performance benchmarking.
O4 Discuss directions for future research to extend the application of edge performance benchmarking.

(2) The *research questions* addressed by this survey are defined as follows:

RQ1 To which systems is edge performance benchmarking applied? This will be discussed in Section 4.
RQ2 Which techniques are analyzed by edge performance benchmarking? This will be discussed in Section 5.
RQ3 What are the runtime environments that edge performance benchmarks operate in? This will be discussed in Section 6.

(3) Publication platforms such as the ACM Digital Library, IEEEXplore, ScienceDirect, arXiv, and Google Scholar were considered. The primary *keywords* were a combination of edge (fog, mobile edge, cloud-edge, cloudlet) and benchmarking (benchmark, benchmark suite, micro benchmark, and macro benchmark) with additional keywords such as performance, system, and distributed systems.

(4) The resulting works were screened to identify the most relevant works. A total of 3,764 publications were considered and screened down to 689 publications. The initial filters applied were based on the title, followed by the relevance of the abstract. Two types of performance benchmarking works are available, namely explicit and implicit edge performance benchmarking. For research to be selected as explicit performance benchmarking, a benchmarking method, specific benchmark, or toolchain for facilitating performance benchmarking had to be presented, which was determined by inspecting complete papers. Such papers generally contribute to the field of edge benchmarking. We selected 21 works containing explicit performance benchmarks, which are cataloged in Section 3.

Implicit performance edge benchmarking works were also included if they presented an evaluation of the performance of a system under testing, technique developed, or runtime, even though such works did not explicitly highlight a benchmark, benchmarking methodology, or benchmarking suite. Studies that did not present a comparative analysis of the system under test, technique developed, or runtime were not considered for implicit benchmarking. The selected works often use novel workloads and measurement approaches that could potentially be incorporated into explicit edge benchmarking research. A total of 99 implicit performance edge benchmarking publications were selected. Selections based on the above mentioned criteria were validated by at least two of the authors of this article.

This article considers a wide range of papers and Internet sources related to edge performance benchmarking. For this purpose, we focus on explicit and implicit edge performance benchmarks. We do not claim completeness for our selected set of implicit benchmarks for the following two reasons. First, the body of work on edge computing (including closely related topics such as fog computing or mobile edge computing) is very large and cannot be considered within a single survey paper. Second, explicit edge benchmarks can be identified objectively, whereas implicit benchmarks are subjective. Based on careful analysis and validation by at least two authors, the set of selected implicit benchmarks may not necessarily be complete, but it contains no false positives. Therefore, the subset of implicit benchmarks considered within the scope of this article adds value to this survey and to the emerging field of edge performance benchmarking.

The remainder of this article is organized as follows. Section 2 provides a brief history of performance benchmarking. Section 3 catalogs different edge performance benchmarks. Section 4 presents a review of systems under testing in edge performance benchmarking. Section 5 reviews the techniques analyzed in edge performance benchmarking. Section 6 surveys runtime execution environments and deployments in edge performance benchmarks. Section 7 discusses future directions for additional research and concludes this article.

2 A BRIEF TIMELINE OF PERFORMANCE BENCHMARKING

Performance benchmarks have played an important role in the evolution of the computing landscape and represent an important field of research and development. CPU or processor-related benchmarks have existed since the 1970s. For example, the Whetstone benchmark was developed to measure the floating-point arithmetic performance [36]. Dhrystone is another benchmark developed in the 1980s to evaluate the performance of integers [157].

This section provides a brief history of the development of the field of benchmarking of distributed systems over the past three decades, as shown in Figure 2. Specifically, both tightly

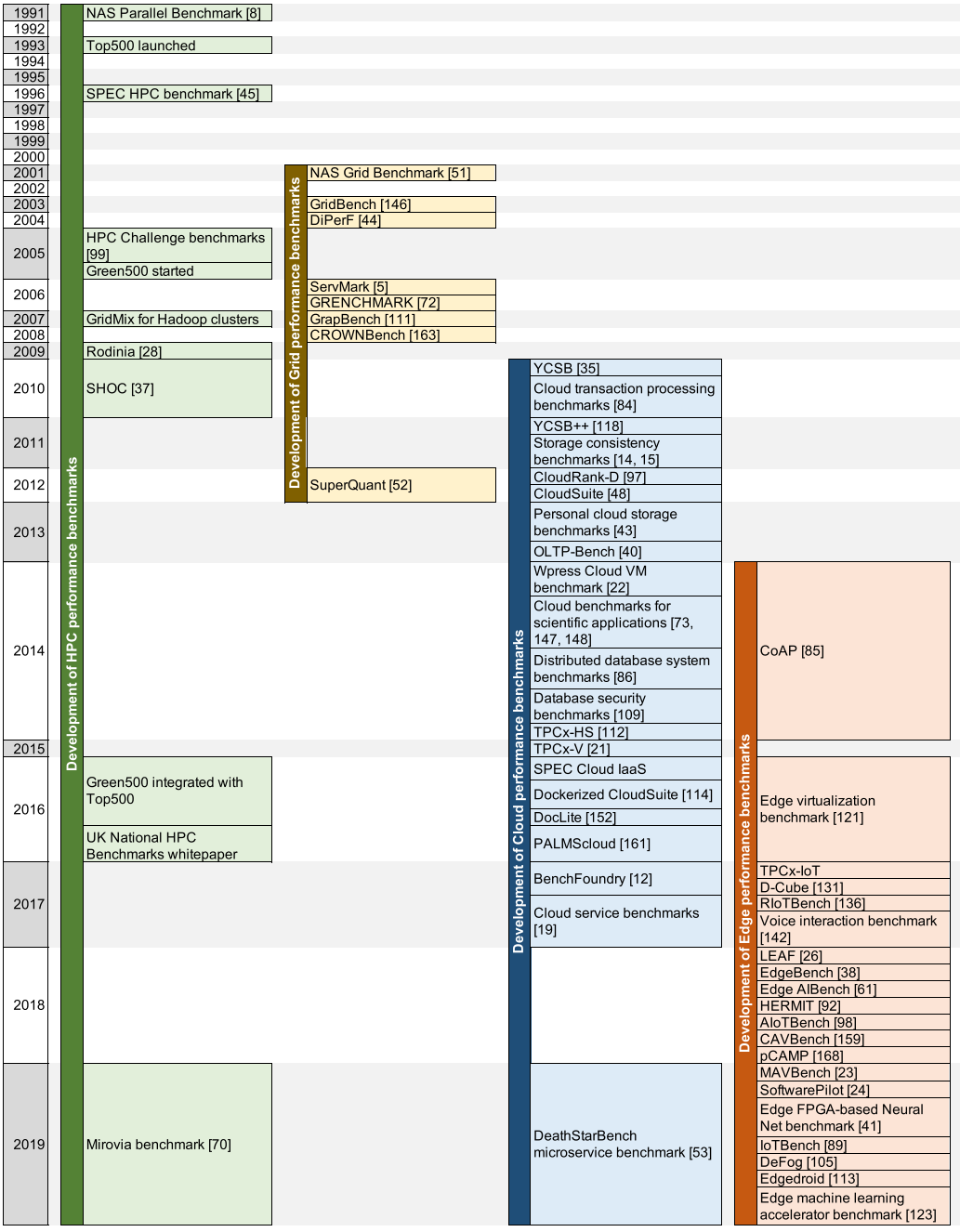

Fig. 2. Brief history of the development of performance benchmarking for HPC, grid, cloud, and edge systems.

coupled HPC clusters and supercomputers (highlighted in green), as well as more loosely coupled infrastructure, such as the grid (highlighted in bronze) and the cloud (highlighted in turquoise), are considered. The next section will consider edge computing benchmarks (highlighted in red color), which are the main focus of this article.

This article does not present an exhaustive timeline of performance benchmarking within computer science, but highlights some of the key milestones that have shaped performance benchmarking research for current distributed systems and more specifically edge computing, which is discussed in Section 3. In general, the observed pattern is that post-1990 HPC benchmarking, post-2000 grid benchmarking, post-2010 cloud benchmarking, and post-2015 edge benchmarking achieved major milestones. This trend seems to suggest that benchmarks for more loosely coupled distributed systems are gaining prominence as a result of the decentralization and distribution of resources within the computing landscape.

2.1 HPC Benchmarking

In 1979, the LINPACK benchmark was developed. This benchmark eventually evolved into the **High-Performance LINPACK (HPL)** benchmark [42]. This benchmark is computationally intensive and measures the floating point rate of execution by solving a dense system of linear equations. HPL is a de facto benchmark used for capturing the performance of supercomputers and clusters in the TOP500[2] list launched in 1993.

The **NAS Parallel Benchmarks (NPB)** were launched in 1991 and are based on **computational fluid dynamics (CFD)** applications [8]. The **Standard Performance Evaluation Corporation (SPEC)** launched several HPC benchmarks in 1996 [45]. They focused on three industrial applications, namely seismic processing, computational chemistry, and climate modeling.

More than a decade after the TOP500 project was launched, energy efficiency became an important metric that influenced the development of parallel computing systems [130]. This resulted in the launch of the Green500[3] project in 2005, which evaluated floating point rates of execution in the context of power consumption. A methodology for measuring and reporting the power used by an HPC system was developed.[4] In 2016, Green500 was integrated with the TOP500 project.

The HPC Challenge benchmark was launched in 2005 to measure performance and productivity [97]. This benchmark includes the STREAM benchmark for measuring sustainable memory bandwidth [101].

The field commonly known as big data became popular in the context of HPC clusters, and hence the GridMix[5] benchmark for Hadoop clusters emerged in 2007.

The High Performance Conjugate Gradient is a relatively new benchmark that was launched in 2013 to rank supercomputers and clusters in a more balanced manner [66]. The performance of this benchmark is influenced by memory bandwidth.

In 2016, several benchmarks utilized on the UK's national supercomputer ARCHER were presented.[6] These benchmarks are a combination of real applications (DFT, molecular mechanics-based, CFD, and climate modeling) developed by the UK Met Office and synthetic benchmarks from the HPC Challenge.

[2] https://www.TOP500.org/.
[3] https://www.TOP500.org/green500/.
[4] https://www.TOP500.org/static/media/uploads/methodology-2.0rc1.pdf.
[5] https://hadoop.apache.org/docs/r1.2.1/gridmix.html.
[6] https://www.archer.ac.uk/documentation/white-papers/benchmarks/UK_National_HPC_Benchmarks.pdf.

Because HPC has become more heterogeneous with the advent of hardware accelerators, such as GPUs, novel benchmarks have begun to emerge. These benchmarks include the RODINIA [28] and SHOC GPU benchmarks [37], as well as the more recent Mirovia benchmarks [70].

2.2 Grid Benchmarking

As academic and research organizations have begun connecting clusters of computers, geographically dispersed and heterogeneous grids have become popular for scientific computing. Key metrics that are relevant to grid benchmarking included turnaround time and throughput, because data originated from different geographic locations in a scientific workflow is executed on grids [136].

One of the first benchmarks for evaluating computing performance on grids was the GridNPB, which was released in 2001. This benchmark is an NPB distribution that uses the Globus grid middleware [51].

In 2003, GridBench, which allows benchmarks to be specified using a definition language that is compiled into specification languages supported by grid middleware, was released [143].

In 2004, the DiPerF benchmarking tool was used to generate performance statistics for grids to predict grid performance [44].

GRENCHMARK was developed for generating synthetic workloads for benchmarking grids [72]. This approach was adopted in ServMark for automating benchmarking pipelines [5].

Additional benchmarks such as GrapBench have provided flexibility to benchmarking approaches by considering variations in problems and machine sizes of applications and the grid, respectively [108]. Furthermore, CROWNBench generates synthetic workloads for benchmarking [160]. Domain-specific benchmarks, such as those relevant to financial workloads, have also been developed [52].

2.3 Cloud Benchmarking

Benchmarking or web-based systems, including open, closed, and partially closed systems, have also been considered [127]. One of the earliest benchmarks for web servers was SPECweb96.[7] In 2010, the first cloud-specific benchmarks were released, namely the **Yahoo! Cloud Service Benchmark (YCSB)** [35] and transaction processing benchmarks [82]. YCSB++ is an extension of the YCSB that benchmarks scalable column stores [115]. Various approaches for benchmarking scientific applications in the cloud [73, 144, 145] and **virtual machines (VMs)** [22] have been proposed. Moreover, the aspect of fairness in benchmarking has been considered [54]. Several benchmarks for the consistency of cloud storage services have been developed [14, 15].

A number of cloud performance benchmarks were developed between 2012 and 2015. Two benchmarks developed by the **Transaction Processing Performance Council (TPC)** are particularly noteworthy. The first is the TPCx-V [21] benchmark, which is a virtual machine benchmark for database workloads. The second is the TPCx-HS [109] benchmark, which is a big data benchmark on the cloud. Other benchmarks include CloudRank-D [95] and CloudSuite [48] for resources and services. Benchmarks for cloud storage [43, 85], relational databases [40], database security [106], and the scalability and elasticity of distributed databases have also been developed [84].

The SPEC benchmarks for infrastructure-as-a-service clouds were launched in 2016 (SPEC Cloud IaaS 2016[8]). At this time, container-based benchmarking suites such as DocLite [149] and a containerized version of CloudSuite [111] also emerged. Moreover, novel cloud hardware architectures could be evaluated using the PALMScloud benchmark suite [158], and a framework

[7] https://www.spec.org/web96/.
[8] https://www.spec.org/cloud_iaas2016/.

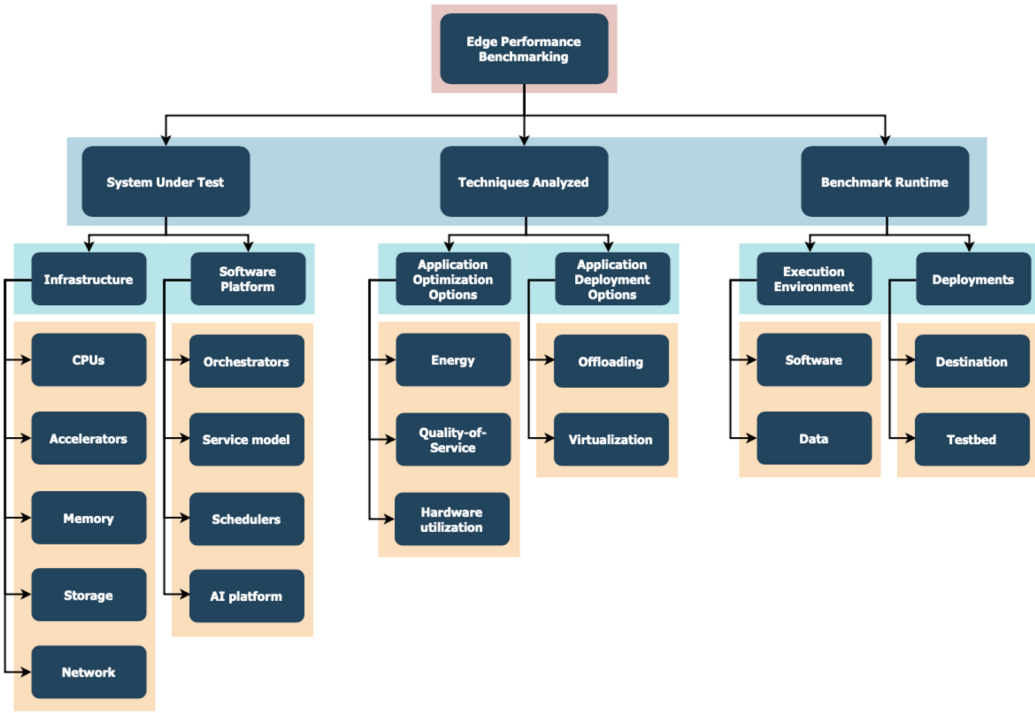

Fig. 3. Classification of edge performance benchmarking under three dimensions: system under test, techniques analyzed, and benchmark runtime.

for benchmarking cloud storage services was introduced [12]. Additionally, a comparison of various benchmarking suites for measuring the quality of cloud services from a client perspective was introduced [19]. More recently, this has led to the development of benchmarks for microservices on distributed clouds [53, 57] and benchmarks that can be used in continuous integration processes [56, 68].

3 EDGE PERFORMANCE BENCHMARKING

This section highlights developments in edge performance benchmarking and then defines a classification that is used in this article for presenting the different dimensions of research undertaken in the context of edge performance benchmarking.

As shown in Figure 2, edge performance benchmarking has been under development since 2015. These benchmarks cover a wide range of resources, including (i) end devices, such as IoT sensors, smartphones, and user gadgets, including wearables; (ii) computational resources located at the edge of a wired network, including routers, switches, gateways, and dedicated resources, including embedded computers and micro clouds; and (iii) cloud resources. Different benchmarks focus on capturing the performance of these resources in both isolated and networked execution contexts.

Existing research on edge performance benchmarking can be classified into the following three dimensions, as shown in Figure 3:

- *System under test* refers to the hardware infrastructure and software platforms that are benchmarked. This aspect is detailed in Section 4 and aims to address **RQ1**, which was posed in Section 1.1.

- *Techniques analyzed* refers to the application optimization options, resource allocation, and application offloading and deployment options analyzed by edge benchmarks, which are discussed in Section 5, where we aim to address **RQ2**, which was posed in Section 1.1.
- *Benchmark runtime* refers to the software and data characteristics of an execution environment and deployment destination, including test beds, which are considered in Section 6, where we aim to address **RQ3**, which was posed in Section 1.1.

As highlighted in Section 1.1, two types of edge performance benchmarking research can be observed in the literature. The first is explicit performance benchmarking, which is defined as research on developing a benchmarking method, benchmark, or toolchain to facilitate performance benchmarking. The second is implicit performance benchmarking, which refers to research that presents evaluations to capture and compare the performances of any of the aforementioned dimensions (i.e., system under test, techniques analyzed, and benchmark runtime) without specifically presenting a method, benchmark, or toolchain.

Table 1 summarizes existing explicit performance benchmarking research that is relevant to edge computing systems by considering the benchmark type (micro/macro), application domain, benchmarks used, and destination platforms (device, edge, and cloud) of 21 edge benchmarking techniques or suites. Micro benchmarks refer to benchmarks that capture system-level (CPU, memory, network, storage) performance metrics. By contrast, macro benchmarks refer to benchmarks that capture application-specific system performance metrics for different application domains. Only two benchmarks capture both micro and macro benchmarks, namely RIoTBench [133] and AIoTBench [96]. The majority of benchmarks are macro benchmarks and only two utilize generic workloads, namely EdgeBench [38] and DeFog [102]. Moreover, only five benchmarks can capture the performance of a complete computation pipeline consisting of a device, edge, and cloud. This can be attributed to the lack of readily available large-scale test beds (although a few are available) for experimentation that integrate a cloud and edge for end users.

The large body of research on edge benchmarking considered in this article relies on either trace data obtained from simulators or on simulators themselves for evaluation, which is contrary to the classic definition of benchmarking. We anticipate experimental benchmarking approaches to be adopted as edge computing matures as a research area and more realistic test beds become readily available.

The next three sections will consider both explicit and implicit edge performance benchmarking research and examine this research across the dimensions of system under test, techniques analyzed, and benchmark runtime. Each subsection is organized to discuss explicit edge performance benchmarks first, followed by implicit edge performance benchmarks.

4 SYSTEM UNDER TEST

In this section, the systems under testing in edge performance benchmarking are examined by considering two components, infrastructure and software platforms, which are discussed in the following subsections. Each subsection will discuss both explicit and implicit edge performance benchmarks. The explicit performance benchmarks are listed in Table 2.

4.1 Infrastructure

Embedded CPUs, accelerators, memory, storage hardware, and the network are the infrastructure resources that are considered in edge performance benchmarking.

4.1.1 CPUs. As shown in Table 2, the majority of edge performance benchmarks can measure the performance of CPUs used on the edge (five of these benchmarks cannot). Many edge benchmarks are evaluated on **single-board computers (SBCs)**, such as Raspberry Pi [67], acting

Table 1. List of Relevant Edge Computing Benchmarks, Including Their Type (micro/macro), Application Domain, Benchmarks Used, and Destination (D, device; E, edge; and C, cloud)

Year	Suite/technique	Type	Domain	Benchmarks used	Destination
2014	CoAP benchmark [83]	Micro	-	lmbench	D
2016	Virt. benchmark [118]	Micro	-	NBench, Sysbench, HPL, bonnie++, DD, Stream, Netperf	D
2017	Virt. benchmark [104]	Micro	-	Sysbench, mbw, fio, iperf, dstat	D
	Voice benchmark [139]	Macro	Voice	Mycroft platform	D, E, C
	RIoTBench [133]	Micro	Stream processing	27 IoT steam processing tasks	D, C
		Macro		Transform and load, Statistical summarization, Predictive analytics	
	TPCx-IoT[9]	Macro	Power grid	Data ingestion and concurrent queries (based on YCSB)	D, E, C
	D-Cube [128]	Micro	Low power systems	6 wireless protocols	D
2018	CAVBench [156]	Macro	Autonomous vehicles	SLAM, Object tracking, Battery diagnostics, Speech recognition, Video analytics	E
	EdgeBench [38]	Macro	Generic	Speech-to-text, Image recognition, Scalar sensor	E, C
	Edge AIBench [61]	Macro	Machine learning	Patient monitor, Surveillance camera, Speech/facial and road sign recognition	D, E, C
	HERMIT [90]	Macro	Internet of Medical Things	Physical activity estimation, Advanced encryption standard, Sleep apnea detection, heart rate variability, Histogram equalization, Inverse radon transform, K-means clustering, Lempel-Ziv-Welch compression, Blood pressure monitor	D
	LEAF [26]	Macro	Federated learning	Image classification, Sentiment analysis, Text prediction	E
	AIoTBench [96]	Micro	Machine learning on mobile devices	Neural network layers - convolution, pointwise convolution, depthwise convolution, matrix multiply, pointwise add, ReLU/sigmoid activation, max/avg. pooling	D
		Macro		Image and Speech recognition, Language translation	
	pCAMP [165]	Macro	Machine learning	TensorFlow, Caffe2, MXNet, PyTorch, TensorFlow Lite	D, E, C
2019	DeFog [102]	Macro	Generic	Object classification, Speech-to-text, Text-audio alignment, Geo-location based mobile game, IoT edge gateway application, Real-time face detection	D, E, C
	Edgedroid [110]	Macro	Human-in-the-loop applications	Cognitive assistance application for assembling LEGO	D, E
	MAVBench [23]	Macro	Micro aerial vehicles	Scanning, Aerial photography, Package delivery, 3D mapping, Search and rescue	D (Drone), C
	IoTBench [87]	Macro	Vision and speech	Video summarization, Stereo image matching, Image recognition, Scan matching, Voice feature extraction, Signals enhancement, Data compression	D
	SoftwarePilot[10] [24]	Macro	Unmanned aerial vehicle	Aerial photography, Crop surveillance, Rescue and discovery, Facial recognition	E, C
	Machine learning accelerator benchmark [120]	Macro	Low power machine learning accelerators	Mobilenet on TensorFlow and OpenVINO	D, E, C
	Edge FPGA-based Neural Net benchmark [41]	Macro	Neural network	Keyword spotting application	E

[9] http://www.tpc.org/tpcx-iot/default5.asp.
[10] https://www.reroutlab.org/softwarepilot/.

Table 2. Comparison of the Characteristics of System under Test in Existing Edge Performance Benchmarks

System under test characteristics considered by edge performance benchmarks	Infrastructure					Software platforms			
	CPUs	Accelerators	Memory	Storage	Network	Orchestrators	Service Model	Schedulers	AI Platforms
CoAP benchmark [83]	Y	N	Y	N	N	N	N	N	N
Virt. benchmark [118]	Y	N	Y	Y	Y	N	N	N	N
Virt. benchmark [104]	Y	N	Y	Y	N	N	N	N	N
Voice benchmark [139]	Y	N	N	N	N	N	N	N	N
RIoTBench [133]	Y	N	Y	N	N	N	N	N	N
D-Cube [128]	Y	N	N	N	N	N	N	N	N
CAVBench [156]	Y	N	Y	N	N	N	N	N	N
EdgeBench [38]	Y	N	Y	N	N	N	Y	N	N
Edge AIBench [61]	N	N	N	N	N	N	N	N	Y
HERMIT [90]	Y	N	Y	N	N	N	N	N	N
LEAF [26]	N	N	N	N	N	N	N	N	Y
AIoTBench [96]	N	N	N	N	N	N	N	N	Y
pCAMP [165]	Y	Y	N	N	N	N	N	N	N
DeFog [102]	Y	N	N	N	Y	N	N	N	N
Edgedroid [110]	N	N	N	N	N	Y	N	N	N
IoTBench [87]	Y	N	N	N	N	N	N	N	N
Machine learning accelerator benchmark [120]	Y	Y	N	N	N	N	N	N	N
Edge FPGA-based Neural Net benchmark [41]	N	Y	N	N	N	N	N	N	N

as edge resources. An SBC is a circuit board consisting of a CPU, memory, network, storage, and other components. Most SBCs adopt ARM processors as CPUs and have low cost and low power characteristics. Additionally, the performance of modern ARM processors is comparable to that of other general-purpose CPUs [117].

The **constrained application protocol (CoAP)** benchmark [83] utilizes lmbench[11] for benchmarking ARM processors in Raspberry Pi, BeagleBone, and BeagleBone Black SBCs in terms of bandwidth and latency. The processing overhead of key operations in a modern WSN gateway can also be measured. An ARM Cortex-A7 dual-core processor hosted by Cubieboard2 was benchmarked [118] using NBench,[12] sysbench,[13] and the HPL benchmark [39]. In a recent study, a wide range of ARM-based SBCs, including the **Raspberry Pi 2 model B (RPi2)**, **Raspberry Pi 3 model B (RPi3)**, **Odroid C1+ (OC1+)**, **Odroid C2 (OC2)**, and **Odroid XU4 (OXU4)** [104], were benchmarked by using sysbench to stress their CPUs. In terms of CPU performance, the Odroid C2 with a Quad-core 2-GHz ARM v8 Cortex-A53 outperformed the other tested SBCs. In terms of power consumption, the Raspberry Pi boards, OC1+, and OC2 achieved high energy efficiency, whereas the OXU4 consumed three to seven times more power than the other SBCs.

[11] http://lmbench.sourceforge.net/.
[12] https://nbench.io/.
[13] https://github.com/akopytov/sysbench.

An ARM Cortex A53 CPU hosted by RPi3 was benchmarked [139] using Mycroft,[14] which is an open-source voice assistant. The latency of the pipeline stages of voice interaction was measured, and the results indicated that cloud services outperform RPi, meaning RPi is more suitable for low-complexity tasks compared to the cloud. RIoTBench [139], an IoT benchmark for stream processing systems, can measure latency, throughput, jitter, and CPU utilization for VMs that process data flow tasks. D-Cube [128] is a benchmarking tool that can profile end-to-end delay, reliability, and power consumption for any CPU designed for deploying IoT protocols on the edge.

An Intel Xeon E3-1275 v5 processor included in the Intel fog reference design was benchmarked using CAVBench [156], which is a benchmarking program for connected and autonomous vehicles. CAVBench deploys computer vision and deep learning applications on a processor and measures the average frames per second and latency. This study concluded that deep learning applications cannot achieve satisfactory performance on the Xeon CPU, meaning accelerators are required. EdgeBench [38] utilizes audio, image, and scalar pipeline applications to evaluate the computation time, end-to-end latency, and CPU utilization of RPi3. HERMIT [90] is a benchmarking suite for medical IoT and was utilized to test RPi3 to determine if it is suitable for medical IoT. Various Intel and ARM CPUs were benchmarked using machine learning models in pCAMP [165]. pCAMP compares TensorFlow, Caffe2, MXNet, PyTorch, and TensorFlow Lite in terms of inference time, total time, and energy consumption. DeFog [102] was used to evaluate ARM-based SBCs, including RPi3 and OXU4, in various fog computing scenarios consisting of object classification and speech-to-text conversion. IoTBench [87] offers diverse IoT applications in the vision, speech recognition, and physiological signal processing domains and was used to evaluate RPi3's performance in terms of executed instructions, cycles, and cache misses.

4.1.2 Accelerators. Although the use of hardware accelerators has been proposed for edge computing [148], there is limited use of such accelerators in the explicit edge performance benchmarks listed in Table 2 (3 of 21 [41, 120, 165]).

pCAMP [165] benchmarks the inference time of deep learning workloads for edge computing on the Nvidia Jetson TX2. Low-power and purpose-built accelerators such as the Intel Movidius Myriad XVPU (Vision Processing Unit) [71], Google Edge TPU [27], NVIDIA 128-core Maxwell and 256-core Pascal architecture-based GPU, and custom **field-programmable gate arrays (FPGAs)** are benchmarked using deep neural networks on edge devices [120]. During the benchmarking process, it was noted that custom FPGAs optimize the performance of specific neural network applications [41] (also refer to Reference [103]). Various FPGA platforms have been evaluated using custom benchmarks that implement separable convolutional neural network keyword spotting [9]. Their performances were compared to that of the Intel NCS platform in terms of inference time, power consumption, and energy per inference [41].

In addition to the abovementioned explicit edge performance benchmarks that evaluate accelerators, implicit edge performance benchmarks consider accelerators as well. The most relevant of these benchmarks are presented below.

The Intel NCS 2 and Google's Coral USB accelerators were benchmarked using popular inference workloads, namely MobileNet-v1 [69] and Inception-v1 [140], in the MLPerf benchmark [119] in terms of inference time and energy efficiency [89].

To explore whether a particular deep learning model can provide sufficient accuracy on edge devices, TomoGAN [92, 93], which is an algorithm for enhancing the quality of X-ray images, was adapted to run on the Google Edge TPU and NVIDIA Jetson TX2 [2]. The benchmarking results

[14] https://mycroft.ai/.

indicated that edge accelerators can provide sufficient accuracy with a novel shallow CNN called the fine-tune network.

In a recent study, the Google Edge TPU, NVIDIA 128-core Maxwell GPU, and Intel Movidius Myriad X VPU were benchmarked by executing eight deep learning models used in personal-scale sensory systems [6]. The benchmarking results revealed that the Google Edge TPU outperformed the other accelerators on all eight models. In terms of energy efficiency, the Google Edge TPU utilized less than 10 mJ of energy for a single execution of any of the eight models, whereas the other platforms consumed between 5 and 274 mJ depending on the model.

4.1.3 Memory. Seven of the explicit benchmarks in Table 2 consider memory when benchmarking. The CoAP benchmark [83] measures memory usage and latency to evaluate the memory performance of SBCs. CAVBench [156] measures the memory bandwidths and footprints of computer vision and deep learning applications, because these applications require significant memory bandwidth and can act as a bottleneck in capacity-limited edge devices. EdgeBench [38] evaluates the memory utilization of benchmark applications on the Raspberry Pi 3B. HERMIT [90] was used to analyze the memory characteristics of benchmark applications, and it was determined that a large L1 D-cache and last-level cache are required in Raspberry Pi to achieve efficient memory access.

A few examples of implicit memory benchmarking in edge computing systems are presented below. The STREAM benchmark,[15] which measures sustainable memory bandwidth, was used to evaluate edge resources [118]. The memory performance of copy, scale, add, and triad operations in STREAM was measured on three different software systems: native Linux, Docker, and **kernel-based virtual machines (KVM)**. The Unix command mbw was utilized to test the memory performance of a wide range of ARM-based SBCs [104]. The mbw command quantifies available memory bandwidth by transferring large arrays of data in memory. OC1+ outperformed RPi2, because OC1+ adopts 792-MHz LPDDR3 RAM, whereas RPi2 uses 400-MHz LPDDR2 RAM. RPi3 utilizes 900-MHz LPDDR2 RAM and outperforms OC1+ in most cases. OC2 and Odroid OXU4 outperform RPi2, RPi3, and OC1+ based on their doubled RAM capacity.

4.1.4 Storage. Only two explicit edge performance benchmarks consider storage devices. Small form factor edge resources typically use flash-based storage devices, such as **embedded MultiMediaCard (eMMC)** and MicroSD. The storage performance of edge resources has been evaluated using Bonnie++[16] and the Unix DD command [118]. Bonnie++ measures data read and write bandwidth, as well as the number of file operations per second. DD is used to measure bandwidth for accessing special device files, such as /dev/zero/. The fio benchmark[17] has been used to perform sequential read/write operations in MicroSD cards, and sysbench has been utilized to perform random disk operations on eMMC cards [104]. The evaluation results revealed that eMMC cards operate at speeds in the order of hundreds of MB/s, whereas MicroSD cards operate in the range of hundreds of Mb/s.

4.1.5 Network. Although the network plays a key role in performance on the edge, there are only two explicit benchmarks that address this aspect. The network performance of Cubieboard2, which provides a 100BASE-TX Fast Ethernet connection, was evaluated using Netperf[18] in native, Docker, and KVM environments [118]. The results revealed that Docker achieves near-native performance, whereas KVM introduces considerable overhead. DeFog [102] was used to measure

[15] https://www.cs.virginia.edu/stream/.
[16] https://www.coker.com.au/bonnie++/.
[17] https://github.com/axboe/fio.
[18] https://hewlettpackard.github.io/netperf/.

communication latency from an edge device to the Amazon cloud when the network bandwidth was fully utilized by the stress-ng operation.[19]

Observation #1. Even though there have been considerable efforts devoted to benchmarking CPUs and moderate efforts for benchmarking memory, additional effort is still required to measure the edge performance of accelerators, storage, and networks effectively.

4.2 Software Platforms

Orchestrators, cloud services, schedulers, and **artificial intelligence (AI)** platforms are examples of software platforms that are benchmarked on edge computing systems. Although other software platforms exist, there has been no research in the context of edge performance benchmarking, so these platforms are not discussed in this article.

4.2.1 Orchestrators. Orchestrators for edge computing manage edge resources by creating containers, deploying and starting servers, and assigning and scaling computational resources. Orchestration in edge computing is challenging because of limited hardware resources, large volumes of edge resources, and the mobility of connected devices [67].

Edgedroid [110] is the only explicit edge performance benchmark that considers orchestrators. Edgedroid evaluates human-in-the-loop applications such as augmented reality and wearable cognitive assistance that are deployed on edge devices. Such applications connect to containers in the cloud for background processing. Edgedroid collects uplink, downlink, and processing time data to identify scaling limits.

However, there are also a few implicit edge performance benchmarks that consider orchestrators. FocusStack coordinates edge resources for moving targets, such as cars and drones [3]. This is a challenging task, because existing orchestrators (e.g., OpenStack) were developed to manage a relatively small number of servers. FocusStack extends OpenStack to support location-based awareness to minimize the number of devices managed at one time. The full-time active monitoring system of FocusStack was benchmarked using the following metric, which represents the total number of bytes transferred every 10 s for monitoring between the orchestrator and a device. FocusStack sent 358 bytes every 10 s while the unmodified OpenStack transferred 17,509 bytes every 10 s.

Edge workload orchestrators that select target edge resources for offloading tasks have been proposed based on fuzzy logic [138]. Such orchestrators can be benchmarked by relying on data obtained from the EdgeCloudSim simulator [137]. The explored applications include augmented reality, healthcare, intensive computing, and infotainment applications, which are evaluated in terms of service time, failed tasks, and virtual machine utilization.

4.2.2 Service Model. Different service models can facilitate the delivery of services on the edge. Service models can be serverless models or **Infrastructure-as-a-Service (IaaS)** models. There is limited consideration for such models in the explicit edge benchmarks listed in Table 2. EdgeBench [38] provides benchmarking applications for the serverless edge computing service model and compares the performances of the AWS IoT Greengrass[20] and Azure IoT Edge[21] platforms.

Some implicit edge performance benchmarks consider the IaaS model and two examples are discussed below. Nebula implements a decentralized edge cloud by utilizing volunteer edge nodes that

[19] https://kernel.ubuntu.com/git/cking/stress-ng.git/.
[20] https://aws.amazon.com/greengrass/.
[21] https://azure.microsoft.com/en-us/services/iot-edge/.

provide computational and storage resources [123]. Nebula offers the IaaS service model (computation and storage services are available). MapReduce, Wordcount, and InvertedIndex applications have been used to benchmark Nebula in terms of performance, fault tolerance, and scalability [81].

Similarly, FemtoClouds offers IaaS-type services by forming small clusters using smartphones and laptops by leveraging idle or less-loaded resources to fulfill user requests [60]. FemtoClouds has been benchmarked for various metrics, including computational throughput, network utilization, and computational resource utilization.

4.2.3 Schedulers. Edge schedulers allow service providers to allocate computing resources efficiently. None of the explicit edge performance benchmarks can compare the performances of edge schedulers. However, several implicit edge performance benchmarks have considered resource scheduling policies to satisfy the real-time requirements of smart manufacturing applications in edge computing [88]. A two-phase scheduling strategy is adopted that first selects a suitable edge computing server based on the target task load, after which additional servers are selected, if necessary, to distribute tasks when one server cannot meet real-time constraints. Schedulers are evaluated on OpenCV applications using metrics such as computing latency, satisfaction degree, and energy consumption. The fairness aspect of edge scheduling was recently evaluated using synthetic benchmarks [99].

4.2.4 Artificial Intelligence Platforms. Three of the explicit edge performance benchmarks consider AI platforms. Edge AIBench [61] provides four AI benchmark applications that can reflect complex scenarios of edge computing, including intensive care unit patient monitoring, surveillance cameras, smart homes, and autonomous vehicles. These test models can be executed using a federated learning framework in a publicly available test bed.[22] LEAF [26] is a modular benchmarking framework for evaluating learning in federated settings. LEAF consists of open-source datasets, statistical and system metrics, and reference implementations. AIoTBench [96] was designed to evaluate the AI capabilities of edge devices for image classification, speech recognition, transformer translation, and micro workloads.

Observation #2. Explicit edge performance benchmarks do not consider software platforms such as orchestrators, service models, and schedulers, which are vital for performance on the edge. Therefore, existing benchmarks cannot capture the integrated performances of services or applications when different software platforms are adopted.

5 TECHNIQUES ANALYZED

This section reviews various techniques that have been analyzed in edge performance benchmarks across the dimensions of application optimization and application deployment options. Table 3 categorizes the benchmarks listed in Table 1. Table 4 summarizes the considered dimensions for a selected set of implicit edge performance benchmarks.

5.1 Application Optimization Options

Tables 3 and 4 list the different optimization options for target applications that are considered in explicit and implicit edge performance benchmarks, respectively. These options are related to energy consumption, **quality of service (QoS)**, and hardware utilization.

5.1.1 Energy Consumption. Energy is an important metric used in many edge performance benchmarks. The virtualization benchmark [104] compares the power consumption rates of five

[22] http://www.benchcouncil.org/testbed.html/.

Table 3. Comparison of the Characteristics of the Techniques Analyzed by Explicit Edge Performance Benchmarks; Offloading: End-user Device, Edge, Cloud, or None; Virtualization: Virtual machine, Container, or None

	Application optimization options			Application deployment options	
Techniques analyzed by edge performance benchmarks	**Energy consumption**	**Quality-of-Service**	**Hardware utilization**	**Offloading**	**Virtualization**
CoAP benchmark [83], HERMIT [90]	N	Y	Y	N	N
Virt. benchmark [118]	N	N	Y	N	V, C
Virt. benchmark [104]	Y	Y	Y	N	C
Voice benchmark [139]	N	Y	N	D → E	N
RIoTBench [133]	N	Y	Y	N	V
TPCx-IoT	N	Y	N	N	N
D-Cube [128]	Y	Y	N	N	N
CAVBench [156]	N	Y	Y	N	N
EdgeBench [38]	N	Y	Y	E → C	V, C
Edge AIBench [61]	N	N	N	N	N
LEAF [26], AIoTBench [96]	N	N	N	N	N
pCAMP [165]	Y	Y	N	N	N
DeFog [102]	N	Y	N	C → E	V, C
Edgedroid [110]	N	Y	N	N	C
MAVBench [23]	Y	Y	N	D → C	N
IoTBench [87]	Y	N	N	N	N
SoftwarePilot	N	N	N	N	V, C
Machine learning accelerator benchmark [120]	Y	N	Y	N	V
Edge FPGA-based Neural Net benchmark [41]	Y	Y	N	N	N

different SoCs acting as end-user devices. This type of evaluation is useful for estimating the battery durations of end-user devices. To select the best low-power wireless protocol, D-Cube [128] measures the power consumption of a target edge system while applying different protocols for an IoT application.

Moreover, energy consumption has been employed to evaluate machine learning packages on systems on the device-edge-cloud continuum [165]. Such benchmarks are useful for optimizing packages for various edge systems. MAVBench [23] compares the energy consumption of a full-on-edge drone to that of a full-on-cloud drone. This performance benchmark provides a breakdown of the energy consumed by different components of MAVs. Similarly, IoTBench [87] evaluates the energy dedicated to different components of benchmarks. Recently, several machine learning benchmarks [41, 120] focusing on the energy consumption of accelerators have been developed.

Several implicit edge performance benchmarking studies have compared different energy consumption techniques, a few of which are discussed below.

Table 4. Comparison of the Characteristics of the Techniques Analyzed by Explicit Edge Performance Benchmarks; Offloading: End-user Device, Edge, Cloud, or None; Virtualization: Virtual Machine, Container, None

	Application optimization options			Application deployment options	
Techniques analyzed by edge performance benchmarks	**Energy consumption**	**Quality-of-Service**	**Hardware utilization**	**Offloading**	**Virtualization**
Resource allocation benchmark [58], MECO benchmark [162], RTLBB [167], EEDOA [32]	Y	N	N	$D \rightarrow E$	N
MIMO benchmark [124]	Y	N	N	$D \rightarrow C$	N
IoT benchmark [134], LBVS [78], MeFoRE [1], VEC benchmark [164]	N	Y	N	$D \rightarrow E$	N
IoT benchmark [135]	N	Y	N	$D \rightarrow E$	N
DYVERSE [154], ENORM [155]	N	Y	N	$C \rightarrow E$	V, C
ECC benchmark [29]	N	Y	N	$D \rightarrow E$	N
YEAST [151], EDAL [161]	Y	Y	N	N	N
Content delivery benchmark [91]	N	Y	Y	N	N
Cloudlet benchmark [33]	N	N	N	$C \rightarrow E$	V
Migration benchmark [98]	N	Y	N	$E \rightarrow E$	C
Cloudlet benchmark [122]	Y	Y	N	$D \rightarrow E$	N
Virt. benchmark [104]	Y	N	Y	N	C

Early efforts led to the formulation of a convex optimization problem for minimizing mobile energy consumption [162]. A multi-user MEC system was considered with a mobile base station and multiple mobile devices, from which tasks were split using a threshold-based policy.

A fine-grained method for multi-resource joint optimization for the energy consumed for task offloading, sub-channel allocation, and CPU-cycle frequency was also developed [167]. Energy efficiency can also be benchmarked when workloads have varying execution times on MEC servers [58]. ThriftyEdge [31] is used to benchmark the performance of delay-aware task graph partitioning and virtual machine selection to minimize IoT device edge resource occupancy. Stochastic optimization for minimizing the energy consumption of task offloading while guaranteeing the average queue length of IoT applications was benchmarked in Reference [32].

5.1.2 Quality-of-Service. QoS is a frequently examined metric in edge performance benchmarks that is represented by execution time, computation and communication latencies, and so on. The CoAP benchmark [83] and HERMIT [90] measure the execution times of IoT applications on end-user devices. When comparing application performance across different layers in device-edge-cloud systems, the voice benchmark [139] and Edgedroid [110] break down execution times according to application components for analysis. A fine-grained study of application latency under varying edge resource availability characteristics and workloads can be performed using DeFog [102]. Existing edge performance benchmarks largely focus on hardware with relatively little emphasis on benchmarking different algorithms to optimize edge systems. Therefore, example

edge performance benchmarking studies that have focused on optimizing QoS based on algorithm designs are discussed below.

Because improving application QoS is a key component of edge computing [150], the evaluation of QoS is very common in edge literature. We discuss a few examples below. ENORM [155] benchmarks the benefits of offloading application services from the cloud to the edge based on the QoS of multiple applications on the same edge node. Priority-based resource scaling approaches can be benchmarked by DYVERSE, which estimates the amount of resources to be added to or removed from a running edge service [154]. The metric used in this evaluation is the QoS violation rate, which is also employed for evaluating the performance of an optimization model for placing IoT services on edge resources to prevent QoS violations [135].

From the perspective of edge service users, MeFoRE [1] uses previous records of **quality of experiences (QoE)**, such as service give-up ratio, to estimate the resources required by different users. QoE is frequently used in utility functions to optimize the performance of running application services (utility-based optimization). Moreover, the use of QoE to measure user utility for running jobs on the edge versus local execution on mobile devices has been benchmarked [30].

5.1.3 Hardware Utilization. The utilization of hardware on the edge is another dimension that can be considered to optimize edge performance. This aspect is also captured by edge performance benchmarks. The virtualization benchmark [118] compares the utilization of different hardware resources on an edge system to select optimal virtualization techniques. Similarly, CPU and memory utilization has been analyzed for benchmarking stream processing platforms [133] and machine learning accelerators [120].

Observation #3. It is noteworthy that the QoS captured by edge performance benchmarks is a dominant criterion for optimizing performance. However, other criteria, such as energy consumption, are not typically captured by edge performance benchmarks.

5.2 Application Deployment Options

Edge benchmarks have captured the performance of application deployment options. These include the direction of deployment and determining how to deploy applications on edge systems, which are considered by reviewing edge performance benchmarks for techniques analyzed for computation offloading and virtualization technologies, respectively. The dimension of deployment in the context of the execution environment is considered in Section 6.2.

5.2.1 Computation Offloading. Only 4 of the 21 explicit edge performance benchmarks listed in Table 3 consider this dimension. This section reviews the computation offloading techniques that are analyzed in explicit edge performance benchmarking.

The following four directions of offloading are relevant to edge environments:

i. Cloud-to-Edge: Voice benchmark [139] characterizes the performance impact of pushing the execution of voice interaction pipelines closer to end-user devices. EdgeBench [38] offloads serverless functions from the cloud to the edge. Edge AIBench [61] and DeFog [102] break down the components of applications and offload some components from the cloud to the edge. Implicit edge performance benchmarks capture the benefits of cloud-to-edge offloading for database replication [91] and application cloning [33].

ii. Edge-to-Cloud: This is the typical direction for the internet of connected vehicles, and utility-based multi-level offloading schemes are evaluated to maximize the utility of vehicles, edge servers, and cloud servers [164]. A collaborative offloading approach has been evaluated to optimize resource allocation and offloading decisions jointly [166].

iii. Edge-to-Edge: When an edge node does not have sufficient resources, it can offload (or migrate) its workload to a peer. Two approaches are typically benchmarked: (i) offload forwarding, which forwards all unprocessed workloads to neighboring edge nodes to meet service objectives [159], and (ii) service migration, which dynamically migrates services across multiple heterogeneous edge nodes [29]. Service handoff approaches have been benchmarked to investigate their feasibility for supporting seamless migration to the nearest edge server when a mobile client is moving [98].

iv. Device-to-Edge: Edgedroid [110] offloads the backend of task guidance for wearable cognitive assistance from end-user devices to the edge. In addition to existing edge benchmarks, typical offloading from end-user devices to the edge has been evaluated for applications that require edge data aggregation for energy saving. For aggregating tasks, data from multiple devices are collected by an edge node for preprocessing and filtering tasks [151, 161].

5.2.2 Virtualization Techniques. More than 50% of the explicit edge performance benchmarks listed in Table 3 execute applications directly on hardware. Some benchmarks have investigated VMs and/or containers, and implicit edge performance benchmarking has also considered unikernels.

Virtual machines are important for edge computing in the context of cloudlets [126]. Implicit benchmarking has focused on the evaluation of VMs on the edge. For example, the selection of the most suitable VM was evaluated for different types of applications [122], and a general approach for SLA-driven scheduling for placing VMs in a multi-network operator-sharing edge environment was benchmarked [78]. The virtualization benchmark [118] compares the performances of VMs to those of containers on different edge systems.

Containers have been extensively evaluated for edge systems based on their reduced boot times and lower resource footprints compared to VMs [47]. Among the explicit edge performance benchmarks, the container is the most frequently adopted virtualization technology for application deployment (e.g., EdgeBench [38], DeFog [102], and Edgedroid [110]). Other benchmarking efforts have highlighted the feasibility of using Docker containers[23] and Linux containers[24] as viable options for providing rapid edge deployment [75, 155].

Unikernels[25] are used for single-purpose applications that use library operating systems and are sealed to modification following deployment [100]. The corresponding small resource footprints are attractive for edge computing. Unikernels and containers have been benchmarked according to the dimensions of scalability, security, and manageability for IoT applications on the edge [105].

Observation #4. It is noteworthy that the majority of existing edge performance benchmarks do not capture performance by deploying applications using virtualization techniques. Therefore, such benchmarks can capture the application performance of edge systems that only run a single application, but not those that operate in a multi-tenant edge environment.

6 BENCHMARK RUNTIME

This section will discuss the execution environments, deployment destinations, and test beds considered by edge performance benchmarks. The execution environments highlight the software- and data-related characteristics considered by different edge performance benchmarks at runtime. Additionally, single and multiple destinations used for deployment by various benchmarks are considered. Finally, different infrastructure deployment options considered by

[23] https://www.docker.com/.
[24] https://linuxcontainers.org/.
[25] http://unikernel.org/.

Table 5. Comparison of the Software and Data Related Execution Environment Characteristics of Edge Performance Benchmarks

	Execution environment characteristics of edge benchmarks	CAVBench [156]	EdgeDroid [110]	EdgeBench [38]	IoTBench [87]	CoAP benchmark [83]	MAVBench [23]	SoftwarePilot [24]	DeFog [102]	Edge AIBench [61]	LEAF [26]	AIoT Bench [96]	HERMIT [90]	pCAMP [165]	RIoTBench [133]
Software (SW)	Reproduced using only open source SW	Y	N	N	N	Y	N	N	N	N	Y	N	Y	Y	N
	Custom/proprietary SW components clearly distinguished and accessible	N	N	Y	N	N	Y	Y	Y	Y	Y	Y	N	N	Y
	Employs commercial-grade software	Y	Y	Y	N	N	Y	Y	Y	Y	N	Y	N	Y	Y
	Consider the effects of compiler, SW runtime and contextual settings on execution	N	Y	N	N	N	Y	Y	Y	N	Y	Y	N	Y	N
Data	Data sets open, accessible and portable	Y	Y	N	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Data generation method: **T**races, **M**arkov Processes or **S**imulation	T	M	T	T	S	S	M	T	T	M	T	T	T	T
	Configure data generation to reproduce a wide range of workload conditions	N	Y	N	N	N	Y	Y	N	N	Y	N	N	N	N

edge performance benchmarks, namely real-world, lab-based, emulated, and simulated test bed infrastructures, are presented.

6.1 Execution Environment

Execution environments consist of the software packages and datasets required to execute a benchmark [25]. Based on growing support from different programming languages (e.g., Python) and virtualization tools (e.g., Docker), execution environments can be reused across projects, research groups, and scientific disciplines. Open, representative, and comprehensive execution environments broaden research avenues by facilitating scientific exploration in new domains. However, execution environments differ significantly across edge computing applications, because such applications change rapidly and frequently. Typically, benchmarks are designed with a narrow focus to target specific workloads by sacrificing the features required to reuse their execution environments.

Table 5 lists seven characteristics of open, representative, and comprehensive execution environments. First, software packages and datasets must be accessible to researchers outside the initial study. Clearly, benchmarks are accessible if they use only open-source software (Characteristic #1). However, benchmarks that represent complex and emergent workloads require custom and/or proprietary software. When it is necessary, such software should be clearly identified and made available (Characteristic #2). Likewise, the datasets used to drive benchmark execution should be open and easily portable across research contexts (Characteristic #5).

Benchmarks represent real workloads by mimicking specific aspects of their functionality. However, software components with limited functionality place less stress on edge devices compared to commercial-grade software. Therefore, execution environments that are at least partially composed of commercial-grade software can ensure representative demand on edge resources (Characteristic #3).

Researchers reuse execution environments by adjusting contextual settings to match their target domains. For example, researchers that study edge resources may ignore user actions related to QoS, whereas researchers that study edge-to-cloud offloading may consider multiple contextual settings for QoS. By design, comprehensive benchmarks support a wide range of settings (Characteristic #7). Execution environments that rely on traces from prior workload executions are often inflexible in terms of runtime and contextual adjustments. Traces have closed data models that cannot be easily extrapolated to what-if conditions. By contrast, execution environments that use data from complex models of simulations or model-driven stochastic methods can be ported to new domains and workload conditions (Characteristics #4 and #6).

Table 5 reveals several interesting trends. Edge benchmarks often employ commercial-grade software components (71%) and use open datasets (84%). These results are likely to be influenced by the machine learning kernels in edge workloads. Commercial-grade versions of neural network platforms, object detection models, and speech processors are open sources that are widely used. By contrast, only 50% of the studied benchmarks support comprehensive evaluation across runtime and contextual settings. This result stems from common dependence on traces from prior executions. Furthermore, 75% of the benchmarks are narrowly tailored to specific contexts and use closed-model traces that are not easily adapted across runtime and contextual settings.

Among the studied benchmarks, EdgeDroid [110, 153], SoftwarePilot [24], and MAVBench [23] have the characteristics of open, representative, and comprehensive execution environments. For example, SoftwarePilot and MAVBench capture virtual reality. Each of these benchmarks employs commercial-grade open-source software components for machine learning and cognitive assistance. Additionally, in the corresponding papers, each benchmark is evaluated across various runtime and contextual settings by adjusting the complexity of machine learning models or adapting user QoS expectations. Leaf [41] is also a comprehensive benchmark that can vary workloads and runtime settings. A subtle difference between these benchmarks is that MavBench relies on simulated environments, which is a design choice that could yield non-representative workloads if simulations deviate from real-world edge conditions. By contrast, SoftwarePilot, EdgeDroid, and Leaf use Markov decision models to adapt real data to new contextual settings. This approach is likely to be more robust for researchers targeting new domains.

Observation #5. Note that most benchmarks do not operate on data from a real-world edge test bed. Instead, they use simulation data, trace data, or data from a test bed adapted to a different contextual setting. While this issue can be attributed to a lack of readily available edge test beds, it also highlights the need for revisiting and validating existing edge performance benchmarking approaches when new edge test beds become available.

Observation #6. There is a limited selection of edge performance benchmarks that can generate data to capture a wide range of workload conditions. This factor translates into benchmarks that are narrowly focused on specific applications and cannot be used generically.

6.2 Deployments

This section will explore the different resource locations at which benchmarks are executed, which are referred to as deployment destinations. It will also review test bed options. The following deployment destinations are considered: (i) single destination and (ii) multiple destinations.

6.2.1 Destination. As mentioned previously, this article considers the edge as resources located at the edge of a wired network. However, a number of researchers have also considered user devices as the edge by adhering to a broader definition of the edge (of a network). Therefore, we consider device-only deployment for a single destination.

i. Single destination refers to benchmark execution on only one device or edge.

a. Device-only: Research that explores device-only benchmarking has been considered for benchmarking in different areas, such as low-power wireless industrial sensors, device-specific protocols, low-overhead virtualization, medical IoT devices, and devices that will execute machine-learning-based workloads.

Low-power wireless industrial sensors: Wireless protocols for industrial sensor devices have been benchmarked using observation modules [128]. The six protocols considered are (i) Enhanced ContikiMAC, (ii) Thompson-sampling-based channel selection, (iii) Glossy, (iv) Chaos, (v) Sparkle, and (vi) Time-slotted channel hopping. Power consumption and end-to-end latencies are profiled, and an additional validation mechanism is incorporated to evaluate the accuracy of benchmarking.

Device-specific protocols: Some researchers have benchmarked system architectures for CoAP-based IoT devices [83]. The key results from benchmarking can be summarized as follows: (i) the selected processor directly impacts CoAP server performance and (ii) the latency of communication channels affects the round-trip times of CoAP requests.

Low-overhead virtualization: The performance of virtualization for user devices has been benchmarked extensively [104, 118]. A range of virtualization techniques such as docker containers and KVMs have been considered [118]. The key results are that hypervisor-based virtualization incurs large overhead and containers seem to be more appropriate for network edges. Container-based virtualization across five different devices has also been considered [104]. Moreover, there is a negligible impact on performance when using containers compared to bare-metal execution. The characteristics of workloads represent key information for estimating the energy efficiency of devices.

Medical IoT devices: The computation and memory characteristics of IoT devices used in the medical domain have been benchmarked [90]. A collection of medical applications (macro benchmarks) was evaluated against the MiBench, PARSEC, and CPU06 micro benchmarks. The evaluation results revealed that execution characteristics differ between micro and macro benchmarks.

Devices that run machine learning workloads: The benchmarking of machine-learning-specific workloads for various devices was presented in Reference [96]. Both micro benchmarks, such as the individual layers of a neural network, and macro benchmarks, such as applications in image classification, speech recognition, and language translation on the TensorFlow and Caffe2 frameworks, were considered. Similarly, benchmarking for the vision and speech domains has also been considered [87].

b. Edge-only: Research that explores benchmarking for edge-only deployment will ideally utilize macro benchmarks that are edge native (edge resources are not merely accelerators, but are essential for an application to be used in the real world). Autonomous vehicles are one such application. CAVBench is a benchmarking suite developed for autonomous vehicles by focusing on real-time applications that must process unstructured data [156]. The edge node employed in this research was designed based on the Intel fog reference design. Hardware accelerators, such as FPGAs on the edge, have been benchmarked for convolutional neural networks in the context of keyword spotting [41]. It is worth nothing that this use case may not necessarily represent an edge-native benchmark.

ii. Multiple-destination: This refers to the execution of a benchmark either across an entire device-edge-cloud resource pipeline or a partial resource pipeline (e.g., device-edge or edge-cloud).

a. Entire device-edge-cloud pipeline: There are a number of examples of benchmarking studies that leverage an entire resource pipeline consisting of a device, the edge of a wired network, and a cloud for benchmarking. Three main types of applications are considered.

Service pushdown: Voice-interactive applications have been used as macro benchmarks for analyzing entire resource pipelines [139]. The goal is to push services from the cloud across weak devices and the edge to optimize applications to obtain consistent dialog latency.

Data aggregation: Benchmarks that are relevant to the data-intensive workflows of power grids have been presented in the context of an entire resource pipeline (TPCx-IoT).[26] This workflow enables real-time analytics to be performed on gateways while ingesting data from 200 different types of power station sensors. This benchmark operates in two runs: one run for a warm up and another for measurement. Performance, price, and availability metrics are considered. The majority of benchmarks considered by DeFog fall under this category [102].

Edge inference: Benchmarks are employed to train machine learning models on the cloud and perform inference on the edge (assuming that a trained model is available on the edge) [61]. A variety of devices or sensors are considered to generate data in patient monitoring, surveillance, or smart home scenarios. Similarly, inference on a device, edge, or cloud for different machine learning platforms, such as TensorFlow, Caffe2, PyTorch, MXNet, and TensorFlowLite, is considered by pCAMP [165], which can also consider different accelerators [120]. It should be noted that in this case, inference is not distributed, because it is performed entirely on the edge.

b. Partial pipeline: The following three combinations for benchmarking on partial resource pipelines are considered.

Device-Edge: Edgedroid [110] is an example of benchmarking human-in-the-loop applications, such as cognitive assistance in an edge-cloud deployment, where the edge is a cloudlet. The underlying benchmarking approach is to mimic applications by replaying traces of sensory inputs that are obtained from running the application in the real world. The feedback generated by processing such sensory inputs on the cloudlet is processed using a model of human reactions. This enhances the understanding of latency tradeoffs in the contexts of both the application and edge-cloud deployment.

Device-Cloud: The device-cloud pipeline focuses on estimating the performance of distributed intelligence systems [76] and IoT stream application compositions [133]. The former type of system deploys a sliced neural network across a user device and cloud for distributed inference by estimating where a neural network should be sliced. The latter facilitates benchmarking by combining distributed stream applications using modular IoT tasks.

Edge-Cloud: This deployment pipeline is typically used to investigate the performance of applications and the lifecycle of application service offloading from the cloud to the edge and vice-versa (although this is not exclusive to the edge-cloud pipeline and is also relevant in other partial pipelines and the entire resource pipeline (refer to Section 5.2.1)). Benchmarks that exploit this pipeline include EdgeBench [38] and SoftwarePilot [24].

Observation #7. A number of explicit edge performance benchmarks focus on either a device or edge as a deployment destination. There are relatively few edge performance benchmarks that consider the entire resource pipeline, which integrates a cloud, edge, and device.

6.2.2 Test Beds. The test bed options for deploying edge benchmarks include real-world and physical, lab-based experimental, emulated, or simulated infrastructures.

i. Real-world physical infrastructure: This refers to "in the wild" physical test beds that closely mimic the operational characteristics of an actual edge deployment. There are only a few such test

[26] http://www.tpc.org/tpc_documents_current_versions/pdf/tpcx-iot_v1.0.4.pdf.

beds available, such as the Living Edge Lab[27] and those reported in References [107, 121]. There has been no evaluation of any of the benchmarks listed in Table 1 on real-world infrastructures.

ii. Lab-based experimental infrastructure: The majority of test beds on which the edge applications or benchmarks listed in Table 1 have been evaluated are lab-based infrastructures. Such test beds may be public cloud offerings or private cloud resources coupled with edge resources in the form of single-board computers [102, 155] (or a cluster [142]) or routers/gateways. End user devices may range from wireless sensors [128] to user gadgets [139, 155]. Moreover, a number of benchmarks do not rely on real data, but instead use simulation data based on the complexity of the environments they benchmark (e.g., autonomous cars [156] or drones [24]).

iii. Emulated infrastructure: Real-world physical infrastructure is not easily accessible to many researchers, and many lab-based experimental infrastructures are small in scale and do not represent the characteristics of a real infrastructure. Therefore, various types of emulated infrastructures have recently been proposed [62]. An emulated environment relies on deploying edge servers on the cloud and configuring these servers, as well as the network and interconnects, such that they represent real edge environments. However, this method assumes knowledge of the parameters required to configure a realistic emulated edge environment, which can only be acquired from a real-world edge infrastructure.

iv. Simulation infrastructure: A number of simulators, such as EdgeCloudSim [137], iFogSim [59], FogExplorer [63, 64], and MyiFogSim [94], are available for edge computing and can provide insights into basic design choices. However, more complex integration tests and application component-specific analysis cannot be performed using such simulators. Therefore, such methods should only be used as a fallback solution when real or emulated benchmarking infrastructures are not available.

Observation #8. Following observation #5, we note that most edge performance benchmarking is conducted on experimental, emulated, or simulation-based infrastructures that may not be representative of large and geo-distributed real-world physical infrastructures.

7 FUTURE DIRECTIONS AND CONCLUSIONS

This survey provided a catalog of explicit edge performance benchmarks and a subset of implicit edge performance benchmarking research to achieve objective O1 stated in Section 1.1. We presented a brief timeline of performance benchmarking for different computing systems and then considered edge performance benchmarking to achieve objective O2 stated in Section 1.1. The key dimensions for edge performance benchmark categorization include the system under test, techniques analyzed, and benchmark runtime, which were examined to achieve objective O3 stated in Section 1.1. In exploring the key dimensions of edge performance benchmarks, we addressed the three key research questions posed in Section 1.1. Furthermore, we highlighted eight observations relevant to the scope of this article. The final objective (O4 in Section 1.1) of presenting future research directions is accomplished in this section. The eight observations will be mapped onto future research directions, and the general areas of research will be discussed.

Eight avenues for pursuing future edge performance benchmarking are presented below.

i. Widening the scale of geo-distribution: Following Observation #8, most current edge performance benchmarking research is not conducted on real test beds with geo-distributed infrastructures. Many existing benchmarks are designed to capture the performances of individual cloud or edge resources in lab-based test beds. Therefore, these benchmarks do not necessarily capture the performance of large-scale geo-distribution that will be observed in

[27] https://www.openedgecomputing.org/living-edge-lab/.

an edge computing environment. The absence of comprehensive edge datasets has been also reported previously [80]. Although existing performance benchmarks have provided important insights, more comprehensive edge performance benchmarks must fully embrace geo-distributed benchmarking for large-scale collections of cloud and edge resources.

ii. Developing edge-specific quality and performance metrics and measurement techniques on different platforms: As noted in Observation #2, current edge performance benchmarks do not capture performance on different software platforms, such as orchestrators, schedulers, and service models. To evaluate such software platforms, multiple multi-instance workloads with workflows for the relevant applications are required to test the resource provisioning and sharing capabilities of edge platforms. Addressing this issue would provide researchers with a valuable tool for quantifying the performance of edge applications that will run on a variety of platforms. Additionally, current efforts largely focus on capturing metrics that are historically relevant to distributed systems, such as grids and clouds. Although such metrics are useful, it is likely that edge-specific metrics considering the transient and massively dispersed nature of such environments have not yet been developed. Additionally, novel techniques to capture these metrics may be required.

iii. Evaluating benchmarks on real-world infrastructures: Lab-based infrastructures are the most common test beds used to evaluate edge benchmarks, as noted in Observation #5. At least in part, this can be attributed to limited access to real test beds. Regardless, additional efforts are required to evaluate existing benchmarks on real and resource-rich test beds to identify limitations in current benchmarking approaches.

iv. Developing lightweight and rapid edge benchmarks that capture application performance in multi-tenant environments: Generally, edge and mobile resources have limited capabilities to execute common and extensive benchmark applications designed for large data center servers. Many HPC applications have been used to capture the performance of CPUs and accelerators for edge computing, but such methods are very time consuming. Additionally, running unmodified Spark or Hadoop applications for big data platforms requires significant time and resources to obtain useful results. Therefore, lightweight benchmarking in terms of actual benchmarks and measurement techniques must be designed and developed for edge platforms. As noted in Observation #4, many current edge performance benchmarks execute a single application without considering virtualization. Recent edge systems are designed to utilize virtual machines and containers to support multi-tenancy [67]. Thus, there is a need to design and develop lightweight and rapid edge benchmarks for multi-tenant edge environments that can quantify the impact of multiple concurrent users competing for the same resources.

v. Developing standardized benchmarks across the entire resource pipeline for capturing offloading performance and varying workload conditions: The premise of edge computing is to offload tasks either from a cloud or devices to an edge to reduce the overall response time of an application and improve energy efficiency. Most edge performance benchmarks do not capture the performance of an entire resource pipeline (devices, edges, and clouds), as noted in Observation #7, meaning they do not capture the performance of offloading coherently. Many evaluations presented in the existing literature have used various workloads and metrics relevant to specific platforms or test beds. Therefore, they are non-standard benchmarks and are not compatible across different infrastructures. Additionally, as noted in Observation #5, existing edge performance benchmarks have limited flexibility in terms of capturing a wide range of workload conditions. Hence, a more comprehensive and standard approach for benchmarking offloading mechanisms and varying workload conditions must be considered.

vi. Maturing edge performance benchmarking: Current edge performance benchmarks typically consider QoS metrics, while other relevant criteria, such as energy consumption, are not considered, as noted in Observation #3. Additionally, most metrics focus on CPUs, but additional

research is required to quantify the performance of accelerators, storage, and networks at the edge, as noted in Observation #1. A benchmarking suite that can holistically capture CPU, accelerator, memory, storage, and network performance on edge platforms and generate performance scores that are normalized against reference platforms is required. These are a few relevant areas that must receive additional attention from the research community to advance edge performance benchmarking research.

vii. Moving beyond performance in edge benchmarking: In addition to performance in benchmarking, there are other quality dimensions such as elastic scalability, data consistency, security and privacy, and availability that are typically in direct or indirect tradeoff relationships [10, 19]. Current *edge* benchmarking approaches do not consider quality dimensions beyond performance. Benchmarking approaches from other closely related domains, such as cloud computing, can often be adapted or reused for edge environments. For example, there are large numbers of starting points for elastic scalability [20, 53, 74, 82, 86, 116], availability [16, 18, 50, 65, 79, 129, 141], data consistency [4, 10, 11, 14, 15, 55, 152, 163], and security and privacy [7, 17, 34, 77, 106, 112, 113, 131]. A more detailed discussion of additional quality dimensions for edge benchmarking is beyond the scope of this paper.

viii. Security/privacy-specific edge benchmarks: Edge systems are significantly more complex than previous iterations of distributed systems (volume of devices connected, heterogeneity of resources and technological domains spanned, and edge resources that are accessible for computing, which were previously unavailable or concealed within networks). This complexity naturally creates a large attack surface and multiple vulnerable spots in terms of data privacy. Therefore, benchmarks that provide insights into identifying security mechanisms for orchestrating services and suitable security standards for complex systems would be very useful.

Relevance of edge performance benchmarking to practitioners: Edge performance benchmarking is a nascent topic within edge computing research. We anticipate that multiple practitioner groups will benefit from edge performance benchmarks. We list four relevant groups below [19, 102]. (i) Edge hardware vendors can tabulate and demonstrate the advantages of edge computing by using performance benchmarks. (ii) System software administrators can investigate the effects on edge applications when changes are introduced within an edge compute infrastructure, such as updates or patches to operating systems, system software, or runtime libraries. (iii) Service providers can select the most appropriate geographic locations for deploying micro and modular data centers on the edge and can quantify the performance of specific applications to justifying their choice of location. (iv) Network administrators may wish to quantify edge application performance when changes are introduced in the network stack, such as a new network protocol or security patch in a specific layer of the stack.

Additionally, real-time edge performance benchmarking can be integrated with automated edge software development and adaptive edge orchestration platforms to select the most appropriate edge resource for deployment based on current performance and network conditions. In this context, edge performance benchmarks will be of significant interest to any edge application developer.

ACKNOWLEDGMENTS

The authors are grateful to the anonymous reviewers for their comments.

REFERENCES

[1] M. Aazam, M. St-Hilaire, C. Lung, and I. Lambadaris. 2016. MeFoRE: QoE based resource estimation at fog to enhance QoS in IoT. In *Proceedings of the International Conference on Telecommunications*. 1–5.

[2] V. Abeykoon, Z. Liu, R. Kettimuthu, G. Fox, and I. Foster. 2019. Scientific image restoration anywhere. In *Proceedings of the IEEE/ACM 1st Annual Workshop on Large-scale Experiment-in-the-Loop Computing*. 8–13.
[3] B. Amento, B. Balasubramanian, R. Hall, K. Joshi, G. Jung, and K. Purdy. 2016. FocusStack: Orchestrating edge clouds using location-based focus of attention. In *Proceedings of the IEEE/ACM Symposium on Edge Computing*. 179–191.
[4] E. Anderson, X. Li, M. A. Shah, J. Tucek, and J. J. Wylie. 2010. What consistency does your key-value store actually provide? In *Proceedings of the 6th Workshop on Hot Topics in System Dependability*. 1–16.
[5] M. I. Andreica, N. Tapus, C. Dumitrescu, A. Iosup, D. Epema, I. Raicu, I. Foster, and M. Ripeanu. 2006. Towards ServMark, an architecture for testing grid services. *Technical Report ServMark-2006-002, Technical University of Delft, July 2006*.
[6] M. Antonini, T. H. Vu, C. Min, A. Montanari, A. Mathur, and F. Kawsar. 2019. Resource characterisation of personal-scale sensing models on edge accelerators. In *Proceedings of the 1st International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*. 49–55.
[7] G. Apostolopoulos, V. Peris, and D. Saha. 1999. Transport layer security: How much does it really cost? In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 2. 717–725.
[8] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, and et al. 1991. The NAS parallel benchmarks—Summary and preliminary results. In *Proceedings of the ACM/IEEE Conference on Supercomputing*. 158–165.
[9] G. Benelli, G. Meoni, and L. Fanucci. 2018. A low power keyword spotting algorithm for memory constrained embedded systems. In *Proceedings of the 2018 IFIP/IEEE International Conference on Very Large Scale Integration*. 267–272.
[10] D. Bermbach. 2014. *Benchmarking Eventually Consistent Distributed Storage Systems*. Ph.D. Dissertation. Karlsruhe Institute of Technology.
[11] D. Bermbach and J. Kuhlenkamp. 2013. Consistency in distributed storage systems: An overview of models, metrics and measurement approaches. In *Networked Systems*, V. Gramoli and R. Guerraoui (Eds.). Lecture Notes in Computer Science, Vol. 7853. Springer, Berlin, 175–189.
[12] D. Bermbach, J. Kuhlenkamp, A. Dey, A. Ramachandran, A. Fekete, and S. Tai. 2017. BenchFoundry: A benchmarking framework for cloud storage services. In *Proceedings of the 15th International Conference on Service Oriented Computing*.
[13] D. Bermbach, F. Pallas, David García Pérez, P. Plebani, M. Anderson, R. Kat, and S. Tai. 2017. A research perspective on fog computing. In *Proceedings of the 2nd Workshop on IoT Systems Provisioning and Management for Context-Aware Smart Cities*.
[14] D. Bermbach and S. Tai. 2011. Eventual consistency: How soon is eventual? An evaluation of Amazon S3's consistency behavior. In *Proceedings of the 6th Workshop on Middleware for Service Oriented Computing*. Article 1, 1:1–1:6 pages.
[15] D. Bermbach and S. Tai. 2014. Benchmarking eventual consistency: Lessons learned from long-term experimental studies. In *Proceedings of the 2nd International Conference on Cloud Engineering*. 47–56.
[16] D. Bermbach and E. Wittern. 2016. Benchmarking web API quality. In *Proceedings of the 16th International Conference on Web Engineering*. 188–206.
[17] D. Bermbach and E. Wittern. 2019. Benchmarking web API quality-revisited. *arXiv:1903.07712*. Retrieved from https://arxiv.org/abs/1903.07712.
[18] David Bermbach and Erik Wittern. 2020. Benchmarking web API quality—Revisited. *J. Web Eng.* 19, 5–6 (2020), 603–646.
[19] D. Bermbach, E. Wittern, and S. Tai. 2017. *Cloud Service Benchmarking: Measuring Quality of Cloud Services from a Client Perspective*. Springer.
[20] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing. 2009. How is the weather tomorrow? Towards a benchmark for the cloud. In *Proceedings of the 2nd International Workshop on Testing Database Systems*. 1–6.
[21] A. Bond, D. Johnson, G. Kopczynski, and H. R. Taheri. 2016. Profiling the performance of virtualized databases with the TPCx-V benchmark. In *Performance Evaluation and Benchmarking: Traditional to Big Data to Internet of Things*, R. Nambiar and M. Poess (Eds.). Springer International Publishing, 156–172.
[22] A. H. Borhani, P. Leitner, B. S. Lee, X. Li, and T. Hung. 2014. WPress: An application-driven performance benchmark for cloud-based virtual machines. In *Proceedings of the IEEE 18th International Enterprise Distributed Object Computing Conference*. 101–109.
[23] B. Boroujerdian, H. Genc, S. Krishnan, W. Cui, A. Faust, and V. J. Reddi. 2018. MAVBench: Micro aerial vehicle benchmarking. In *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture*. 894–907.
[24] J. Boubin, N. Babu, C. Stewart, J. Chumley, and S. Zhang. 2019. Managing edge resources for fully autonomous aerial systems. In *Proceedings of the ACM Symposium on Edge Computing*.

[25] A. Burns and A. J. Wellings. 2001. *Real-time Systems and Programming Languages: Ada 95, Real-time Java, and Real-time POSIX*. Pearson Education.
[26] S. Caldas, P. Wu andd T. Li, J.b Konecný, H. B. McMahan, V. Smith, and A. Talwalkar. 2018. LEAF: A benchmark for federated settings. arxiv:1812.01097. Retrieved from http://arxiv.org/abs/1812.01097.
[27] S. Cass. 2019. Taking AI to the edge: Google's TPU now comes in a maker-friendly package. *IEEE Spectr.* 56, 5 (2019), 16–17.
[28] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S. Lee, and K. Skadron. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *Proceedings of the IEEE International Symposium on Workload Characterization*. 44–54.
[29] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, and I. Humar. 2019. A dynamic service migration mechanism in edge cognitive computing. *ACM Trans. Internet Technol.* 19, 2 (2019), 1–15.
[30] X. Chen, L. Jiao, W. Li, and X. Fu. 2015. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Netw.* 24, 5 (2015), 2795–2808.
[31] X. Chen, Q. Shi, L. Yang, and J. Xu. 2018. ThriftyEdge: Resource-efficient edge computing for intelligent IoT applications. *IEEE Netw.* 32, 1 (2018), 61–65.
[32] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen. 2019. Energy efficient dynamic offloading in mobile edge computing for Internet of Things. *IEEE Trans. Cloud Comput.* (2019).
[33] Z. Chen, L. Jiang, W. Hu, K. Ha, B. Amos, P. Pillai, A. Hauptmann, and M. Satyanarayanan. 2015. Early implementation experience with wearable cognitive assistance applications. In *Proceedings of the Workshop on Wearable Systems and Applications*. ACM, 33–38.
[34] C. Coarfa, P. Druschel, and D. S. Wallach. 2006. Performance analysis of TLS web servers. *ACM Trans. Comput. Syst.* 24, 1 (2006), 39–69.
[35] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. 2010. Benchmarking cloud serving systems with YCSB. In *Proceedings of the 1st ACM Symposium on Cloud Computing*. 143–154.
[36] H. J. Curnow and B. A. Wichmann. 1976. A synthetic benchmark. *Comput. J.* 19, 1 (01 1976), 43–49.
[37] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter. 2010. The scalable heterogeneous computing (SHOC) benchmark suite. In *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*. 63–74.
[38] A. Das, S. Patterson, and M. P. Wittie. 2018. EdgeBench: Benchmarking edge computing platforms. In *Proceedings of the 4th International Workshop on Serverless Computing*.
[39] T. Davies, C. Karlsson, H. Liu, C. Ding, and Z. Chen. 2011. High performance LINPACK benchmark: A fault tolerant implementation without checkpointing. In *Proceedings of the International Conference on Supercomputing*. 162–171.
[40] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux. 2013. OLTP-bench: An extensible testbed for benchmarking relational databases. *Proc. VLDB Endow.* 7, 4 (2013), 277–288.
[41] G. Dinelli, G. Meoni, E. Rapuano, G. Benelli, and L. Fanucci. 2019. An FPGA-based hardware accelerator for CNNs using on-chip memories only: Design and benchmarking with Intel movidius neural compute stick. *Int. J. Reconfig. Comput.* 2019, Article 7218758 (2019).
[42] J. J. Dongarra, P. Luszczek, and A. Petitet. 2003. The LINPACK benchmark: Past, present and future. *Concurr. Comput.: Pract. Exp.* 15, 9 (2003), 803–820.
[43] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras. 2013. Benchmarking personal cloud storage. In *Proceedings of the Internet Measurement Conference*. 205–212.
[44] C. Dumitrescu, I. Raicu, M. Ripeanu, and I. Foster. 2004. DiPerF: An automated distributed performance testing framework. In *Proceedings of the IEEE/ACM International Workshop on Grid Computing*. 289–296.
[45] R. Eigenmann and S. Hassanzadeh. 1996. Benchmarking with real industrial applications: The SPEC high-performance group. *IEEE Comput. Sci. Eng.* 3, 1 (1996), 18–23.
[46] A. Elhabbash, F. Samreen, J. Hadley, and Y. Elkhatib. 2019. Cloud brokerage: A systematic survey. *Comput. Surv.* 51, 6, Article 119 (2019).
[47] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio. 2015. An updated performance comparison of virtual machines and Linux containers. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*. 171–172.
[48] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. Daniel Popescu, A. Ailamaki, and B. Falsafi. 2012. Clearing the clouds: A study of emerging scale-out workloads on modern hardware. *ACM SIGPLAN Not.* 47, 4 (2012), 37–48.
[49] I. Foster and C. Kesselman (Eds.). 1998. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann.
[50] A. Fox and E. A. Brewer. 1999. Harvest, yield, and scalable tolerant systems. In *Proceedings of the 7th Workshop on Hot Topics in Operating Systems*. 174–178.

[51] M. Frumkin and R. F. Van der Wijngaart. 2001. NAS grid benchmarks: A tool for grid space exploration. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*. 315–322.

[52] A. Gaikwad, V. Doan, M. Bossy, F. Baude, and F. Abergel. 2012. SuperQuant financial benchmark suite for performance analysis of grid middlewares. In *Modeling, Simulation and Optimization of Complex Processes*, H. G. Bock, X. P. Hoang, R. Rannacher, and J. P. Schlöder (Eds.). Springer, Berlin, 103–113.

[53] Y. Gan, Y. Zhang, D. Cheng, A. Shetty, P. Rathi, N. Katarki, A. Bruno, J. Hu, B. Ritchken, B. Jackson, K. Hu, M. Pancholi, Y. He, B. Clancy, C. Colen, F. Wen, C. Leung, S. Wang, L. Zaruvinsky, M. Espinosa, R. Lin, Z. Liu, J. Padilla, and C. Delimitrou. 2019. An open-source benchmark suite for microservices and their hardware-software implications for cloud and edge systems. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*. 3–18.

[54] L. Gillam, B. Li, J. O'Loughlin, and A. P. S. Tomar. 2013. Fair benchmarking for cloud computing system. *J. Cloud Comput.: Adv. Syst. Appl.* 2, 1 (2013).

[55] W. Golab, X. Li, and M. A. Shah. 2011. Analyzing consistency properties for fun and profit. In *Proceedings of the 30th Symposium on Principles of Distributed Computing*. 197–206.

[56] M. Grambow, F. Lehmann, and D. Bermbach. 2019. Continuous benchmarking: Using system benchmarking in build pipelines. In *Proceedings of the 1st Workshop on Service Quality and Quantitative Evaluation in New Emerging Technologies*.

[57] M. Grambow, L. Meusel, E. Wittern, and D. Bermbach. 2020. Benchmarking microservice performance: A pattern-based approach. In *Proceedings of the 35th ACM Symposium on Applied Computing*.

[58] J. Guo, Z. Song, Y. Cui, Z. Liu, and Y. Ji. 2017. Energy-efficient resource allocation for multi-user mobile edge computing. In *Proceedings of the IEEE Global Communications Conference*. 1–7.

[59] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya. 2017. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments. *Softw.: Pract. Exp.* 47, 9 (2017), 1275–1296.

[60] K. Habak, M. Ammar, K. A. Harras, and E. Zegura. 2015. Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In *Proceedings of the IEEE 8th International Conference on Cloud Computing*. 9–16.

[61] T. Hao, Y. Huang, X. Wen, W. Gao, F. Zhang, C. Zheng, L. Wang, H. Ye, K. Hwang, Z. Ren, and J. Zhan. 2018. Edge AIBench: Towards comprehensive end-to-end edge computing benchmarking. In *Proceedings of the 1st BenchCouncil International Symposium on Benchmarking, Measuring, and Optimizing*. 23–30.

[62] J. Hasenburg, M. Grambow, E. Grunewald, S. Huk, and D. Bermbach. 2019. MockFog: Emulating fog computing infrastructure in the cloud. In *Proceedings of the 1st IEEE International Conference on Fog Computing*.

[63] Jonathan Hasenburg, Sebastian Werner, and David Bermbach. 2018. FogExplorer. In *Proceedings of the 19th International Middleware Conference, Demos, and Posters (MIDDLEWARE'18)*. ACM.

[64] Jonathan Hasenburg, Sebastian Werner, and David Bermbach. 2018. Supporting the evaluation of fog-based IoT applications during the design phase. In *Proceedings of the 5th Workshop on Middleware and Applications for the Internet of Things (M4IoT'18)*. ACM.

[65] T. Hauer, P. Hoffmann, J. Lunney, D. Ardelean, and A. Diwan. 2020. Meaningful availability. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation*. 545–557.

[66] M. A. Heroux and J. J. Dongarra. 2013. *Toward a New Metric for Ranking High Performance Computing Systems*. Technical Report SAND2013-4744. Sandia National Lab.

[67] C.-H. Hong and B. Varghese. 2019. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *Comput. Surv.* 52, 5 (2019).

[68] A. Van Hoorn, J. Waller, and W. Hasselbring. 2012. Kieker: A framework for application performance monitoring and dynamic software analysis. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*. 247–248.

[69] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*. Retrieved from https://arxiv.org/abs/1704.04861.

[70] B. Hu and C. J. Rossbach. 2019. Mirovia: A benchmarking suite for modern heterogeneous computing. arXiv:1906.10347. Retrieved from http://arxiv.org/abs/1906.10347.

[71] M. H. Ionica and D. Gregg. 2015. The movidius myriad architecture's potential for scientific computing. *IEEE Micro* 35, 1 (2015), 6–14.

[72] A. Iosup and D. Epema. 2006. GRENCHMARK: A framework for analyzing, testing, and comparing grids. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*. 313–320.

[73] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. 2011. Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Trans. Parallel Distrib. Syst.* 22, 6 (2011), 931–945.

[74] S. Islam, K. Lee, A. Fekete, and A. Liu. 2012. How a consumer can measure elasticity for cloud platforms. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*. 85–96.
[75] B. Ismail, E. Goortani, M. Ab Karim, W. Tat, S. Setapa, J. Luke, and O. Hoe. 2015. Evaluation of Docker as edge computing platform. In *Proceedings of the IEEE Conference on Open Systems*. IEEE, 130–135.
[76] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. 2017. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *Proceedings of the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems*. 615–629.
[77] K. Kant, R. Iyer, and P. Mohapatra. 2000. Architectural impact of secure socket layer on Internet servers. In *Proceedings of the International Conference on Computer Design*. 7–14.
[78] K. Katsalis, T. Papaioannou, N. Nikaein, and L. Tassiulas. 2016. SLA-driven VM scheduling in mobile edge computing. In *Proceedings of the IEEE International Conference on Cloud Computing*. 750–757.
[79] M. Klems, M. Menzel, and R. Fischer. 2010. Consistency benchmarking: Evaluating the consistency behavior of middleware services in the cloud. In *Service-Oriented Computing*, P. Maglio, M. Weske, J. Yang, and M. Fantinato (Eds.). Lecture Notes in Computer Science, Vol. 6470. Springer, Berlin, 627–634.
[80] O. Kolosov, G. Yadgar, S. Maheshwari, and E. Soljanin. 2020. Benchmarking in the dark: On the absence of comprehensive edge datasets. In *Proceedings of the 3rd USENIX Workshop on Hot Topics in Edge Computing*.
[81] D. Komosny, J. Pruzinsky, P. Ilko, J. Polasek, P. Masek, and O. Kocatepe. 2015. On geographic coordinates of PlanetLab europe. In *Proceedings of the 38th International Conference on Telecommunications and Signal Processing*. 642–646.
[82] D. Kossmann, T. Kraska, and S. Loesing. 2010. An evaluation of alternative architectures for transaction processing in the cloud. In *Proceedings of the 30th International Conference on Management of Data*. 579–590.
[83] C. P. Kruger and G. P. Hancke. 2014. Benchmarking Internet of Things devices. In *Proceedings of the 12th IEEE International Conference on Industrial Informatics*. 611–616.
[84] J. Kuhlenkamp, M. Klems, and O. Röss. 2014. Benchmarking scalability and elasticity of distributed database systems. In *Proceedings of the International Conference on Very Large Databases*. 1219–1230.
[85] J. Kuhlenkamp, K. Rudolph, and D. Bermbach. 2015. AISLE: Assessment of provisioned service levels in public IaaS-based database systems. In *Proceedings of the 13th International Conference on Service-Oriented Computing*. 154–168.
[86] J. Kuhlenkamp, S. Werner, M. C. Borges, D. Ernst, and D. Wenzel. 2020. Benchmarking elasticity of faas platforms as a foundation for objective-driven design of serverless applications. In *Proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing*.
[87] C. Lee, M. Lin, C. Yang, and Y. Chen. 2019. IoTBench: A benchmark suite for intelligent Internet of Things edge devices. In *Proceedings of the IEEE International Conference on Image Processing*. 170–174.
[88] X. Li, J. Wan, H.-N. Dai, M. Imran, M. Xia, and A. Celesti. 2019. A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing. *IEEE Trans. Industr. Inf.* 15, 7 (2019), 4225–4234.
[89] L. A. Libutti, F. D. Igual, L. Pinuel, L. De Giusti, and M. Naiouf. 2020. Benchmarking performance and power of USB accelerators for inference with MLPerf. In *Proceedings of the 2nd Workshop on Accelerated Machine Learning*.
[90] A. Limaye and T. Adegbija. 2018. HERMIT: A benchmark suite for the Internet of Medical Things. *IEEE IoT J.* 5, 5 (2018), 4212–4222.
[91] Y. Lin, B. Kemme, M. Patino-Martinez, and R. Jimenez-Peris. 2007. Enhancing edge computing with database replication. In *Proceedings of the IEEE International Symposium on Reliable Distributed Systems*. IEEE, 45–54.
[92] Z. Liu, T. Bicer, R. Kettimuthu, and I. Foster. 2019. Deep learning accelerated light source experiments. In *Proceedings of the IEEE/ACM 3rd Workshop on Deep Learning on Supercomputers*. 20–28.
[93] Z. Liu, T. Bicer, R. Kettimuthu, D. Gursoy, F. De Carlo, and I. Foster. 2019. Tomogan: Low-dose X-ray tomography with generative adversarial networks. *arXiv:1902.07582*. Retrieved from https://arxiv.org/abs/1902.07582.
[94] M. M. Lopes, W. A. Higashino, M. A. M. Capretz, and L. F. Bittencourt. 2017. MyiFogSim: A simulator for virtual machine migration in fog computing. In *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*. 47–52.
[95] C. Luo, J. Zhan, Z. Jia, L. Wang, G. Lu, L. Zhang, and N. Sun C. Z. Xu, 3. 2012. CloudRank-D: Benchmarking and ranking cloud computing systems for data processing applications. *Front. Comput. Sci.* 6, 4 (2012), 347–362.
[96] C. Luo, F. Zhang, C. Huang, X. Xiong, J. Chen, L. Wang, W. Gao, H. Ye, T. Wu, R. Zhou, and J. Zhan. 2019. AIoT bench: Towards comprehensive benchmarking mobile and embedded device intelligence. In *Benchmarking, Measuring, and Optimizing*, C. Zheng and J. Zhan (Eds.). Springer International Publishing, 31–35.
[97] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi. 2006. The HPC challenge (HPCC) benchmark suite. In *Proceedings of the ACM/IEEE Conference on Supercomputing*.
[98] L. Ma, S. Yi, and Q. Li. 2017. Efficient service handoff across edge servers via Docker container migration. In *Proceedings of the ACM/IEEE Symposium on Edge Computing*. 1–13.
[99] A. Madej, N. Wang, N. Athanasopoulos, R. Ranjan, and B. Varghese. 2020. Priority-based fair scheduling in edge computing. In *Proceedings of the International Conference on Fog and Edge Computing*.

[100] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft. 2013. Unikernels: Library operating systems for the cloud. *ACM SIGARCH Comput. Arch. News* 41, 1 (2013), 461–472.

[101] J. D. McCalpin. 1995. Memory bandwidth and machine balance in current high performance computers. *IEEE Techn. Commit. Comput. Arch. Newslett.* 2 (1995).

[102] J. McChesney, N. Wang, A. Tanwer, E. de Lara, and B. Varghese. 2019. DeFog: Fog computing benchmarks. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. 47–58.

[103] S. Mittal. 2018. Survey of FPGA-based accelerators for convolutional neural networks. *Neural Comput. Appl.* 32, 4 (2018), 1–31.

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

[105] R. Morabito, V. Cozzolino, A. Ding, N. Beijar, and J. Ott. 2018. Consolidate IoT edge computing with lightweight virtualization. *IEEE Network* 32, 1 (2018), 102–111.

[106] S. Müller, D. Bermbach, S. Tai, and F. Pallas. 2014. Benchmarking the performance impact of transport layer security in cloud database systems. In *Proceedings of the 2nd International Conference on Cloud Engineering*. IEEE.

[107] Raul Muñoz, Laia Nadal, Ramon Casellas, Michela Svaluto Moreolo, Ricard Vilalta, Josep Maria Fàbrega, Ricardo Martínez, Arturo Mayoral, and Fco. Javier Vílchez. 2017. The ADRENALINE testbed: An SDN/NFV packet/optical transport network and edge/core cloud platform for end-to-end 5G and IoT services. In *Proceedings of the European Conference on Networks and Communications*. 1–5.

[108] F. Nadeem, R. Prodan, T. Fahringer, and A. Iosup. 2008. *Benchmarking Grid Applications*. Springer US, Boston, MA, 19–37.

[109] R. Nambiar, M. Poess, A. Dey, P. Cao, T. Magdon-Ismail, D. Q. Ren, and A. Bond. 2015. Introducing TPCx-HS: The first industry standard for benchmarking big data systems. In *Performance Characterization and Benchmarking. Traditional to Big Data*, R. Nambiar and M. Poess (Eds.). Springer International Publishing, 1–12.

[110] M. O. J. Olguín Muñoz, J. Wang, M. Satyanarayanan, and J. Gross. 2019. EdgeDroid: An experimental approach to benchmarking human-in-the-loop applications. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. 93–98.

[111] T. Palit, Yongming Shen, and M. Ferdman. 2016. Demystifying cloud benchmarking. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*. 122–132.

[112] F. Pallas, D. Bermbach, S. Müller, and S. Tai. 2017. Evidence-based security configurations for cloud datastores. In *Proceedings of the the 32nd ACM Symposium on Applied Computing*.

[113] F. Pallas, J. Günther, and D. Bermbach. 2017. Pick your choice in HBase: Security or performance. In *Proceedings of the IEEE International Conference on Big Data*.

[114] F. Pallas, P. Raschke, and D. Bermbach. 2020. Fog computing as privacy enabler. *IEEE Internet Comput.* 24, 4 (2020), 15–21.

[115] S. Patil, M. Polte, K. Ren, W. Tantisiriroj, L. Xiao, J. López, G. Gibson, A. Fuchs, and B. Rinaldi. 2011. YCSB++: Benchmarking and performance debugging advanced features in scalable table stores. In *Proceedings of the 2nd Symposium on Cloud Computing*. Article 9, 9:1–9:14 pages.

[116] T. Rabl, M. Sadoghi, H.-A. Jacobsen, S. Gómez-Villamor, V. Muntés-Mulero, and S. Mankovskii. 2012. Solving big data challenges for enterprise application performance management. *Proc. VLDB Endow.* 5, 12 (2012).

[117] N. Rajovic, A. Rico, N. Puzovic, C. Adeniyi-Jones, and A. Ramirez. 2014. Tibidabo: Making the case for an ARM-based HPC system. *Fut. Gener. Comput. Syst.* 36 (2014), 322–334.

[118] F. Ramalho and A. Neto. 2016. Virtualization at the network edge: A performance comparison. In *Proceedings of the IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks*. 1–6.

[119] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, and W. Chou et al. 2019. MLPerf inference benchmark. *arXiv:1911.02549*. Retrieved from https://arxiv.org/abs/1911.02549.

[120] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner. 2019. Survey and benchmarking of machine learning accelerators. In *Proceedings of the 24th Annual IEEE High Performance Extreme Computing Conference*.

[121] B. P. Rimal, M. Maier, and M. Satyanarayanan. 2018. Experimental testbed for edge computing in fiber-wireless broadband access networks. *IEEE Commun. Mag.* 56, 8 (2018), 160–167.

[122] D. Roy, D. De, A. Mukherjee, and R. Buyya. 2016. Application-aware cloudlet selection for computation offloading in multi-cloudlet environment. *J. Supercomput.* 73, 4 (2016), 1672–1690.

[123] M. Ryden, K. Oh, A. Chandra, and J. Weissman. 2014. Nebula: Distributed edge cloud for data intensive computing. In *Proceedings of the IEEE International Conference on Cloud Engineering*. 57–66.

[124] S. Sardellitti, G. Scutari, and S. Barbarossa. 2015. Joint optimisation of radio and computational resources for multicell mobile-edge computing. *IEEE Trans. Sign. Inf. Process. Netw.* 1, 2 (2015), 89–103.

[125] M. Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (2017), 30–39.

[126] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. 2009. The case for VM-based cloudlets in mobile computing. *IEEE Perv. Comput.* 8, 4 (2009), 14–23.
[127] B. Schroeder, A. Wierman, and M. Harchol-Balter. 2006. Open versus closed: A cautionary tale. In *Proceedings of the 3rd Conference on Networked Systems Design & Implementation*.
[128] M. Schuundefined, C. A. Boano, M. Weber, and K. Römer. 2017. A competition to push the dependability of low-power wireless protocols to the edge. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks*. 54–65.
[129] L. Shao, J. Zhao, T. Xie, L. Zhang, B. Xie, and H. Mei. 2009. User-perceived service availability: A metric and an estimation approach. In *Proceedings of the IEEE International Conference on Web Services*. 647–654.
[130] S. Sharma, C.-H. Hsu, and W.-C. Feng. 2006. Making a case for a Green500 list. In *Proceedings of the 20th IEEE International Parallel Distributed Processing Symposium*.
[131] S. Shastri, V. Banakar, M. Wasserman, A. Kumar, and V. Chidambaram. 2019. Understanding and benchmarking the impact of GDPR on database systems. *arXiv:1910.00728*. Retrieved from https://arxiv.org/abs/1910.00728.
[132] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. 2016. Edge computing: Vision and challenges. *IEEE IoT J.* 3, 5 (2016), 637–646.
[133] A. Shukla, S. Chaturvedi, and Y. Simmhan. 2017. RIoTBench: An IoT benchmark for distributed stream processing systems. *Concurr. Comput.: Pract. Exp.* 29, 21 (2017), e4257.
[134] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner. 2017. Optimized IoT service placement in the fog. *Serv. Orient. Comput. Appl.* 11, 4 (2017), 427–443.
[135] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar. 2017. Towards QoS-aware fog service placement. In *Proceedings of the IEEE International Conference on Fog and Edge Computing*. IEEE. 89–96.
[136] A. Snavely, G. Chun, H. Casanova, R. F. Van der Wijngaart, and M. Frumkin. 2003. Benchmarks for grid computing: A review of ongoing efforts and future directions. *SIGMETRICS Perf. Eval. Rev.* 30, 4 (2003), 27–32.
[137] C. Sonmez, A. Ozgovde, and C. Ersoy. 2017. EdgeCloudSim: An environment for performance evaluation of edge computing systems. In *Proceedings of the International Conference on Fog and Mobile Edge Computing*. 39–44.
[138] C. Sonmez, A. Ozgovde, and C. Ersoy. 2019. Fuzzy workload orchestration for edge computing. *IEEE Trans. Netw. Serv. Manage.* 16, 2 (2019), 769–782.
[139] S. Sridhar and M. E. Tolentino. 2017. Evaluating voice interaction pipelines at the edge. In *Proceedings of the IEEE International Conference on Edge Computing*. 248–251.
[140] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1–9.
[141] M. Toeroe and F. Tam. 2012. *Service Availability: Principles and Practice*. John Wiley & Sons.
[142] F. P. Tso, D. R. White, S. Jouet, J. Singer, and D. P. Pezaros. 2013. The Glasgow Raspberry Pi Cloud: A scale model for cloud computing infrastructures. In *Proceedings of the 1st International Workshop on Resource Management of Cloud Computing*. 108–112.
[143] G. Tsouloupas and M. D. Dikaiakos. 2003. GridBench: A tool for benchmarking grids. In *Proceedings of the 4th International Workshop on Grid Computing*.
[144] B. Varghese, O. Akgun, I. Miguel, L. Thai, and A. Barker. 2014. Cloud benchmarking for performance. In *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science*. 535–540.
[145] B. Varghese, O. Akgun, I. Miguel, L. Thai, and A. Barker. 2019. Cloud benchmarking for maximising performance of scientific applications. *IEEE Trans. Cloud Comput.* 7, 1 (2019), 170–182.
[146] B. Varghese and R. Buyya. 2018. Next generation cloud computing: New trends and research directions. *Fut. Gener. Comput. Syst.* 79, 3 (2018), 849–861.
[147] B. Varghese, P. Leitner, S. Ray, K. Chard, A. Barker, Y. Elkhatib, H. Herry, C. Hong, J. Singer, F. P. Tso, E. Yoneki, and M. Zhani. 2019. Cloud futurology. *Computer* 52, 9 (2019), 68–77.
[148] B. Varghese, C. Reaño, and F. Silla. 2018. Accelerator virtualization in fog computing: Moving from the cloud to the edge. *IEEE Cloud Comput.* 5, 6 (2018), 28–37.
[149] B. Varghese, L. T. Subba, L. Thai, and A. Barker. 2016. Container-based cloud virtual machine benchmarking. In *Proceedings of the IEEE International Conference on Cloud Engineering*. 192–201.
[150] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos. 2016. Challenges and opportunities in edge computing. In *Proceedings of the IEEE International Conference on Smart Cloud*. 20–26.
[151] L. Villas, A. Boukerche, H. De Oliveira, R. De Araujo, and A. Loureiro. 2014. A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Netw.* 12 (2014), 69–85.
[152] H. Wada, A. Fekete, L. Zhao, K. Lee, and A. Liu. 2011. Data consistency properties and the trade-offs in commercial cloud storages: The consumers' perspective. In *Proceedings of the 5th Conference on Innovative Data Systems Research*. 134–143.

[153] J. Wang, Z. Feng, S. George, R. Iyengar, P. Pillai, and M. Satyanarayanan. 2019. Towards scalable edge-native applications. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. 152–165.

[154] N. Wang, M. Matthaiou, D. S. Nikolopoulos, and B. Varghese. 2020. DYVERSE: DYnamic VERtical scaling in multi-tenant edge environments. *Fut. Gener. Comput. Syst.* 108 (2020), 598–612.

[155] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos. 2020. ENORM: A framework for edge node resource management. *IEEE Trans. Serv. Comput.* 13, 6 (2020), 1086–1099.

[156] Y. Wang, S. Liu, X. Wu, and W. Shi. 2018. CAVBench: A benchmark suite for connected and autonomous vehicles. In *Proceedings of the IEEE/ACM Symposium on Edge Computing*. 30–42.

[157] R. P. Weicker. 1984. Dhrystone: A synthetic systems programming benchmark. *Commun. ACM* 27, 10 (1984), 1013–1030.

[158] H. Wu, F. Liu, and R. B. Lee. 2016. Cloud server benchmark suite for evaluating new hardware architectures. *IEEE Comput. Arch. Lett.* (2016).

[159] Y. Xiao and M. Krunz. 2017. QoE and power efficiency tradeoff for fog computing networks with fog node cooperation. In *Proceedings of the IEEE Conference on Computer Communications*. IEEE, 1–9.

[160] X. Yang, X. Li, Y. Ji, and M. Sha. 2008. CROWNBench: A grid performance testing system using customizable synthetic workload. In *Progress in WWW Research and Development*, Y. Zhang, G. Yu, E. Bertino, and G. Xu (Eds.). Springer, Berlin, 190–201.

[161] Y. Yao, Q. Cao, and A. Vasilakos. 2013. EDAL: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for wireless sensor networks. In *Proceedings of the IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*. 182–190.

[162] C. You, K. Huang, H. Chae, and B.-H. Kim. 2016. Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Trans. Wireless Commun.* 16, 3 (2016), 1397–1411.

[163] K. Zellag and B. Kemme. 2012. How consistent is your cloud application? In *Proceedings of the 3rd Symposium on Cloud Computing*. Article 6, 6:1–6:14 pages.

[164] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang. 2017. Optimal delay constrained offloading for vehicular edge computing networks. In *Proceedings of the IEEE International Conference on Communications*. IEEE, 1–6.

[165] X. Zhang, Y. Wang, and W. Shi. 2018. pCAMP: Performance comparison of machine learning packages on the edges. In *Proceedings of the USENIX Workshop on Hot Topics in Edge Computing*.

[166] J. Zhao, Q. Li, Y. Gong, and K. Zhang. 2019. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. *IEEE Trans. Vehic. Technol.* 68, 8 (2019), 7944–7956.

[167] P. Zhao, H. Tian, C. Qin, and G. Nie. 2017. Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. *IEEE Access* 5 (2017), 11255–11268.

Received April 2020; revised December 2020; accepted December 2020