A Survey on Edge Benchmarking

Blesson Varghese[a], Nan Wang[b], David Bermbach[c], Cheol-Ho Hong[d], Eyal de Lara[e], Weisong Shi[f], and Christopher Stewart[g]

[a]Queen's University Belfast, UK; [b]Durham University, UK; [c]TU Berlin, Germany, and Einstein Center Digital Future, Germany; [d]Chung-Ang University, S. Korea; [e]University of Toronto, Canada; [f]Wayne State University, USA; [g]Ohio State University, USA

This manuscript was compiled on April 24, 2020

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

Edge computing | Edge benchmarking | System under test | Benchmarking techniques | Quality | Benchmark runtime

1. Introduction

The computing landscape has significantly changed over the last three decades – loosely coupled and geographically dispersed systems started replacing tightly coupled monolithic systems (1). 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 then offered the unique capability of processing large datasets near the source without requiring to transfer data of a distributed workflow to a central system (2). Subsequently, computing became a utility offered remotely through the cloud (3).

While the cloud is the current computing model for most 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 data generated from these cannot be transferred and processed in geographically distant cloud data centers without incurring delays (4). The next disruption in the computing landscape therefore is to further distribute infrastructure resources, and subsequently, application services to bring compute closer to the edge of the network near the data source (5). In this article, we use the term 'edge computing' to refer to the use of resources located at the edge of the network, such as routers and gateways or dedicated micro data centers, to either provide applications with acceleration by co-hosting services along with the cloud or by hosting them natively on edge resources (6).

The inclusion of edge resources for computing creates a large-scale, geographically dispersed and resource-rich distributed system that spans multiple technological domains and ownership boundaries. Such a complex system will be transient – the resources, their availability and characteristics change over time. For example, an edge resource previously available to an application may be unavailable due to a recent fault or the operating system of a target resource may have changed during maintenance (7, 8). In this context, it is essential to address the challenge of understanding the relative performance of applications by comparing target hardware platforms from different vendors due to diversity of hardware architectures and the impact on performance when system software level changes or new networking protocols are introduced (9). These are considered in ***edge benchmarking*** (10, 11).

Benchmarking is the process of creating stress on a system while closely observing its response on a wide-range of quality metrics. Typically, synthetic or application-driven workloads are executed against a system under test, for example, virtual machines, storage systems, stream processing systems, or specific application components, while measuring quality behaviors such as I/O throughput, data staleness, or end-to-end communication or computation latency. In contrast to alternative approaches, such as predictive methods or simulation, insight into real system behavior is obtained by replicating the analyzed conditions of a production deployment (12).

The focus of this article is three-fold. Firstly, it traces the history of the development of benchmarks over the last three decades for high-performance computing (HPC), grid, and cloud systems. Secondly, it catalogs and examines different edge benchmarks. Finally, the system under test, techniques analyzed, quality, and benchmark runtime that underpin edge benchmarking are reviewed.

Figure 1 shows a histogram of the total number of research publications reviewed by this article between 1976 and 2020 under the categories: (i) books and book chapters, (ii) reports, including pre-print articles or white papers, and doctoral research thesis, (iii) conference or workshop papers, and (iv) journal or magazine articles. More than 85% of the articles reviewed have been published since 2010 and more than 65% of the articles since 2015.

The remainder of this article is organized as follows. Section 2 provides a brief history of benchmarking. Section 3 catalogs different edge benchmarks. Section 4 presents a review of the system under test in edge benchmarking. Section 5 reviews the techniques analyzed in edge benchmarking. Section 6 highlights the quality aspect of edge benchmarking. Section 7 surveys the runtime execution environment and deployments in edge benchmarking. Section 8 suggests future directions for furthering research in edge benchmarking and concludes the paper.

[a]Corresponding e-mail: b.varghese@qub.ac.uk

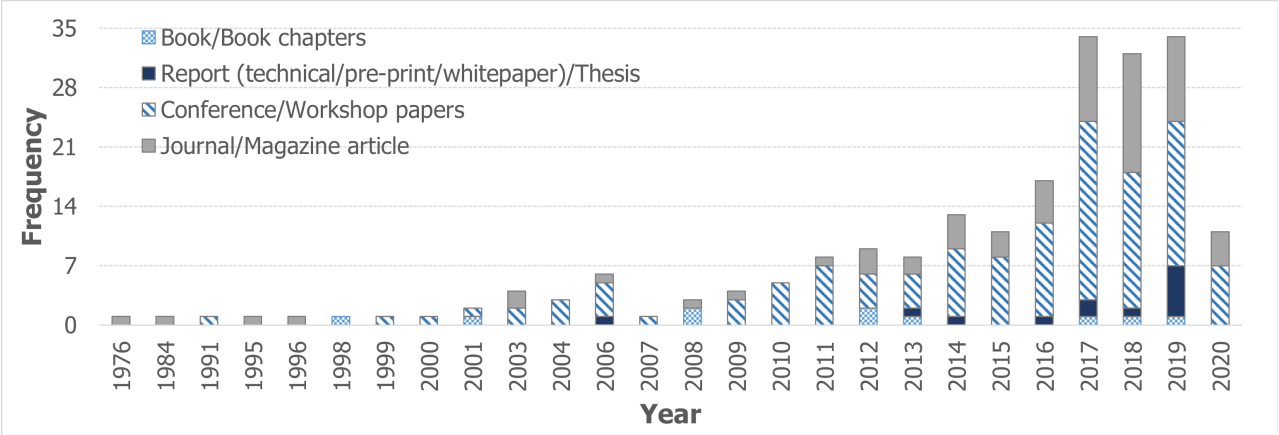

Fig. 1. A histogram of the research publications reviewed by this article.

2. Background

Benchmarks have played an important role in the evolution of the computing landscape and are an important area of research and development. CPU or processor related benchmarks already existed in the 1970s. For example, the Whetstone benchmark was developed to measure the floating-point arithmetic performance (13). Dhrystone is another benchmark developed in the 1980s to represent performance of integers (14).

This section provides a brief history of the progression of the area of benchmarking of distributed systems over the last three decades as shown in Figure 2. More specifically, both tightly coupled HPC clusters and supercomputers (highlighted in green color), as well as more loosely coupled infrastructure, such as the grid (shown in bronze color) and the cloud (indicated in turquoise color) are considered. The next section will consider edge computing benchmarks (highlighted in red color), which is the focus of this article.

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

A. HPC benchmarking. In 1979, the LINPACK benchmark was developed, which evolved into the High Performance LINPACK (HPL) (15). 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[1] list launched in 1993.

The NAS Parallel Benchmarks (NPB) were launched in 1991 and are based on computational fluid dynamics (CFD) applications (16). The Standard Performance Evaluation Corporation (SPEC) launched HPC benchmarks in 1996 (17). They comprised three industry 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 (18). This resulted in the launch of the Green500[2] project in 2005 that captured floating point rate of execution with respect to power. A methodology for measuring and reporting the power used by an HPC system was developed[3]. In 2016, Green500 was integrated with the Top500 project.

The HPC Challenge benchmark was launched in 2005 to measure performance and productivity (19), which includes the STREAM benchmark to measure sustainable memory bandwidth (20).

The area commonly known as big data became popular in the context of HPC clusters, and hence the GridMix[4] benchmark for Hadoop clusters emerged in 2007.

The High Performance Conjugate Gradient (HPCG) is a relatively newer benchmark launched in 2013 to rank supercomputers and clusters in a more balanced way (21). The performance of this benchmark is influenced by memory bandwidth.

In 2016, benchmarks utilized on the UK's national supercomputer ARCHER were presented[5]. They are a combination of real applications (DFT, molecular mechanics-based, CFD, and climate modeling) developed by the UK Met Office and synthetic benchmarks from HPC Challenge.

As HPC has become more heterogeneous with the advent of hardware accelerators, such as GPUs, additional benchmarks began to emerge. These include the RODINIA (22) and SHOC GPU benchmarks (23) and the more recent Mirovia benchmarks (24).

B. Grid Benchmarking. With academic and research organizations connecting clusters of computers, geographically dispersed and heterogeneous grids became popular for scientific computing. Key metrics relevant to grid benchmarking included turnaround time and throughput since data originated from different geographic locations in a scientific workflow that was executed on grids (25).

[1] https://www.top500.org/

[2] https://www.top500.org/green500/

[3] https://www.top500.org/static/media/uploads/methodology-2.0rc1.pdf

[4] https://hadoop.apache.org/docs/r1.2.1/gridmix.html

[5] https://www.archer.ac.uk/documentation/white-papers/benchmarks/UK_National_HPC_Benchmarks.pdf

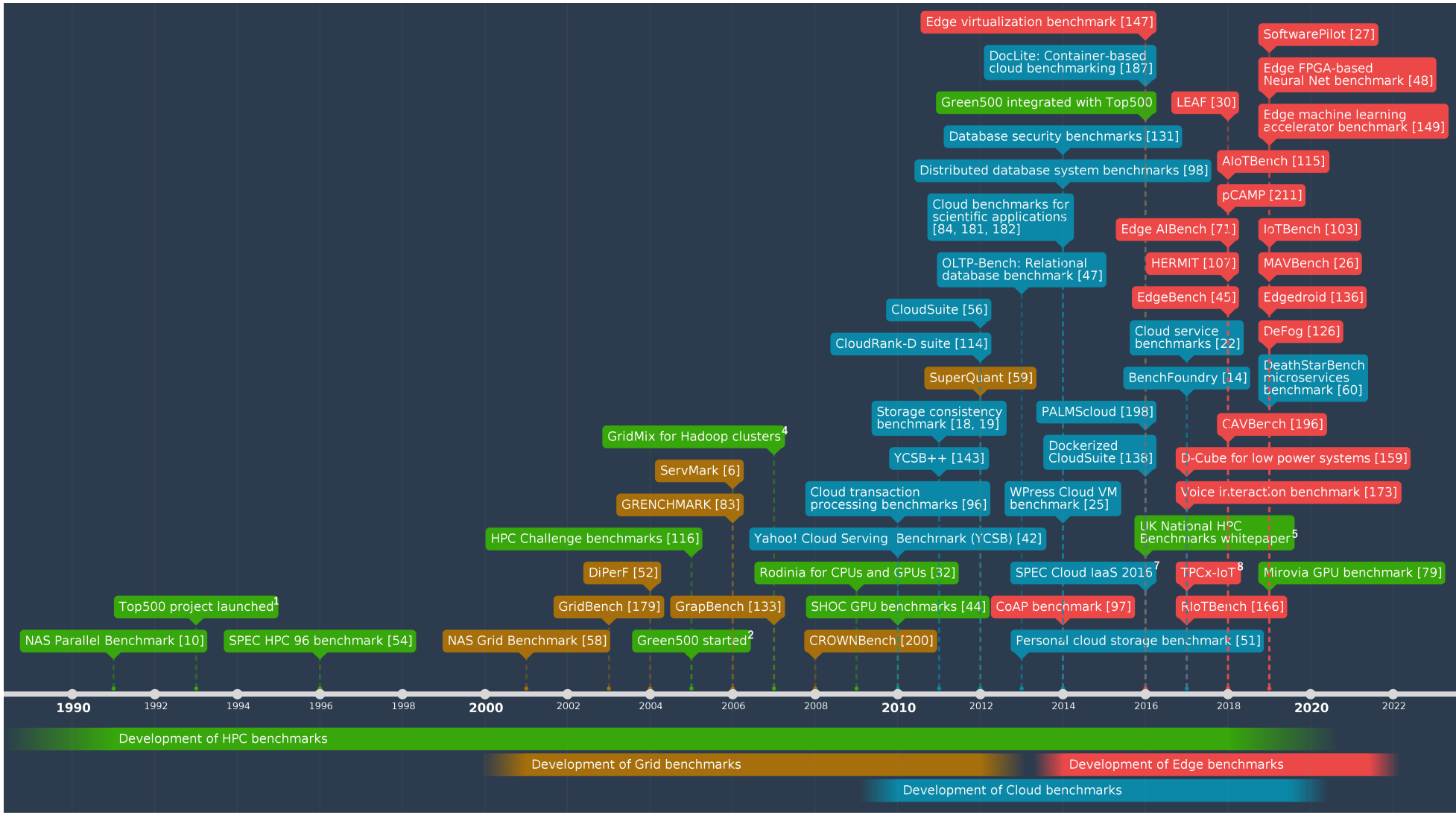

Fig. 2. A brief history of the development of benchmarking for HPC, grid, cloud and edge systems.

One of the first benchmarks to capture the compute performance on grids was the GridNPB in 2001, which was the NPB distribution that used the Globus grid middleware (26).

In 2003, GridBench which allowed benchmarks to be specified using a definition language that was compiled into specification languages supported by grid middleware became available (27).

In 2004, the DiPerF benchmarking tool was used to generate performance statistics of grids for predicting grid performance (28).

GRENCHMARK was developed for generating synthetic workloads used for benchmarking grids (29). This approach was adopted in ServMark for automating the benchmarking pipeline (30).

Additional benchmarks, such as GrapBench brought flexibility to the benchmarking approach by considering variations in the problem and machine sizes of applications and the grid respectively (31). CROWNBench also generated synthetic workloads for benchmarking (32). Domain specific benchmarks, such as those relevant to financial workloads, were developed (33).

C. Cloud Benchmarking. Benchmarking web-based systems, including open, closed, and partially closed systems were considered (34). One of the earliest benchmarks for web-based servers was SPECweb96[6]. In 2010, the first cloud-specific benchmarks were released – the Yahoo! Cloud Service Benchmark (YCSB) (35) and transaction processing benchmarks (36). YCSB++ is an extension to YCSB that benchmarks scalable column stores (37). Approaches for benchmarking scientific applications on the cloud (38–40) and on VMs (41) were proposed. The aspect of fairness in benchmarking was also considered (42). Benchmarks for consistency of cloud storage services were developed (43, 44).

Other early attempts in developing cloud benchmarks between 2012 and 2014, include CloudRank-D (45) and CloudSuite (46) for resources and services. Benchmarks for personal cloud storage (47), relational databases (48), security of databases (49) and the scalability and elasticity of distributed databases were developed (50).

The SPEC benchmarks for infrastructure-as-a-service cloud were launched in 2016 (SPEC Cloud IaaS 2016[7]). During this time, container-based benchmarking suites, namely, DocLite (51) and a containerized version of CloudSuite (52) emerged. New cloud hardware architectures can be evaluated using the PALMScloud benchmark suite (53) and a framework for benchmarking cloud storage services was introduced (54). Also, a comparison of various benchmark suites for measuring the quality of cloud services from a client perspective was presented (12) which also, more recently, led to benchmarks for microservices on distributed clouds (55, 56).

3. Edge Benchmarking

This section start by highlighting the developments in edge benchmarking and then provides a classification for understanding the research undertaken in the area of edge benchmarking.

As shown in Figure 2, edge benchmarking has developed rapidly since 2015. These benchmarks cover a wide range of resources including (i) end-devices, namely Internet-of-Things sensors, smartphones, user gadgets, including wearables, (ii) compute resources located at the edge of the wired network, including routers, switches, gateways, or dedicated resources, including embedded computers or micro clouds, and (iii) cloud resources, capturing their performance in isolated and networked execution contexts.

Table 1 summarizes the benchmark type (micro/macro), application domain, benchmarks used, and the destination platforms (device, edge, and cloud) of 21 edge benchmarking techniques or suites. Micro benchmarks refer to those benchmarks that capture system-level (CPU, memory, network, storage) performance metrics. On the other hand, macro benchmarks refer to those benchmarks that capture application-specific system performance for different application domains. Only two benchmarks capture both micro and macro benchmarks, namely RIoTBench (57) and AIoTBench (58). The majority of benchmarks are macro benchmarks and only two comprise generic workloads, namely EdgeBench (59) and DeFog (9). It is also noted that only five benchmarks can capture the performance of the computation pipeline comprising the device, edge and cloud. This is potentially due to the lack of readily available large-scale test-beds (although a few are available) for experimentation that integrate the cloud and the edge for end-users.

Existing research is classified across four dimensions as follows to explain edge benchmarking (as shown in Figure 3):

- *System under test* refers to the hardware infrastructure and software platforms that are benchmarked. This is considered in Section 4.
- *Techniques analyzed* refers to the resource allocation, offloading, and deployment options analyzed by edge benchmarks which are presented in Section 5.
- *Quality* includes quality indicators, metrics, and trade-offs in capturing non-functional properties through edge benchmarking which is discussed in Section 6.
- *Benchmark runtime* refers to the software and data characteristics of the execution environment and the deployment destination, including test-beds which is considered in Section 7.

A large body of research on edge benchmarking considered in this article relies on either trace data obtained from simulators or simulators 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 testbeds become readily available.

4. System Under Test

This section categorizes the system under test in edge benchmarking into two classes, namely infrastructure and software platforms. The applications (macro and micro) used to test these systems are captured in Table 1 under 'Domain' and 'Benchmarks used'. However, it is noted that edge specific application and micro benchmarks are not readily available.

A. Infrastructure. Embedded CPUs, accelerators, memory, and storage hardware are the infrastructure resources considered in edge benchmarking.

[6] https://www.spec.org/web96/

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

Table 1. A 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 (64)	Micro	-	lmbench	D
2016	Virt. benchmark (61)	Micro	-	NBench, Sysbench, HPL, bonnie++, DD, Stream, Netperf	D
2017	Virt. benchmark (63)	Micro	-	Sysbench, mbw, fio, iperf, dstat	D
	Voice benchmark (65)	Macro	Voice	Mycroft platform	D, E, C
	RIoTBench (57)	Micro	Stream processing	27 IoT steam processing tasks	D, C
		Macro		Transform and load, Statistical summarization, Predictive analytics	
	TPCx-IoT[10]	Macro	Power grid	Data ingestion and concurrent queries (based on YCSB)	D, E, C
	D-Cube (66)	Micro	Low power systems	6 wireless protocols	D
2018	CAVBench (67)	Macro	Autonomous vehicles	SLAM, Object tracking, Battery diagnostics, Speech recognition, Video analytics	E
	EdgeBench (59)	Macro	Generic	Speech-to-text, Image recognition, Scalar sensor	E, C
	Edge AIBench (68)	Macro	Machine learning	Patient monitor, Surveillance camera, Speech/facial and road sign recognition	D, E, C
	HERMIT (69)	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 (70)	Macro	Federated learning	Image classification, Sentiment analysis, Text prediction	E
	AIoTBench (58)	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 (71)	Macro	Machine learning	TensorFlow, Caffe2, MXNet, PyTorch, TensorFlow Lite	D, E, C
2019	DeFog (9)	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 (72)	Macro	Human-in-the-loop applications	Cognitive assistance application for assembling LEGO	D, E
	MAVBench (73)	Macro	Micro aerial vehicles	Scanning, Aerial photography, Package delivery, 3D mapping, Search and rescue	D (Drone), C
	IoTBench (74)	Macro	Vision and speech	Video summarization, Stereo image matching, Image recognition, Scan matching, Voice feature extraction, Signals enhancement, Data compression	D
	SoftwarePilot[11] (75)	Macro	Unmanned aerial vehicle	Aerial photography, Crop surveillance, Rescue and discovery, Facial recognition	E, C
	Machine learning accelerator benchmark (76)	Macro	Low power machine learning accelerators	Mobilenet on TensorFlow and OpenVINO	D, E, C
	Edge FPGA-based Neural Net benchmark (77)	Macro	Neural network	Keyword spotting application	E

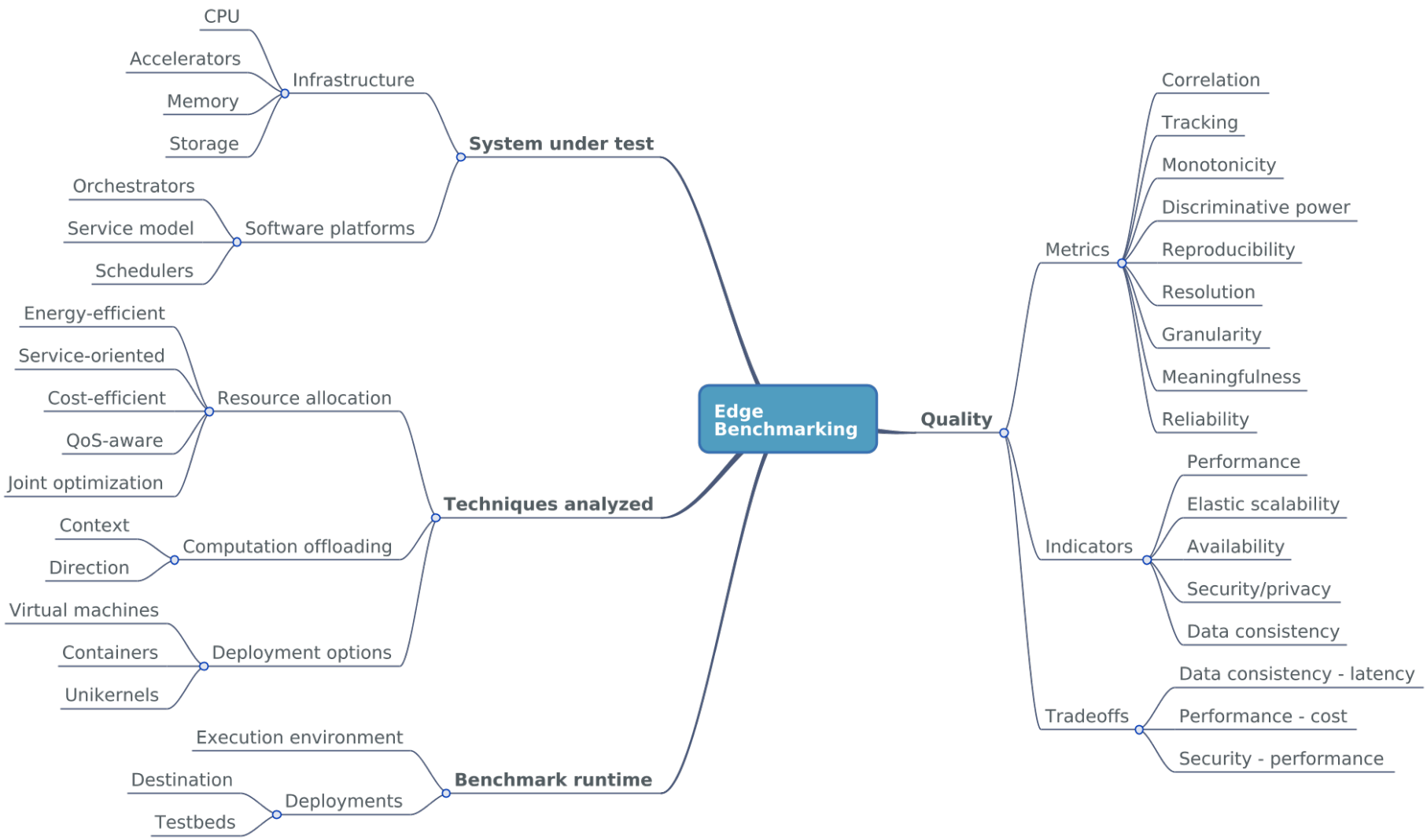

Fig. 3. A classification of edge benchmarking under four dimensions, namely system under test, techniques analyzed, quality, and benchmark runtime.

A.1. CPUs. Early edge computing utilized single-board computers (SBCs) such as Raspberry Pi (10) as edge devices. An SBC is a circuit board comprising a CPU, memory, network, storage, and other components. Most SBCs adopt ARM processors as CPUs due to low cost and low power characteristics. In addition, the performance of modern ARM processors is comparable with general purpose CPUs (60). An ARM Cortex-A7 Dual-Core processor hosted by Cubieboard2 is benchmarked (61) using NBench[8], sysbench[9], and High-Performance Linpack (HPL) benchmark (62). A recent study benchmarked a wide range of ARM-based SBCs including Raspberry Pi 2 model B (RPi2), Raspberry Pi 3 model B (RPi3), Odroid C1+ (OC1+), Odroid C2 (OC2), and Odroid XU4 (OXU4) (63). sysbench was used in order to stress the CPU. In terms of CPU performance, Odroid C2 with Quad-core 2 GHz ARM v8 Cortex-A53 outperforms other tested SBCs. With regard to power consumption, Raspberry Pi boards, OC1+, and OC2 achieved high energy efficiency whereas OXU4 consumed 3 – 7 times more power than others.

A.2. Accelerators. The use of hardware accelerators have been proposed for edge computing (78). For running deep neural networks on edge devices, low power and purpose-built accelerators such as Intel Movidius Myriad X VPU (Vision Processing Unit) (79), Google Edge TPU (80), NVIDIA 128-core Maxwell and 256-core Pascal architecture-based GPU, and custom field-programmable gate arrays (FPGAs) were introduced (76).

Intel's VPU uses a dedicated accelerator called the Neural Compute Engine for on-device deep neural networks inferences that is optimized for machine vision. The accelerator consists of multiple SHAVE (Streaming Hybrid Architecture Vector Engine) cores, which is optimized for machine vision, and a high throughput memory fabric (81). The Intel Neural Compute Stick (NCS) is a USB dongle that includes the Intel VPU.

Google Edge TPU is the edge version of Google's Tensor Processing Unit (TPU), which is a custom-developed application-specific integrated circuits (ASICs) for accelerating machine learning workloads (82). Edge TPUs are power efficient, supports TensorFlow Lite, and can execute deep feed-forward neural networks (DFF) (83). The Coral Dev Board is an SBC that uses the Edge TPU, and the Coral USB Accelerator is a USB dongle that can be connected to other SBCs.

The NVIDIA 128-core Maxwell GPU is hosted by NVIDIA Jetson Nano, an SBC with one Quad-core ARM A57. This small-core GPU delivers up to 472 GFLOPS performance for machine learning workloads with low power consumption. The NVIDIA 256-core Pascal GPU delivers up to 1.3 TFLOPS and is hosted by NVIDIA Jetson TX2 for more performance-oriented inference tasks. NVIDIA Jetson TX2 has one Denver 64-bit CPU and one Quad-Core A57.

Custom FPGAs optimize performance of specific neural network applications (77, 84).

Performance Comparisons: The Intel NCS 2 and Google's Coral USB accelerators were compared with popular inference workloads in the MLPerf benchmark (85), which are MobileNet-v1 (86) and Inception-v1 (87), in terms of inference time and energy efficiency (88). The research modified the MLPerf benchmark to support each accelerator. It developed the Tensorflow Lite backend for Google Coral and the OpenVINO backend for Intel NCS 2. The Multi-stream mode in the MLPerf benchmark is used to periodically send a set of inference requests every 50 to 100 ms. The research shows that Google's Coral is 3.2 – 3.7 times faster with MobileNet-v1 and 1.5 – 1.9 with Inception-v1 than the Intel NCS 2. In addition, Google Coral achieves roughly 6.1 – 7.6 times better energy efficiency for MobileNet-v1 and 4.0 – 4.9 for Inception-v1 than the Intel NCS 2.

Inference at the edge may sacrifice accuracy because original models, which are developed on devices that support 32-bit floating point arithmetic operations, are adjusted to edge devices that support lower-precision integer arithmetic operations. This is called quantization. To explore whether a scientific deep learning model can provide sufficient accuracy on edge devices, TomoGAN (89, 90), an algorithm to enhance the quality of X-ray images, was adapted to run on the Google Edge TPU and NVIDIA Jetson TX2 (91). The study observes that the Coral Dev Board and the Coral USB Accelerator achieve three times faster inference performance than an Intel Core i7- 6700HQ 2.6 GHz CPU, and the Coral USB Accelerator is 1.5 times faster than NVIDIA Jetson TX2. The authors claim that the edge accelerators can deliver sufficient accuracy with a new shallow CNN called Fine-Tune network.

A recent study benchmarked the Google Edge TPU, the NVIDIA 128-core Maxwell GPU, and the Intel Movidius Myriad X VPU by running eight deep learning models used in personal-scale sensory systems (92). The eight deep learning models include Aroma (93), Audio Classification (94), DKWS (95), SqueezeNet-v1 (96), MobileNet-v1 (86), EfficientNet-EdgeTPU (97), Inception-v1 (87), and DenseNet121 (98). The benchmark results show that the Google Edge TPU outperforms other accelerators on the eight models. For example, the Google Edge TPU takes 2 ms for one execution of SqueezeNet-v1 whereas it takes 11 ms for the Intel Myriad X VPU and 26 ms for the NVIDIA Maxwell GPU. In term of energy efficiency, the Google Edge TPU utilizes less than 10 mJ for a single execution on the eight models whereas other platforms consume between 5 mJ to 274 mJ depending on the model.

Various FPGA platforms are evaluated by a custom benchmark that implements separable convolutional neural network (SCNN) keyword spotting (99). Their performances are compared with that of the Intel NCS platform with regard to inference time, power consumption, and energy per inference (77). The authors observe that the inference time for the SCNN on the NCS is about 10ms, but the custom FPGA platforms achieve between 0.39 ms (Zynq-US+) to 1.45 ms (Cyclone V). However, the the NCS shows low power consumption as 0.81 W whereas the power consumption of FPGA platforms is higher than the NCS ranging from 0.969 to 4.010 W. Finally, the energy per inference of FPGA platforms is lower than the NCS because the inference time of the NCS is longer.

A.3. Memory. The Unix command mbw was utilized to test the memory performance of a wide range of ARM-based SBCs (63). mbw quantifies the available memory bandwidth by transferring large arrays of data in memory. The STREAM benchmark[12], which measures sustainable memory bandwidth, was

[8] https://nbench.io/

[9] https://github.com/akopytov/sysbench

[12] https://www.cs.virginia.edu/stream/

used to evaluate an edge resource (61). The memory performance with Copy, Scale, Add, and Triad operations in STREAM was measured on three different software systems, namely native Linux, Docker, and KVM. OC1+ outperformed RPi2 as OC1+ adopts 792 MHz LPDDR3 RAM while RPi2 uses 400 MHz LPDDR2 RAM. RPi3 utilizes 900 MHz LPDDR2 RAM and in most cases outperforms OC1+. Odroid C2 (OC2) and Odroid XU4 (OXU4) outperform RPi2, RPi3, and OC1+ due to their twice larger RAM capacity.

A.4. Storage. Small form factor edge resources usually use flash-based storage devices, such as embedded MultiMediaCard (eMMC) and MicroSD. The storage performance of edge resources (61) have been evaluated using Bonnie++[13] and the Unix DD command. Bonnie++ measures the data read and write bandwidth as well as the number of file operations per second. DD was used to measure the bandwidth for accessing special device files such as /dev/zero/. The fio benchmark[14] has been used to perform sequential read/write operations on MicroSD cards and sysbench was utilized to perform random disk operations on the eMMC (63). The evaluation result shows that eMMC cards operate at speed in the order of hundreds of 'MB/s' whereas the the MicroSD card operates in the range of hundreds of 'Mb/s'.

B. Software Platforms. Orchestrators, cloud computing services, and schedulers are examples of software platforms benchmarked on edge computing systems. Although other software platform exist, there is minimal research in the context of edge benchmarking and are therefore not presented.

B.1. Orchestrators. The orchestrators for edge computing manage edge resources by creating containers, deploying and starting servers, and assigning and scaling computation resources. Orchestration in edge computing is challenging due to the limited hardware resources, large volume of edge resources, and mobility of end device connected to them (10).

FocusStack coordinates edge resources for moving targets, such as cars and drones (100). This is a challenging task since existing orchestrators, such as OpenStack are developed to mange a relatively small number of servers. FocusStack extends OpenStack for supporting location-based awareness that minimizes managed devices at a single time. The full-time active monitoring system of FocusStack was benchmarked with the following metric, which is the total bytes every ten seconds transferred for monitoring between the orchestrator and the device. FocusStack sent 358 bytes per 10 seconds while unmodified OpenStack transferred 17,509 bytes per 10 second interval.

Edge workload orchestrators that select target edge resources for offloading tasks have been proposed using Fuzzy Logic (101). The orchestrator is benchmarked by relying on data obtained from the EdgeCloudSim simulator (102). The applications explored include augmented reality, health, compute intensive and infotainment applications, to evaluate the service time, failed tasks, and virtual machine utilization.

B.2. Service Model. Edge computing may employ computation and network resources at the edge of a network to build a cloud (103). Therefore, similar to conventional clouds, the Infrastructure-as-a-Service (IaaS) model can be obtained at the edge (104), and two such examples are discussed. Nebula implements a decentralized edge cloud by utilizing volunteer edge nodes that provide computation and storage resources (105). Nebula offers the IaaS service model (compute and storage services are available). Benchmarking with the Nebula MapReduce Wordcount and InvertedIndex applications in terms of performance, fault tolerance, and scalability have been presented (106).

FemtoClouds offers IaaS type service by forming a small-sized clusters using smartphones and laptops by leveraging idle or less loaded resources to complete a user's request (107). FemtoClouds has been benchmarked against metrics, such as computational throughput, network utilization, and computational resource utilization.

B.3. Schedulers. Edge schedulers allow service providers to allocate computing resources in an efficient manner. For example, mobile edge computing offloads computations to the edge in order to enhance the capability of mobile devices. The scheduler in this system allocates computational resources for offloaded tasks considering both the completion time and energy (108). In addition, if a data analytics application requires latency-constraints, real-time scheduling needs to be provided.

Resource scheduling policies for satisfying the real-time requirements of smart manufacturing applications in edge computing are benchmarked in (109). A two-phase scheduling strategy is adopted that firstly selects a suitable edge computing server based on the task load, after which additional servers are selected if necessary to distribute the task when one server cannot meet the real-time constraint. The scheduler is evaluated with OpenCV applications using metrics, such as computing latency, satisfaction degree, and energy consumption. The fairness aspect of edge scheduling has been evaluated recently using synthetic benchmarks (110).

5. Techniques Analyzed

This section reviews various techniques that have been analyzed through edge benchmarking. Table 2 summarizes the characteristics of selected techniques, which are considered further from the perspective of resource allocation, offloading mechanisms, and deployment options in the following subsections.

A. Resource Allocation. Resource allocation is benchmarked to explore characteristics relevant to the edge paradigm, for example an edge computing system may comprise multiple hierarchical layers of compute resources from the cloud to the extreme edge of the network. The allocation mechanisms (shown in Figure 4) are benchmarked against energy consumption, costs, or quality-of-service (QoS).

A.1. Energy-efficient allocation. Reducing the energy consumed by mobile applications has been benchmarked extensively on edge resources given that the joint allocation of communication and computation resources in a Mobile Edge Computing (MEC) system is essential.

Early efforts formulated a *convex optimization* problem for minimizing mobile energy consumption (111). A multi-user MEC system is considered with a mobile base station and multiple mobile devices, from which tasks are split using a threshold-based policy.

[13] https://www.coker.com.au/bonnie++/

[14] https://github.com/axboe/fio

Techniques Analyzed by edge performance benchmarking	Optimization criteria			Deployment options				
	Energy consumption	Monetary cost	Quality-of-Service	End-user device	Edge	Cloud	Simulated platform	Virtualization
Resource allocation (111–113) RTLBB (114) EEDOA (115) Offloading (116)	Y	N	N	N	N	N	Y	N
Service placement (117–120) MeFoRE (121) FogTorch (122) Scheduling (123) Offloading (124)	N	N	Y	N	N	N	Y	N
ITEM (125) Service placement (126–128)	N	Y	Y	N	N	N	Y	N
DYVERSE (8) ENORM (7)	N	N	Y	Y	Y	Y	N	C
Service migration (129)	N	Y	Y	Y	Y	N	N	N
Resource allocation (130, 131)	N	Y	N	N	N	N	Y	N
Resource allocation (132, 133) JCORAO (134) Application placement (135)	Y	Y	N	N	N	N	Y	N
HMAOA (136) Offloading (137–139) YEAST (140) EDAL (141)	Y	N	Y	N	N	N	Y	N
Data replication (142)	N	N	Y	N	Y	N	N	N
Data replication (143)	N	N	Y	Y	Y	N	N	N
Wearable cognitive assistance (144)	N	N	N	Y	Y	Y	N	V
FocusStack (100) ParaDrop (145)	N	N	N	Y	Y	Y	N	C
Migration (146)	N	N	Y	N	Y	N	N	C
Offloading (147)	Y	N	Y	Y	Y	Y	N	N
Virtualization (63)	Y	N	N	N	Y	N	N	C

Table 2. Comparing the characteristics of techniques analyzed in edge performance benchmarks; Virtualization: Virtual machine, Container, or None

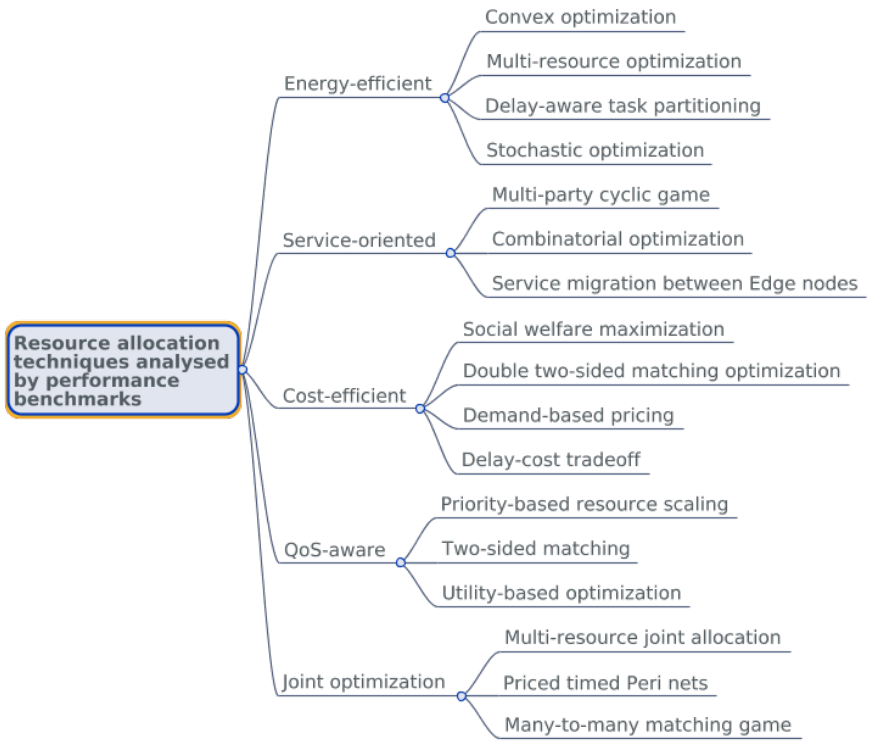

Fig. 4. A classification of resource allocation techniques analyzed by edge performance benchmarks

A fine-grained *multi-resource joint optimization* of the energy consumed for task offloading, sub-channel allocation and CPU-cycle frequency is developed (114). Energy efficiency is benchmarked when workloads have varying executing times on MEC servers (112). ThriftyEdge (148) is used to benchmark the performance of *delay-aware task graph partitioning* and a virtual machine selection method to minimize an Internet-of-Things (IoT) device's edge resource occupancy. *Stochastic optimization* to minimize the energy consumption of task offloading while guaranteeing the average queue length of IoT applications is benchmarked in (115).

A.2. Service-oriented allocation. Benchmarking service-oriented resource allocation accounts for the fact that application service providers may be independent from edge operators with different operating objectives. The following techniques have been analyzed by benchmarks:

A *multi-party cyclic game* involving three parties, namely application users, edge nodes, and application service providers, is developed to quantify the benefits for different participants (149).

Taking the heterogeneity of applications and their Quality-of-Services (QoS) into account, models for edge service placement and for reducing service delay and increasing edge resource utilization are benchmarked (117). ITEM (125) explores the performance of a *combinatorial optimization* problem of deploying services of social VR applications. It benchmarks both the economic operations of the edge systems and the QoS of applications for the end users.

DYVERSE (8) periodically adjusts the allocation of edge resources for an edge application using vertical scaling. It assigns dynamic priorities to service-based applications that utilizing an edge computing node, and then performs scale-up or scale-down actions for each applications in order to reduce the overall violation rates of the service-level objectives.

A dynamic service migration mechanism to improve the serviceability of applications is benchmarked for edge-based cognitive computing (129). Therefore, application *service migration between edge nodes* based on the behavioral cognition of mobile users is considered.

A.3. Cost-efficient Resource Allocation. Monetary costs are also a consideration in edge benchmarking. An auction-based edge computing resource market is proposed to *maximize social welfare* of both users and edge Service Providers (ESP) (130). The auction-based algorithm considers the ESP's cost of running computing services on edge systems.

Resource allocation problem modeled as *double two-sided matching optimization* is explored for a three-tier device-edge-cloud computing environment (131). The double matching strategy is benchmarked, which ensures high cost efficiency while achieving a stable status when none of the three participants, namely the users, the edge and the cloud, can change their paired partners for achieving more cost-efficiency.

Equilibrium solutions for resource allocation using *demand-based pricing* is benchmarked for exploring competitive pricing strategies of heterogeneous capacity-limited edge resources (150). The proposed market equilibrium solution maximizes both the utilization of edge resources and allocates optimal resources to services under their budget constraints.

It is demonstrated that the system delay and cost tradeoff on a resource constrained edge node can be balanced by scheduling mobile workloads and outsourcing cloud resources (151). The performance evaluation shows that the system delay can be minimized by considering both computation delay and transmission overhead while maintaining a

constant outsourcing cost.

A.4. QoS-aware allocation. Improving the QoS of applications is a key advantage of edge computing (4). ENORM (7) benchmarks the benefit of offloading application services from the cloud to the edge based on the QoS of multiple applications on the same edge node. A similar approach of periodically distributing tasks in edge environments is proposed to maximize the number of tasks accommodated in an edge network while satisfying the QoS requirements such as task completion deadline and security (118).

Priority-based resource scaling is benchmarked in DYVERSE that estimates the amount of resources to be added to or removed from a running edge service (8). The metric used in the evaluation is QoS violation rate, which is also employed in evaluating the performance of an optimization model to place IoT services on edge resources to prevent QoS violations (126).

A *two-sided matching* approach is used for optimizing costs (131) and QoS (127). A QoS-aware matching algorithm that provides incentives to IoT users and cloudlets to participate in the two-sided matching game is evaluated.

From the perspective of edge service users, MeFoRE (121) uses previous records of the Quality-of-Experiences (QoE, such as service give-up ratio) for estimating the resources required by different users. QoE is frequently used in utility functions for optimizing the performance of running application services (*utility-based optimization*). The use QoE to measure the user utility of running jobs in the edge against locally running on mobile devices is benchmarked (152).

A.5. Joint Optimization. The above research focused on optimizing a single factor that can improve the performance of edge resource allocation. However, there is research in which the optimization of allocating edge resources by considering multiple factors has been evaluated.

Multi-resource joint allocation has been evaluated using a distributed proximal algorithm for jointly allocating edge resources and minimizing carbon footprint (132). The location diversity of the requested video streaming utility and costs is modeled to maximize the social welfare of both the computing service provider and the content provider.

JCORAO (134) is introduced as a distributed joint computation offloading and resource allocation scheme to provide the optimal solution in heterogeneous networks with MEC. Computation offloading strategy policies are generated, uplink sub-channels and transmission power are allocated, and compute resources are scheduled. JCORAO aims to minimize the cost of mobile terminals while satisfying offloading latency constraints.

A resource allocation strategy for edge computing based on *priced timed Petri* nets is benchmarked to solve the problem of improving the efficiency of edge resource utilization, satisfying the users' QoS requirements, and maximizing the profit of both resource providers and users (153). The price cost, the time cost, and the credibility of participants in the edge computing services are considered.

A *many-to-many matching game* is developed to evaluate pairing between data service operators and edge nodes in a three-tier edge network (133). The proposed joint optimization framework for data service operators and subscribers, and edge nodes is benchmarked to demonstrate that it is effective for achieving optimal resource allocation. Similarly, a matching game is developed in mobile edge networks (154). Both the radio and computational resources are jointly allocated to optimize performance and improve user satisfaction for IoT applications. The proposed approach evaluates the service delay, link quality, and mandatory benefit of IoT services.

B. Computation Offloading. Bringing services closer to the data source is the key premise of edge computing, referred to as computation offloading. This section as shown in Figure 5 reviews the computation offloading techniques that are analyzed in edge performance benchmarking. There are multiple ways in which computation can be offloaded (155). Offloading techniques can be broadly classified on the basis of: (i) the context of the edge environments, namely static and dynamic edge contexts, and (ii) the direction of the offload, namely from the cloud to edge, from the edge to cloud, from the edge to edge, and from the device to edge.

B.1. Context. Both static and dynamic strategies are presented in the literature. Static offloading strategies offload the same services or components of an application at all times and do not adapt to varying operational conditions of the edge environment. Dynamic offloading strategies, however, adapt to system and network (internal or external) and application specific conditions for maximizing service level objectives by redeploying a different set of services or components to the edge than the initial deployment.

i. Static offloading: The problem of identifying tasks for offloading has been explored as a *multi-objective optimization* problem across the dimensions of energy and execution time (with the aim of minimizing them) (116). To this end, tasks are classified on the basis of execution time and energy cost of the computing process after which they are prioritized for offloading.

FogTorch (122) is an offloading framework that uses a *greedy heuristic* approach for generating offloading plans. Multi-component IoT applications in a multi-layer edge computing system is in view by exhaustively determining all possible deployment plans by considering different combination of application components.

ii. Dynamic offloading: A module mapping algorithm is used to place different modules of an IoT application in an edge-cloud infrastructure by considering module characteristics (135). This results in better edge resource utilization. Similarly, the placement problem of multi-component applications in the edge by designing an efficient heuristic on-line algorithm is considered (119). In addition, the dynamic nature of users' location and the network capabilities when distributing computation is accounted for.

Dynamic offloading is considered under the constraints of user mobility and multi-tenancy.

User mobility is one reason for the transient nature observed in edge environments. There is research that benchmarks the performance of *mobility-aware scheduling* (123). The scheduling policies considered, include concurrent, first-come-first-serve, and delay-priority for improving the overall execution time. A *heuristic offloading* approach is benchmarked to evaluate the performance of an approximate offloading scheme in a vehicular scenario (136). Similarly, a mobility-aware dynamic service placement framework is considered for enabling cross-edge service migration to support user mobility (128).

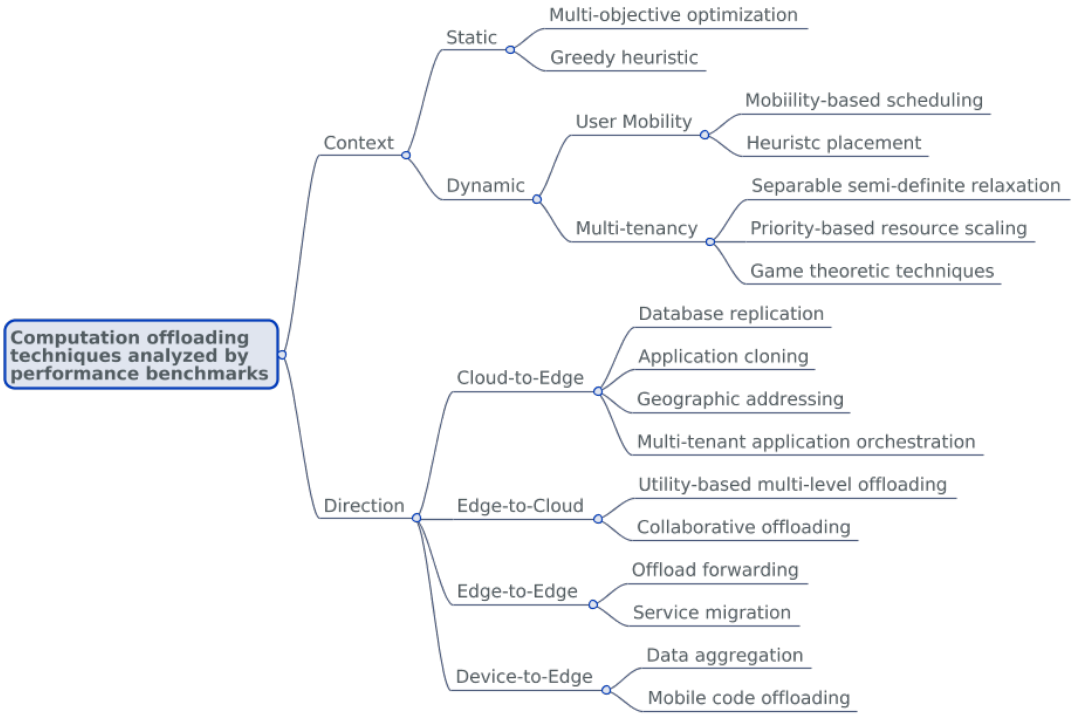

Fig. 5. A classification of computation offloading techniques analyzed by edge performance benchmarks

Multi-tenancy, the co-location of multiple workloads on the same edge resource can reduce the performance of an application, for which efficient management of resources is required. Multi-user MEC is explored to jointly optimize the offloading decision and communication resource allocation by taking a *separable semi-definite relaxation* approach (137). The research is further extended to consider the resource allocation and processing cost for computation (140). DYVERSE (8) benchmarks a *priority-based resource scaling* mechanism after an application is deployed. By accounting for interactions between multiple users, game-theoretic techniques are designed to achieve a Nash equilibrium for a multi-user MEC system to minimize energy consumption and offloading latency (138). This research is further extended to the scenario when multiple devices offload computation to multiple edge nodes (139).

B.2. Direction. The following four directions of offloading are considered in edge environments:

i. Cloud-to-Edge: Computation from distant clouds is brought to the edge of the network in an effort to reduce data transmission time to a server and thereby reduce the application response time. Typical approaches that are benchmarked to evaluate the benefit of cloud-to-edge offload include: (i) *Database replication* on the edge (142). A similar approach is adopted for copying and transferring application specific data from the cloud to the edge (143). (ii) *Application cloning* on cloudlets is another approach that is benchmarked (144). (iii) *Geographic addressing* is another approach evaluated in the discovery of edge nodes for deploying workloads from the cloud in FocusStack (100). (iv) *Multi-tenant application orchestration* across multiple edge nodes is evaluated on WiFi access points in ParaDrop (145).

ii. Edge-to-Cloud: Edge nodes are expected to take different form factors, including mini data centers, cloudlets, mobile base stations, embedded systems. This is typical in the area of the Internet of connected vehicles in which a *utility-based multi-level offloading* scheme is evaluated to maximize the utility of vehicles, edge servers and cloud servers (124). A *collaborative offloading* approach is evaluated to jointly optimize resource allocation and the offloading decision (156).

iii. Edge-to-Edge: When an edge node does not have sufficient resources, it may offload (or migrate) the workload to a peer. Two approaches are typically benchmarked: (i) *Offload forwarding* which forwards all unprocessed workloads to neighboring edge nodes for meeting service objectives (157). (ii) *Service migration* that dynamically migrates services across multiple heterogeneous edge nodes (129). Service hand-off approaches are evaluated to investigate their feasibility for supporting seamless migration to the nearest edge server while a mobile client is moving (146).

iv. Device-to-Edge: Typically 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 pre-processing and filtering tasks (140, 141). For energy-constraint devices such as smart-phones, workloads are transferred from a user device to an edge node, referred to as *mobile code offloading* (113). Such an offloading model has been benchmarked to demonstrate its effectiveness for reducing the energy consumption of the device and application latency (147).

C. Deployment Options. Three virtualization technologies have been used in edge deployment research, namely Virtual Machines (VM), containers, and unikernels, as shown in Figure 6.

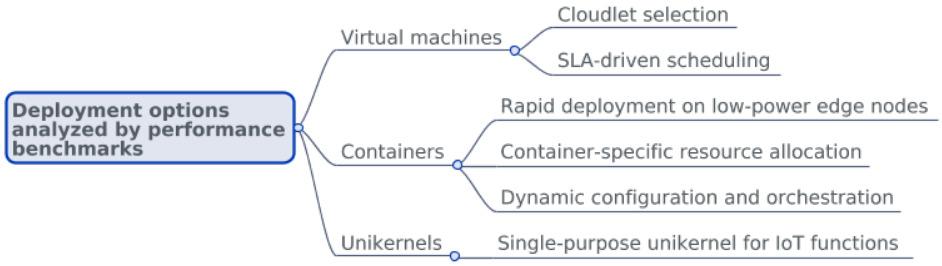

Fig. 6. A classification of deployment options analyzed by edge performance benchmarks

C.1. Virtual Machines. These were an essential technology in the development of the cloud and became important for edge computing through the development of *cloudlets* (158). Subsequently, the literature focused on evaluation of VMs on the edge. For example, selecting the most suitable VM was evaluated for different types of applications (147) and a general approach for *SLA-driven scheduling* for placing VMs in a multi-network operator-sharing edge environment was benchmarked (120).

C.2. Containers. Containerization is extensively employed and evaluated in edge systems given the reduced boot up times and lower resource footprint compared to VMs (159). Early benchmarking efforts highlight the feasibility of using Docker containers[15] as a viable option for providing rapid edge deployment (160). The performance overheads of *rapid deployment on low-power edge nodes*, such as single-board computers are benchmarked for a wide set of edge nodes used for IoT applications (63). An alternative container technology, namely Linux containers[16] is used by the ENORM framework (7).

Numerous approaches for efficiently managing edge containers have been evaluated. For example, a task-scheduling algorithm to ensure minimum task completion time and an optimal number of concurrent tasks on an edge node is considered (161). A *container-specific resource allocation* approach is benchmarked in DYVERSE by taking the QoS of the containerized applications into account (8). Cloud4IoT (162) is a platform that facilitates roaming and the offloading of IoT functions. An evaluation of *dynamic configuration and orchestration* of IoT functions in a three-tier edge computing architecture is presented.

C.3. Unikernels. Unikernels[17] are relatively less popular and used for *single-purpose applications* that use library operating systems and are sealed against modification after deployment (163). The small resource footprint is an attractive aspect for edge computing. Unikernels and containers have been benchmarked against the dimensions of the scalability, security and manageability for IoT applications on the edge (164).

6. Quality

As already discussed, benchmarking is the process of stressing a system while observing its response to a range of quality metrics. In this, benchmarking captures non-functional properties or "how well" a given system can fulfill its functional properties. In the context of non-functional properties, there are three concepts which need to be considered: quality indicators, quality metrics, and quality tradeoffs. In this section, we present an overview of these three concepts along with examples of quality indicators for edge benchmarking.

A. Quality Indicators. There is a broad range of different quality indicators that can be further divided into sub-qualities. For instance, performance as a quality indicator can be broken down into its two qualities: throughput – amount of load a system can handle, and latency – time to process a single or batch request (12).

Most quality indicators describe a deviation from an ideal state which is usually unreachable. For example, latency can never be zero and availability can (on a long time horizon) never be "always". Throughput as a counter example, may be zero but its ideal state is "infinity", which is also an unreachable (ideal) state. It is also important to distinguish between a measurement of the ideal state and the system operating at that ideal state. When the measurement resolution is low, it may be possible to measure, for example, a latency of zero, but the actual system will have a latency that is negligible, but greater than a zero value. Overall, this means that quality indicators are usually bounded in at least one direction – they either have an upper or lower limit (12).

B. Quality Metrics. To measure quality, a quality metric which is a function that maps a concrete quality level to a specific value and a unit is required. "A quality metric is, therefore, the function that expresses differences in quality" (12); for each quality metric that has to be observed, there may be more than one such function.

There are atleast nine requirements that quality metrics need to fulfill (12, 165, 166): (1) *Correlation*: There needs to be a (preferably linear) correlation between the quality indicator under observation and the metric output, i.e., "changes in quality behavior should always be visible in metric output" (12). (2) *Tracking*: When a quality level changes, the metric output should change and do so (preferably) instantly. (3) *Monotonicity*: When a quality level changes in a particular direction then the metric output should always change in the same direction, no matter what the quality level was at the start or at the end. In other words, increasing a quality may either increase or decrease the metric output but must do so consistently for all quality levels. (4) *Discriminative Power*: "The metric should be able to clearly differentiate between low and high quality levels" (12). (5) *Reproducibility*: Aside from measurement errors, a repeated measurement should always yield the same metric output for an unchanged quality level as long as the experiment setup is not changed (167). (6) *Resolu-*

[15] https://www.docker.com/
[16] https://linuxcontainers.org/
[17] http://unikernel.org/

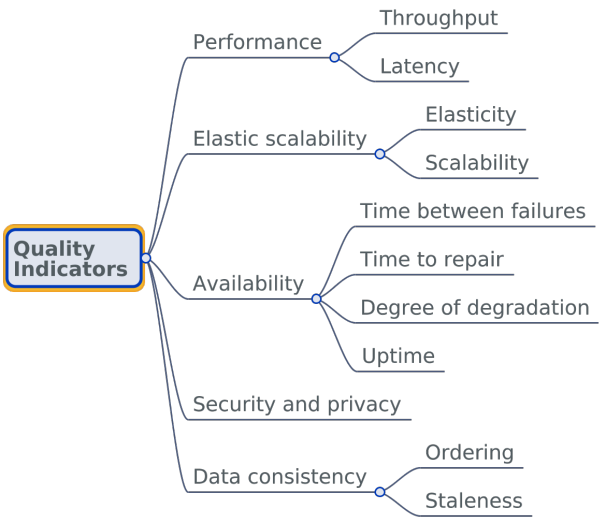

Fig. 7. An overview of a subset of quality indicators relevant to edge benchmarking

tion: A metric needs to have a large set of metric output values and should preferably be continuous to differentiate between arbitrary small differences in quality level. This also implies that metrics should usually not be aggregates but should rather "report raw, single values that can be aggregated in a second phase based on standard aggregation methods" (12). (7) *Granularity*: A metric should only measure a single quality indicator at a time. Otherwise, changes in one quality may offset changes in another. (8) *Meaningfulness*: "A metric may have a (specific) target audience but there must be at least one target audience for which the (metric outputs) are meaningful and understandable...(so) that they can actually use them for their purposes" (12). (9) *Reliability*: The above requirements should be fulfilled "for at least P% of the application of the metric" (165). Concrete values for P depend on the specific use case but should be as high as possible, preferably 100%.

To develop a measurement method in edge benchmarking, quality metrics that fulfill these requirements as much as possible (sometimes, there are constraints which make it hard to fulfill them completely) are required. In most cases, such a metric already hints how it can be measured. For instance, response time (as a performance metric for latency) is usually measured by noting the timestamp before sending a request and upon receipt of the response. Then the duration between both is calculated and reported as the metric value.

C. Examples of Quality Indicators. There is a wide range of different quality indicators - those that are omnipresent, such as performance or availability, those that are relevant to a class of systems, for example, data consistency for systems replicating state, and those than can only be observed indirectly, such as elastic scalability. In this section, an overview of a selected set of quality indicators (shown in Figure 7) and their relevance to edge benchmarking is discussed. The indicators considered are not comprehensive, but only a subset is chosen for discussion in this paper.

C.1. Performance. Performance is an omnipresent, directly observable quality indicator and is broken down into the two quality dimensions throughput and latency. *Throughput* generally describes the amount of load a system can simultaneously handle, while *latency* describes the time necessary to handle unit load. For a system that is accessed in a request-response manner, throughput is usually measured as requests per second (potentially as maximum throughput while not violating SLAs, such as latency) where as latency is measured as the time between sending a request and receiving a response.

In the context of edge benchmarking, performance is an important quality indicator for a number of reasons. First, resources at the edge are usually hardware limited. The capability to handle as many requests in parallel (throughput) and to service a single request as fast as possible (latency) will directly affect how rapidly requests will need to be offloaded from the edge to the cloud (168). This has an impact on latency for requests originating from latency-critical applications. Second, most edge computing use cases have latency concerns that are alleviated by placing a few components or services of a system at the edge to reach QoS goals. An example is interconnected or autonomous driving where a delayed response leads to a degradation of the service.

Most TPC[18] and SPEC[19] benchmarks are performance benchmarks. Benchmark approaches quantify performance of relational database systems (48), NoSQL stores (35, 37, 54), VMs (41, 169–171), block storage (172), or serverless Function-as-a-Service (FaaS) platforms (173–175). They can also be used in continuous integration processes to avoid performance degradation of software systems (176, 177).

C.2. Elastic Scalability. Elastic scalability is omnipresent in that it is relevant to all systems but is generally not directly observable. However, a less elasticly scalable system is observable through performance degradation. Elastic scalability can be broken down into the qualities scalability and elasticity. *Scalability* describes how well a system adapts to changes based on available resources, which is how much load can be handled, for example, when doubling the available resources (ideally, at least twice the load). When a system is scaled out by adding more machines or using a bigger machine (and correspondingly scaled up), the new "load" state is not reached instantaneously. Instead, there is an intermediate period during which neither the previous nor the new load level is reached. While this may be a gradual change for some systems, others may experience a significant performance drop (for example, when large data chunks need to be redistributed across machines). *Elasticity* describes what happens in between the previous and the new stable state, how long it takes to reach the stable state and how severely performance is impacted during scaling.

In the context of edge benchmarking, elastic scalability has applicability in multiple scenarios. First, during on-demand federation with local peer nodes to increase capacity (scaling out). Second, during offloading operations when an application component is moved across multiple edge, intermediary, or cloud resources, the application will need to adapt to the underlying infrastructure (scaling up or down). The assessment whether a system under test follows a design that supports this, is a key goal when benchmarking applications that can move between edge and cloud (11).

There are a number of benchmarking approaches that quantify elastic scalability. For example, metrics for measuring scalability of cloud database systems have been defined (36, 178), scalability experiments have been carried out (55, 179), generic

[18] www.tpc.org
[19] www.spec.org

cloud elasticity metrics have been defined (180), and elasticity of serverless FaaS platforms has been benchmarked (181).

C.3. Availability. Availability is an omnipresent quality that is relevant to all types of systems and can be directly observed. It is directly related to fault-tolerance (availability in the presence of failures) and reliability. Availability is understood to define whether a system responds to requests. However, it can be challenging to determine which of the following conditions meet the criteria for availability. Is a system available (i) when the host can be reached, (ii) when the host returns any response, or (iii) when the host returns a correct response? (182). In addition to considering individual requests, availability can be aggregated to *uptime* of an entire service. This is based on *mean time between failures* (how often a failure occurs), and *mean time to repair* (how long a service remains unavailable upon a failure) (183, 184). Recently, *windowed user-uptime* has been proposed as a more meaningful (and aggregated) availability metric (185).

In the context of edge benchmarking, availability is relevant for large scale edge resource deployments to understand aggregated failure statistics across a large number of small distributed machines that are connected through unreliable networks. Fault-tolerance and the resulting degree of degradation is often more important than uptime for edge-based applications raising the questions – how well does the system deal with failures and how many failures can the fault-tolerance mechanisms cope with?

C.4. Security and Privacy. Security and privacy are fundamental and required by design, unless a system is running in an entirely isolated environment. Privacy refers to the individual's rights to control of their own personal data is multi-faceted, across concepts of consent management (186) or data minimization (187). It is usually governed by legislation such as the General Data Protection Regulation (GDPR). Security, on the other hand, is about avoiding any misuse of a system and associated data. Security cannot be completely benchmarked as this would require knowledge of *all* possible attacks on a system, but it can be approximated using all *known* attacks. Security benchmarks consider preferences of providers (188) or performance overheads of security mechanisms (for example, of database systems (49, 189, 190), homomorphic encryption (191), web servers (192–194), and SOAP messaging (195)). The only benchmarking approach on privacy that the authors are aware of evaluates the effects of GDPR compliance on database workloads (196).

In the context of edge benchmarking, novel security mechanisms using entirely new attack vectors need to be explored, since physical access to an edge device are a possibility, which is nearly impossible with cloud resources embedded within data center distant from consumers (11). Nonetheless, edge resources can improve privacy, e.g., by separating data lakes (187).

C.5. Data Consistency. Data consistency is not necessarily an omnipresent quality as it only matters to systems managing data. Data consistency has two dimensions, ordering and staleness, which can be considered from a client and a provider perspective (12). Usually, the client perspective is taken, which is more relevant for benchmarking as it describes the effects directly visible to a benchmarking client. *Staleness* occurs when an update becomes (partially) visible before executing on all replicas. In this sense, applications then read outdated data. Staleness, however, can in many cases be tolerated by applications (197). *Ordering* from a provider perspective describes the degree to which requests can be executed in a different order on different replicas. From a client perspective, such ordering issues become visible as violations of monotonic reads, monotonic writes, read your writes, and write follows reads (198). Violations of these guarantees from an underlying datastore can either become visible to end users or make the application development quite challenging. For instance, violations of monotonic writes imply that writing and then updating some value is not guaranteed to result in the updated value on all replicas. There are multiple benchmarking approaches that consider data consistency (43, 44, 199–202).

In the context of edge benchmarking, data consistency as a quality is only relevant when considering multi-node deployments, for example, across several edge nodes or when integrating the edge and cloud. In these scenarios, consistency guarantees are affected by inter-node latency of replicas (tends to be large) when replicas are deployed on edge nodes. Furthermore, design choices of the replica synchronization protocols in LAN and WAN deployments for edge systems need to be analyzed through experimental evaluation. Additional data related challenges that should be considered by edge benchmarks have been reported in (203).

D. Quality Tradeoffs. The quality indicators considered above are rarely isolated; instead, they usually have a direct or indirect tradeoff relationship (12, 166). A direct tradeoff relationship between two qualities A and B implies that increasing A is likely to decrease B. Indirect relationships are transitive. For example, direct tradeoffs between qualities A and B as well as B and C mean that an increase of A reduces B, which may be offset by a decrease in C instead. There are a wide range of direct tradeoffs, such as the CAP theorem[20] and the PACELC model (204), which describe tradeoffs between *data consistency and latency* as well as data consistency and availability (in the presence of failures). Other known direct tradeoffs are between *performance and cost* (adding more infrastructure resources increases cost) or *security and performance* (additional efforts for encryption or authentication schemes introduce an overhead and hence a performance degradation).

Benchmarking should carefully analyze the quality of interest for tradeoff relationships and measure both sides of the tradeoff (12). The same holds for edge benchmarking. For example, consider performance benchmarks for a distributed storage system deployed across multiple geo-distributed edge nodes. System A replicates updates synchronously across all replicas, system B replicates updates asynchronously. Any performance benchmark is likely to identify system B as the superior system. This changes when a data consistency benchmark is run in addition and both quality indicators are weighed against each other.

7. Benchmark Runtime

This section will discuss the execution environment and the deployment destination and test-beds relevant to edge benchmarking. The execution environment will highlight the software and data related characteristics considered by different edge benchmarks during runtime. Then, single and multiple

[20] http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf

Execution environment characteristics of edge benchmarks		CAVBench (67)	EdgeDroid (72)	EdgeBench (59)	IoTBench (74)	CoAP benchmark (64)	MAVBench (73)	SoftwarePilot (75)	DeFog (9)	Edge AIBench (68)	LEAF (70)	AIoT Bench (58)	HERMIT (69)	pCAMP (71)	RIoTBench (57)
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

Table 3. Comparing the software and data related execution environment characteristics of edge benchmarks

destinations for deployment are considered. Finally, real-world, lab-based, emulated, and simulated test-bed infrastructure are presented.

A. Execution Environment. Execution environments comprise software packages and data sets needed to execute a benchmark (205). With growing support from programming languages (for example, Python) and virtualization tools (for example, Docker), execution environments can be reused across projects, research groups, and scientific disciplines. Open, representative and comprehensive execution environments broaden research avenues by enabling scientific exploration in new domains. However, execution environments differ greatly across edge computing applications, changing rapidly and frequently. Often benchmarks are designed narrowly to target specific workloads, sacrificing features needed to reuse their execution environments.

A.1. Characteristics. Table 3 lists seven characteristics of open, representative, and comprehensive execution environments. First, software packages and data sets must be accessible to researchers outside of the initial study. Clearly, benchmarks are accessible if they use only open-source software (Characteristic #1). However, benchmarks that represent complex and emerging workloads will require custom and/or proprietary software. When needed, such software should be clearly identified and made available (Characteristic #2). Likewise, data sets used to drive benchmark execution should be open and easily ported across research contexts (Characteristic #5).

Benchmarks represent real workloads by mimicking aspects of their functionality. However, software components with limited functionality place less stress on edge devices than commercial-grade software. Execution environments composed, in part, from commercial-grade software ensure representative demand on edge devices (Characteristic #3).

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

A.2. Analysis. Table 3 reveals interesting trends. Edge benchmarks often (1) employ commercial-grade software components (71%) and (2) use open data sets (84%). These results are likely influenced by machine learning kernels in edge workloads. Commercial-grade versions of neural network platforms, object detection models, and speech processors are open source and widely used. In contrast, only 50% of the studied benchmarks support comprehensive evaluation across runtime and contextual settings. This stems from dependence on traces from prior executions. 75% of the benchmarks were narrowly tailored to specific contexts, using closed-model traces that are not easily adapted across runtime and contextual settings.

Among the studied benchmarks, EdgeDroid (72, 206), SoftwarePilot (75), MAVBench (73) have the characteristics of open, representative and comprehensive execution environments. SoftwarePilot and MAVBench capture, for example, virtual reality. Each of these workloads employ commercial-grade open-source software components for machine learning and cognitive assistance. Also, in the corresponding papers, each benchmark is evaluated across runtime and contextual settings, for example, adjusting the complexity of machine learning models or adapting user quality-of-service expectations. Leaf (77) is also a comprehensive benchmark that can vary workloads and runtime settings. A subtle difference among these benchmarks is that MavBench relies on simulated environments, a design choice that could yield non-representative workloads if simulations deviate from actual edge conditions. In contrast, SoftwarePilot, EdgeDroid, and Leaf use Markov decision models to adapt real data to new contextual settings. This approach is likely more robust for researchers targeting new domains.

B. Deployments. This section will explore the different resource locations where the benchmark is executed, referred to as deployment destination, as shown in Figure 8 and test-bed options. The following deployment destinations are considered: (i) Single destination, and (ii) Multiple destinations.

B.1. Destination. As mentioned previously, this article refers to the edge as resources located at the edge of the wired network. However, a number of researchers also consider user-devices to be the edge by adhering to a broader definition of the edge (of the network). Therefore, we additionally consider a device-only deployment under single destination.

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

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

Low-power wireless industrial sensors: Wireless protocols of industrial sensor devices have been benchmarked using an observing module (66). 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. The 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: There is research that benchmarks the system architecture for Constrained Application Protocol (CoAP) based IoT devices (64). The key results from benchmarking are that: (i) the processor chosen directly impacts the CoAP server performance, and (ii) the latency of the communication channels affected the round-trip times of CoAP requests.

Low overhead virtualization: The performance of virtualization for user-devices has been benchmarked (61, 63). A range of virtualization techniques, such as Docker containers and Kernel-based Virtual Machine (KVM) have been considered (61). The key result is that hypervisor-based virtualization has larger overheads and containers seem to be more appropriate for the network edge. Container-based virtualization across five different devices is considered (63). It is noted that there is negligible impact on performance when using containers compared to bare-metal execution. The characteristics of the workload are key to estimating energy efficiency of the devices.

Medical IoT devices: The compute and memory characteristics of IoT devices used in the medical domain is benchmarked (69). A collection of medical applications (macro benchmarks) are evaluated against the MiBench, PARSEC and CPU06 micro benchmarks. The evaluation shows the execution characteristics differ across micro and macro benchmarks.

Devices that run machine-learning workloads: Benchmarking machine learning-specific workloads for devices has been presented in (58). 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 are considered. Similarly, benchmarking for the vision and speech domain has also been considered (74).

b. Edge-only: Research that explores benchmarking for edge-only deployment will ideally require macro-benchmarks that are edge native (the edge resource is not merely an accelerator, but is essential for the application to be used in the real-world). Autonomous vehicles are one such example. CAVBench is a benchmarking suite developed for autonomous vehicles by focusing on real-time applications that need to process unstructured data (67). The edge node employed in this research is 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 (77). It is noted that this use-case may not necessarily be an edge-native benchmark.

ii. Multiple destination refers to the execution of the benchmark either across the entire device-edge-cloud resource pipeline or a partial resource pipeline (for example, device-edge or edge-cloud).

a. Entire Device-Edge-Cloud pipeline: There are a number of examples of benchmarking research that leverages the entire resource pipeline, comprising the device, the edge of the wired network, and the cloud for benchmarking. Three types of applications are considered:

Service pushdown: Voice interactive applications have been used as macro benchmarks for examining the entire resource pipeline (65). The aim is to push services from the cloud across weak devices and the edge to optimize applications for obtaining dialogue consistent latency.

Data aggregation: Benchmarks relevant to data-intensive workflows of power grids have been presented in the context of the entire resource pipeline (TPCx-IoT)[21]. The workflow enables real-time analytics to be performed on gateways while ingesting data from 200 different types of power station sensors. The benchmark operates in two runs, with each run comprising warm-up and measured runs. Performance, price and availability metrics are gathered. The majority of benchmarks considered by DeFog fall under this category (9).

Edge inference: Benchmarks are employed to train machine learning models on the cloud and carry out inference on the edge (assuming that the trained model is available on the edge) (68). A variety of devices or sensors are envisioned that generate data in patient monitoring, surveillance, or smart home scenarios. Similarly, inferring on the device, edge, and cloud on different machine learning platforms, namely TensorFlow, Caffe2, PyTorch, MXNet, and TensorFlowLite is considered by pCAMP (71) (also considered on different accelerators (76)). It is to be noted that in this case inference is not distributed, rather inference is entirely carried out on the edge.

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

Device-Edge: Edgedroid (72) is an exemplar of benchmarking human-in-the-loop applications, such as cognitive assistance on an edge-cloud deployment, in this case the edge being a cloudlet. The underlying benchmarking approach is to mimic applications by replaying traces of sensory input that are obtained from running the application in the real-world. The feedback generated by processing the sensory input on the cloudlet is processed using a model of human reactions. This enhances the understanding of latency trade-offs in relation

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

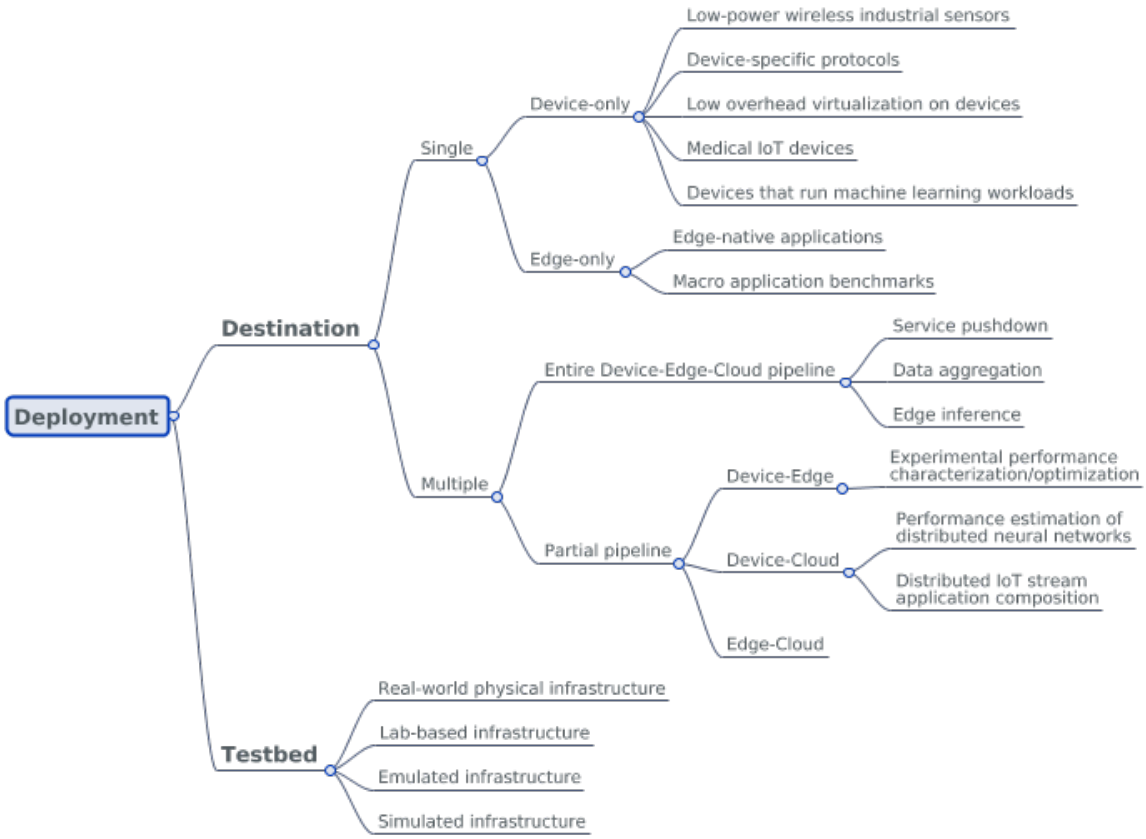

Fig. 8. A classification of edge benchmarking runtime across the dimensions of execution environment.

to both the application and the edge-cloud deployment.

Device-Cloud: The device-cloud pipeline is demonstrated for estimating performance of distributed intelligence (207) and IoT stream application composition (57). The former deploys a sliced neural network across a user device and the cloud for distributed inference by estimating where a neural network should be sliced. The latter enables benchmarking by composing distributed stream applications using modular IoT tasks.

Edge-Cloud: This deployment pipeline is usually 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 to other partial pipelines and the entire resource pipeline; refer to Section B.2. Benchmarks that exploit this pipeline include EdgeBench (59) and SoftwarePilot (75).

B.2. Testbeds. The test-bed options for deploying edge benchmarks include real-world and physical, lab-based experimental, emulated, or simulated infrastructure as shown in Figure 8.

i. Real-world physical infrastructure: These refer to "in the wild" physical test-beds that are close to in operational characteristics to an actual edge deployment. There are only a few such test-beds available; for example, the Living Edge Lab[22] and those reported in the literature (208, 209). There is no evaluation of any of the benchmarks listed in Table 1 on real-world infrastructure.

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

ii. Lab-based experimental infrastructure: The majority of testbeds in which edge applications or benchmarks listed in Table 1 have been evaluated are lab-based infrastructure. These may comprise using public cloud offering or private cloud resources coupled with edge resources in the form of single board computers (7, 9) (or a cluster (210)) or routers/gateways. End-user devices may range from wireless sensors (66) to user gadgets (7, 65). It is noted that a number of benchmarks do not rely on real-data, but instead use simulation data given the complexity of the environment they benchmark; for example, autonomous cars (67) or drones (75).

iii. Emulated infrastructure: While real-world physical infrastructure is not easily accessible to all and many lab-based experimental infrastructure are small-scale and do not represent the characteristics of a real infrastructure. Therefore, it has recently been proposed to use emulated infrastructure (211). An emulated environment will rely on deploying edge servers on the cloud and configure them and the network and interconnects, such that they represent real edge environments. However, this assumes knowledge of the parameters required to configure a realistic emulated edge environment that is acquired from a real-world edge infrastructure.

iv. Simulation infrastructure: A number of simulators, such as EdgeCloudSim (102), iFogSim (212), and MyiFogSim (213), are available for edge computing that can provide insight into basic design choices. However, more complex integration tests and application component specific analysis cannot be performed on simulators which can, thus, be only a fallback solution when real or emulated benchmarking infrastructure

is not available.

8. Conclusions

This survey: (i) Identified the trend in benchmarking tightly coupled systems to more loosely coupled systems (for example, edge computing systems). (ii) Cataloged edge benchmarking efforts and examined key contributions in this nascent area. (iii) Classified edge benchmarking research across four dimensions, namely system under test, techniques analyzed, quality, and benchmark runtime. Multiple gaps have become evident, and hence, directions for future research are presented.

A. Future Research Directions. The following seven avenues for pursuing edge benchmarking research are highlighted:

i. Widening the scale of geo-distribution: Many existing benchmarks are designed for capturing the performance of single cloud and edge resources in a lab-based test-bed. While this has provided initial insight, more comprehensive edge benchmarking will need to fully embrace geo-distribution – benchmarking large-scale collection of cloud and edge resources.

ii. Developing edge specific quality and performance metrics and measurement techniques: Current efforts mostly focus on capturing metrics that were historically relevant to distributed systems, such as grids and clouds. While these are useful, it is probable that edge specific metrics given the transient and massively dispersed nature of the environment have not yet been articulated. In addition, novel techniques to capture these metrics may be required.

iii. Evaluating benchmarks on real-world infrastructure: Lab-based infrastructure is the most common test-beds used to evaluate edge benchmarks. This in part may be attributed to the limited access to real test-beds. Nonetheless, effort is required to evaluate existing benchmarks on real and resource-rich test-beds for identifying limitations in current benchmarking approaches.

iv. Benchmarking edge pricing models: The impact of different pricing models involving static and dynamic costs are currently unknown. Benchmarks that consider the trade-offs between application performance (measured by communication latency, throughput etc) and monetary costs (offloading compute, data analysis given location and transfer, and maintenance) need to be developed.

v. Security/privacy specific edge benchmarks: Edge systems are undoubtedly more complex than previous iterations of distributed systems (volume of devices connected, heterogeneity of resources and technological domains spanned, and of edge resources that are accessible for compute, which were previously unavailable or concealed within networks). This naturally exposes a large attack surface and creates multiple vulnerable spots for data privacy. Benchmarks that provide insight in identifying security mechanisms for orchestrating services and suitable security standards for complex systems would prove to be useful.

vi. Developing light-weight and rapid edge benchmarks: Edge and mobile resources in general have limited capabilities to execute typical and extensive benchmark applications designed for large data center servers. Many HPC applications have currently been used to capture the performance of CPUs and accelerators for edge computing, but they are time-consuming. Also, running unmodified Spark or Hadoop applications for big data platforms requires significant time and resources to get results. Therefore, lightweight benchmarking, both in terms of the actual benchmarks and the measurement technique need to be designed and developed for the edge.

vii. Developing standardized benchmarks for capturing offload performance: The premise of edge relies on offloaded tasks either from the cloud or from the devices to the edge to reduce overall execution time an increase energy efficiency. Evaluations presented in the existing literature have used workloads and metrics relevant to specific platforms or testbeds. They are not standard benchmarks or compatible across different infrastructure. Therefore, a more comprehensive and standard approach for benchmarking offload mechanisms need to be considered.

ACKNOWLEDGMENTS. The first author is supported by funds from Rakuten Mobile, Japan and by a Royal Society Short Industry Fellowship.

1. B Varghese, R Buyya, Next Generation Cloud Computing: New Trends and Research Directions. *Futur. Gener. Comput. Syst.* **79**, 849–861 (2018).
2. I Foster, C Kesselman, eds., *The Grid: Blueprint for a New Computing Infrastructure*. (Morgan Kaufmann Publishers Inc.), (1998).
3. B Varghese, et al., Cloud Futurology. *Computer* **52**, 68–77 (2019).
4. B Varghese, N Wang, S Barbhuiya, P Kilpatrick, DS Nikolopoulos, Challenges and Opportunities in Edge Computing in *Proceedings of the IEEE International Conference on Smart Cloud*. pp. 20–26 (2016).
5. W Shi, J Cao, Q Zhang, Y Li, L Xu, Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **3**, 637–646 (2016).
6. M Satyanarayanan, The Emergence of Edge Computing. *Computer* **50**, 30–39 (2017).
7. N Wang, B Varghese, M Matthaiou, DS Nikolopoulos, ENORM: A Framework for Edge Node Resource Management. *IEEE Transactions on Serv. Comput.* (2017).
8. N Wang, M Matthaiou, DS Nikolopoulos, B Varghese, DYVERSE: DYnamic VERtical Scaling in Multi-tenant Edge Environments. *Futur. Gener. Comput. Syst.* **108**, 598–612 (2020).
9. J McChesney, N Wang, A Tanwer, E de Lara, B Varghese, DeFog: Fog Computing Benchmarks in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. p. 47–58 (2019).
10. CH Hong, B Varghese, Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms. *ACM Comput. Surv.* **52** (2019).
11. D Bermbach, et al., A research perspective on fog computing in *Proceedings of the 2nd Workshop on IoT Systems Provisioning and Management for Context-Aware Smart Cities*. (2017).
12. D Bermbach, E Wittern, S Tai, *Cloud Service Benchmarking: Measuring Quality of Cloud Services from a Client Perspective*. (Springer), (2017).
13. HJ Curnow, BA Wichmann, A Synthetic Benchmark. *The Comput. J.* **19**, 43–49 (1976).
14. RP Weicker, Dhrystone: A Synthetic Systems Programming Benchmark. *Commun. ACM* **27**, 1013–1030 (1984).
15. JJ Dongarra, P Luszczek, A Petitet, The LINPACK Benchmark: Past, Present and Future. *Concurr. Comput. Pract. Exp.* **15**, 803–820 (2003).
16. DH Bailey, et al., The NAS Parallel Benchmarks—Summary and Preliminary Results in *Proceedings of the ACM/IEEE Conference on Supercomputing*. p. 158–165 (1991).
17. R Eigenmann, S Hassanzadeh, Benchmarking with Real Industrial Applications: The SPEC High-Performance Group. *IEEE Comput. Sci. Eng.* **3**, 18–23 (1996).
18. S Sharma, C.-H. Hsu, W.-C. Feng, Making a Case for a Green500 List in *Proceedings of the 20th IEEE International Parallel Distributed Processing Symposium*. (2006).
19. PR Luszczek, et al., The HPC Challenge (HPCC) Benchmark Suite in *Proceedings of the ACM/IEEE Conference on Supercomputing*. (2006).
20. JD McCalpin, Memory Bandwidth and Machine Balance in Current High Performance Computers. *IEEE Tech. Comm. on Comput. Archit. Newsl.* (1995).
21. MA Heroux, JJ Dongarra, Toward a New Metric for Ranking High Performance Computing Systems, (Sandia National Lab), Technical Report SAND2013-4744 (2013).
22. S Che, et al., Rodinia: A Benchmark Suite for Heterogeneous Computing in *Proceedings of the IEEE International Symposium on Workload Characterization*. pp. 44–54 (2009).
23. A Danalis, et al., The Scalable Heterogeneous Computing (SHOC) Benchmark Suite in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*. p. 63–74 (2010).
24. B Hu, CJ Rossbach, Mirovia: A Benchmarking Suite for Modern Heterogeneous Computing. *CoRR* **abs/1906.10347** (2019).
25. A Snavely, G Chun, H Casanova, RFV der Wijngaart, M Frumkin, Benchmarks for Grid Computing: A Review of Ongoing Efforts and Future Directions. *SIGMETRICS Perform. Eval. Rev.* **30**, 27–32 (2003).
26. M Frumkin, RF Van der Wijngaart, NAS Grid Benchmarks: A Tool for Grid Space Exploration in *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*. pp. 315–322 (2001).
27. G Tsouloupas, MD Dikaiakos, GridBench: A Tool for Benchmarking Grids in *Proceedings of the 4th International Workshop on Grid Computing*. (2003).
28. C Dumitrescu, I Raicu, M Ripeanu, I Foster, DiPerF: An Automated Distributed Performance Testing Framework in *Proceedings of the IEEE/ACM International Workshop on Grid Computing*. pp. 289–296 (2004).

29. A Iosup, D Epema, GRENCHMARK: A Framework for Analyzing, Testing, and Comparing Grids in *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*. pp. 313–320 (2006).
30. MI Andreica, et al., Towards ServMark, an Architecture for Testing Grid Services. *Technical University of Delft - Technical Report ServMark-2006-002, July 2006* (2006).
31. F Nadeem, R Prodan, T Fahringer, A Iosup, *Benchmarking Grid Applications*. (Springer US, Boston, MA), pp. 19–37 (2008).
32. X Yang, X Li, Y Ji, M Sha, CROWNBench: A Grid Performance Testing System Using Customizable Synthetic Workload in *Progress in WWW Research and Development*, eds. Y Zhang, G Yu, E Bertino, G Xu. (Springer Berlin Heidelberg), pp. 190–201 (2008).
33. A Gaikwad, V Doan, M Bossy, F Baude, F Abergel, SuperQuant Financial Benchmark Suite for Performance Analysis of Grid Middlewares in *Modeling, Simulation and Optimization of Complex Processes*, eds. HG Bock, XP Hoang, R Rannacher, JP Schlöder. (Springer Berlin Heidelberg), pp. 103–113 (2012).
34. B Schroeder, A Wierman, M Harchol-Balter, Open Versus Closed: A Cautionary Tale in *Proceedings of the 3rd Conference on Networked Systems Design & Implementation*. (2006).
35. BF Cooper, A Silberstein, E Tam, R Ramakrishnan, R Sears, Benchmarking Cloud Serving Systems with YCSB in *Proceedings of the 1st ACM Symposium on Cloud Computing*. p. 143–154 (2010).
36. D Kossmann, T Kraska, S Loesing, An Evaluation of Alternative Architectures for Transaction Processing in the Cloud in *Proceedings of the 30th International Conference on Management of Data*. pp. 579–590 (2010).
37. S Patil, et al., YCSB++: Benchmarking and Performance Debugging Advanced Features in Scalable Table Stores in *Proceedings of the 2nd Symposium on Cloud Computing*. pp. 9:1–9:14 (2011).
38. A Iosup, et al., Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. *IEEE Transactions on Parallel Distributed Syst.* **22**, 931–945 (2011).
39. B Varghese, O Akgun, I Miguel, L Thai, A Barker, Cloud Benchmarking for Performance in *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science*. pp. 535–540 (2014).
40. B Varghese, O Akgun, I Miguel, L Thai, A Barker, Cloud Benchmarking for Maximising Performance of Scientific Applications. *IEEE Transactions on Cloud Comput.* **7**, 170–182 (2019).
41. AH Borhani, P Leitner, BS Lee, X Li, T Hung, WPress: An Application-Driven Performance Benchmark for Cloud-Based Virtual Machines. *Proc. IEEE 18th Int. Enterp. Distributed Object Comput. Conf.*, 101–109 (2014).
42. L Gillam, B Li, J O'Loughlin, APS Tomar, Cloud Benchmarking for Maximising Performance of Scientific Applications. *J. Cloud Comput. Adv. Syst. Appl.* **2** (2013).
43. D Bermbach, S Tai, 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*. pp. 1:1–1:6 (2011).
44. D Bermbach, S Tai, Benchmarking eventual consistency: Lessons learned from long-term experimental studies in *Proceedings of the 2nd International Conference on Cloud Engineering*. pp. 47–56 (2014).
45. C Luo, et al., CloudRank-D: Benchmarking and Ranking Cloud Computing Systems for Data Processing Applications. *Front. Comput. Sci.* **6**, 347–362 (2012).
46. M Ferdman, et al., Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware. *ACM SIGPLAN Notices* **47**, 37–48 (2012).
47. I Drago, E Bocchi, M Mellia, H Slatman, A Pras, Benchmarking Personal Cloud Storage in *Proceedings of the Internet Measurement Conference*. p. 205–212 (2013).
48. DE Difallah, A Pavlo, C Curino, P Cudre-Mauroux, OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases. *Proc. VLDB Endow.* **7**, 277–288 (2013).
49. S Müller, D Bermbach, S Tai, F Pallas, Benchmarking the Performance Impact of Transport Layer Security in Cloud Database Systems in *Proceedings of the 2nd International Conference on Cloud Engineering*. (IEEE), (2014).
50. J Kuhlenkamp, M Klems, O Röss, Benchmarking Scalability and Elasticity of Distributed Database Systems in *Proceedings of the International Conference on Very Large Databases*. pp. 1219–1230 (2014).
51. B Varghese, LT Subba, L Thai, A Barker, Container-Based Cloud Virtual Machine Benchmarking in *Proceedings of the IEEE International Conference on Cloud Engineering*. pp. 192–201 (2016).
52. T Palit, Yongming Shen, M Ferdman, Demystifying Cloud Benchmarking in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*. pp. 122–132 (2016).
53. H Wu, F Liu, RB Lee, Cloud Server Benchmark Suite for Evaluating New Hardware Architectures. *IEEE Comput. Archit. Lett.* (2016).
54. D Bermbach, et al., BenchFoundry: A Benchmarking Framework for Cloud Storage Services in *Proceedings of the 15th International Conference on Service Oriented Computing*. (2017).
55. Y Gan, et al., 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*. p. 3–18 (2019).
56. M Grambow, L Meusel, E Wittern, D Bermbach, Benchmarking Microservice Performance: A Pattern-based Approach in *Proceedings of the 35th ACM Symposium on Applied Computing*. (2020).
57. A Shukla, S Chaturvedi, Y Simmhan, RIoTBench: An IoT Benchmark for Distributed Stream Processing Systems. *Concurr. Comput. Pract. Exp.* **29**, e4257 (2017).
58. C Luo, et al., AIoT Bench: Towards Comprehensive Benchmarking Mobile and Embedded Device Intelligence in *Benchmarking, Measuring, and Optimizing*, eds. C Zheng, J Zhan. (Springer International Publishing), pp. 31–35 (2019).
59. A Das, S Patterson, MP Wittie, EdgeBench: Benchmarking Edge Computing Platforms in *Proceedings of the 4th International Workshop on Serverless Computing*. (2018).
60. N Rajovic, A Rico, N Puzovic, C Adeniyi-Jones, A Ramirez, Tibidabo: Making the Case for an ARM-based HPC System. *Futur. Gener. Comput. Syst.* **36**, 322–334 (2014).
61. F Ramalho, A Neto, 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*. pp. 1–6 (2016).
62. T Davies, C Karlsson, H Liu, C Ding, Z Chen, High Performance LINPACK Benchmark: A Fault Tolerant Implementation without Checkpointing in *Proceedings of the International Conference on Supercomputing*. pp. 162–171 (2011).
63. R Morabito, Virtualization on Internet of Things Edge Devices With Container Technologies: A Performance Evaluation. *IEEE Access* **5**, 8835–8850 (2017).
64. CP Kruger, GP Hancke, Benchmarking Internet of Things Devices in *Proceedings of the 12th IEEE International Conference on Industrial Informatics*. pp. 611–616 (2014).
65. S Sridhar, ME Tolentino, Evaluating Voice Interaction Pipelines at the Edge in *Proceedings of the IEEE International Conference on Edge Computing*. pp. 248–251 (2017).
66. M Schuundefined, CA Boano, M Weber, K Römer, 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*. p. 54–65 (2017).
67. Y Wang, S Liu, X Wu, W Shi, CAVBench: A Benchmark Suite for Connected and Autonomous Vehicles in *Proceedings of the IEEE/ACM Symposium on Edge Computing*. pp. 30–42 (2018).
68. T Hao, et al., Edge AIBench: Towards Comprehensive End-to-end Edge Computing Benchmarking in *Proceedings of the First BenchCouncil International Symposium on Benchmarking, Measuring, and Optimizing*. pp. 23–30 (2018).
69. A Limaye, T Adegbija, HERMIT: A Benchmark Suite for the Internet of Medical Things. *IEEE Internet Things J.* **5**, 4212–4222 (2018).
70. S Caldas, et al., LEAF: A Benchmark for Federated Settings. *CoRR* **abs/1812.01097** (2018).
71. X Zhang, Y Wang, W Shi, pCAMP: Performance Comparison of Machine Learning Packages on the Edges in *Proceedings of the USENIX Workshop on Hot Topics in Edge Computing*. (2018).
72. MOJOM noz, J Wang, M Satyanarayanan, J Gross, EdgeDroid: An Experimental Approach to Benchmarking Human-in-the-Loop Applications in *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. p. 93–98 (2019).
73. B Boroujerdian, et al., MAVBench: Micro Aerial Vehicle Benchmarking in *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture*. p. 894–907 (2018).
74. C Lee, M Lin, C Yang, Y Chen, IoTBench: A Benchmark Suite for Intelligent Internet of Things Edge Devices in *Proceedings of the IEEE International Conference on Image Processing*. pp. 170–174 (2019).
75. J Boubin, N Babu, C Stewart, J Chumley, S Zhang, Managing Edge Resources for Fully Autonomous Aerial Systems in *Proceedings of the ACM Symposium on Edge Computing*. (2019).
76. A Reuther, et al., Survey and Benchmarking of Machine Learning Accelerators in *Proceedings of the 24th Annual IEEE High Performance Extreme Computing Conference*. (2019).
77. G Dinelli, G Meoni, E Rapuano, G Benelli, L Fanucci, An FPGA-Based Hardware Accelerator for CNNs Using On-Chip Memories Only: Design and Benchmarking with Intel Movidius Neural Compute Stick. *Int. J. Reconfigurable Comput.* (2019).
78. B Varghese, C Reaño, F Silla, Accelerator Virtualization in Fog Computing: Moving from the Cloud to the Edge. *IEEE Cloud Comput.* **5**, 28–37 (2018).
79. MH Ionica, D Gregg, The movidius myriad architecture's potential for scientific computing. *IEEE Micro* **35**, 6–14 (2015).
80. S Cass, Taking AI to the Edge: Google's TPU now Comes in a Maker-friendly Package. *IEEE Spectr.* **56**, 16–17 (2019).
81. D Moloney, 1TOPS/W Software Programmable Media Processor in *Proceedings of the IEEE Symposium on Hot Chips*. pp. 1–24 (2011).
82. N Jouppi, C Young, N Patil, D Patterson, Motivation for and Evaluation of the First Tensor Processing Unit. *IEEE Micro* **38**, 10–19 (2018).
83. X Glorot, Y Bengio, Understanding the Difficulty of Training Deep Feedforward Neural Networks in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. pp. 249–256 (2010).
84. S Mittal, Survey of FPGA-based Accelerators for Convolutional Neural Networks. *Neural Comput. Appl.*, 1–31 (2018).
85. VJ Reddi, et al., MLPerf Inference Benchmark. *arXiv preprint arXiv:1911.02549* (2019).
86. AG Howard, et al., Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861* (2017).
87. C Szegedy, et al., Going Deeper with Convolutions in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1–9 (2015).
88. LA Libutti, FD Igual, L Pinuel, LD Giusti, M Naiouf, Benchmarking Performance and Power of USB Accelerators for Inference with MLPerf in *Proceedings of the 2nd Workshop on Accelerated Machine Learning*. (2020).
89. Z Liu, et al., Tomogan: Low-dose X-ray Tomography with Generative Adversarial Networks. *arXiv preprint arXiv:1902.07582* (2019).
90. Z Liu, T Bicer, R Kettimuthu, I Foster, Deep Learning Accelerated Light Source Experiments in *Proceedings of the IEEE/ACM Third Workshop on Deep Learning on Supercomputers*. pp. 20–28 (2019).
91. V Abeykoon, Z Liu, R Kettimuthu, G Fox, I Foster, Scientific Image Restoration Anywhere in *Proceedings of the IEEE/ACM 1st Annual Workshop on Large-scale Experiment-in-the-Loop Computing*. pp. 8–13 (2019).
92. M Antonini, et al., Resource Characterisation of Personal-Scale Sensing Models on Edge Accelerators in *Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*. pp. 49–55 (2019).
93. L Peng, L Chen, Z Ye, Y Zhang, Aroma: A Deep Multi-task Learning Based Simple and Complex Human Activity Recognition Method Using Wearable Sensors. *Proc. ACM on Interactive, Mobile, Wearable Ubiquitous Technol.* **2**, 1–16 (2018).
94. S Katayama, A Mathur, T Okoshi, J Nakazawa, F Kawsar, Situation-aware Conversational Agent with Kinetic Earables in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. pp. 657–658 (2019).

95. A Mathur, A Isopoussu, F Kawsar, N Berthouze, ND Lane, Mic2Mic: Using Cycle-consistent Generative Adversarial Networks to Overcome Microphone Variability in Speech Systems in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*. pp. 169–180 (2019).
96. FN Iandola, et al., SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and< 0.5 MB Model Size. *arXiv preprint arXiv:1602.07360* (2016).
97. M Tan, QV Le, Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv preprint arXiv:1905.11946* (2019).
98. G Huang, Z Liu, LVD Maaten, KQ Weinberger, Densely Connected Convolutional Networks in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 4700–4708 (2017).
99. G Benelli, G Meoni, L Fanucci, 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*. pp. 267–272 (2018).
100. B Amento, et al., FocusStack: Orchestrating Edge Clouds Using Location-Based Focus of Attention in *Proceedings of IEEE/ACM Symposium on Edge Computing*. pp. 179–191 (2016).
101. C Sonmez, A Ozgovde, C Ersoy, Fuzzy Workload Orchestration for Edge Computing. *IEEE Transactions on Netw. Serv. Manag.* **16**, 769–782 (2019).
102. C Sonmez, A Ozgovde, C Ersoy, EdgeCloudSim: An Environment for Performance Evaluation of Edge Computing Systems in *International Conference on Fog and Mobile Edge Computing*. pp. 39–44 (2017).
103. CH Hong, K Lee, M Kang, C Yoo, qCon: QoS-Aware Network Resource Management for Fog Computing. *Sensors* **18** (2018).
104. R Mahmud, R Kotagiri, R Buyya, Fog computing: A Taxonomy, Survey and Future Directions in *Internet of Everything*. (Springer), pp. 103–130 (2018).
105. M Ryden, K Oh, A Chandra, J Weissman, Nebula: Distributed Edge Cloud for Data Intensive Computing in *IEEE International Conference on Cloud Engineering*. pp. 57–66 (2014).
106. D Komosny, et al., On Geographic Coordinates of PlanetLab Europe in *Proceedings of the 38th International Conference on Telecommunications and Signal Processing*. pp. 642–646 (2015).
107. K Habak, M Ammar, KA Harras, E Zegura, Femto Clouds: Leveraging Mobile Devices to Provide Cloud Service at the Edge in *Proceedings of the IEEE 8th International Conference on Cloud Computing*. pp. 9–16 (2015).
108. H Yu, Q Wang, S Guo, Energy-efficient Task Offloading and Resource Scheduling for Mobile Edge Computing in *Proceedings of the IEEE International Conference on Networking, Architecture and Storage*. pp. 1–4 (2018).
109. X Li, et al., A Hybrid Computing Solution and Resource Scheduling Strategy for Edge Computing in Smart Manufacturing. *IEEE Transactions on Ind. Informatics* **15**, 4225–4234 (2019).
110. A Madej, N Wang, N Athanasopoulos, R Ranjan, B Varghese, Priority-based Fair Scheduling in Edge Computing in *Proceedings of the International Conference on Fog and Edge Computing*. (2020).
111. C You, K Huang, H Chae, BH Kim, Energy-efficient Resource Allocation for Mobile-Edge Computation Offloading. *IEEE Transactions on Wirel. Commun.* **16**, 1397–1411 (2016).
112. J Guo, Z Song, Y Cui, Z Liu, Y Ji, Energy-efficient Resource Allocation for Multi-user Mobile Edge Computing in *Proceedings of the IEEE Global Communications Conference*. pp. 1–7 (2017).
113. S Sardellitti, G Scutari, S Barbarossa, Joint Optimisation of Radio and Computational Resources for Multicell Mobile-Edge Computing. *IEEE Transactions on Signal Inf. Process. over Networks* **1**, 89–103 (2015).
114. P Zhao, H Tian, C Qin, G Nie, Energy-saving Offloading by Jointly Allocating Radio and Computational Resources for Mobile Edge Computing. *IEEE Access* **5**, 11255–11268 (2017).
115. Y Chen, et al., Energy Efficient Dynamic Offloading in Mobile Edge Computing for Internet of Things. *IEEE Transactions on Cloud Comput.* (2019).
116. K Zhang, et al., Energy-efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks. *IEEE Access* **4**, 5896–5907 (2016).
117. O Skarlat, M Nardelli, S Schulte, M Borkowski, P Leitner, Optimized IoT Service Placement in the Fog. *Serv. Oriented Comput. Appl.* **11**, 427–443 (2017).
118. Y Song, S Yau, R Yu, X Zhang, G Xue, An Approach to Qos-based Task Distribution in Edge Computing Networks for IoT Applications in *Proceedings of the IEEE International Conference on Edge Computing*. (IEEE), pp. 32–39 (2017).
119. T Bahreini, D Grosu, Efficient Placement of Multi-component Applications in Edge Computing Systems in *Proceedings of the ACM/IEEE Symposium on Edge Computing*. (2017).
120. K Katsalis, T Papaioannou, N Nikaein, L Tassiulas, SLA-driven VM Scheduling in Mobile Edge Computing in *Proceedings of the IEEE International Conference on Cloud Computing*. pp. 750–757 (2016).
121. M Aazam, M St-Hilaire, C Lung, I Lambadaris, MeFoRE: QoE Based Resource Estimation at Fog to Enhance QoS in IoT in *Proceedings of the International Conference on Telecommunications*. pp. 1–5 (2016).
122. A Brogi, S Forti, QoS-aware Deployment of IoT Applications Through the Fog. *IEEE Internet Things J.* **4**, 1185–1192 (2017).
123. L Bittencourt, J Diaz-Montes, R Buyya, O Rana, M Parashar, Mobility-aware Application Scheduling in Fog Computing. *IEEE Cloud Comput.* **4**, 26–35 (2017).
124. K Zhang, Y Mao, S Leng, S Maharjan, Y Zhang, Optimal Delay Constrained Offloading for Vehicular Edge Computing Networks in *Proceedings of the IEEE International Conference on Communications*. (IEEE), pp. 1–6 (2017).
125. L Wang, L Jiao, T He, J Li, M Mühlhäuser, Service Entity Placement for Social Virtual Reality Applications in Edge Computing in *Proceedings of the IEEE Conference on Computer Communications*. (IEEE), pp. 468–476 (2018).
126. O Skarlat, M Nardelli, S Schulte, S Dustdar, Towards QoS-aware Fog Service Placement in *Proceedings of the IEEE International Conference on Fog and Edge Computing*. (IEEE), pp. 89–96 (2017).
127. N Sharghivand, F Derakhshan, L Mashayekhy, QoS-aware Matching of Edge Computing Services to Internet of Things in *Proceedings of the IEEE International Performance Computing and Communications Conference*. (IEEE), pp. 1–8 (2018).
128. T Ouyang, Z Zhou, X Chen, Follow Me at the Edge: Mobility-aware Dynamic Service Placement for Mobile Edge Computing. *IEEE J. on Sel. Areas Commun.* **36**, 2333–2345 (2018).
129. M Chen, et al., A Dynamic Service Migration Mechanism in Edge Cognitive Computing. *ACM Transactions on Internet Technol.* **19**, 1–15 (2019).
130. Y Jiao, P Wang, D Niyato, Z Xiong, Social Welfare Maximization Auction in Edge Computing Resource Allocation for Mobile Blockchain in *Proceedings of the IEEE International Conference on Communications*. pp. 1–6 (2018).
131. DT Nguyen, LB Le, V Bhargava, Price-based Resource Allocation for Edge Computing: A Market Equilibrium Approach. *IEEE Transactions on Cloud Comput.* (2018).
132. C Do, et al., A Proximal Algorithm for Joint Resource Allocation and Minimizing Carbon Footprint in Geo-distributed Fog Computing in *Proceedings of the International Conference on Information Networking*. pp. 324–329 (2015).
133. H Zhang, et al., Computing Resource Allocation in Three-tier IoT Fog Networks: A Joint Optimization Approach Combining Stackelberg Game and Matching. *IEEE Internet Things J.* **4**, 1204–1215 (2017).
134. J Zhang, W Xia, F Yan, L Shen, Joint Computation Offloading and Resource Allocation Optimization in Heterogeneous Networks with Mobile Edge Computing. *IEEE Access* **6**, 19324–19337 (2018).
135. M Taneja, A Davy, Resource Aware Placement of IoT Application Modules in Fog-Cloud Computing Paradigm in *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management*. pp. 1222–1228 (2017).
136. W Zhan, et al., Mobility-Aware Multi-User Offloading Optimization for Mobile Edge Computing. *IEEE Transactions on Veh. Technol.* (2020).
137. M Chen, B Liang, M Dong, Joint Offloading Decision and Resource Allocation for Multi-user Multi-task Mobile Cloud in *Proceedings of the IEEE International Conference on Communications*. (IEEE), pp. 1–6 (2016).
138. X Chen, Decentralized Computation Offloading Game for Mobile Cloud Computing. *IEEE Transactions on Parallel Distributed Syst.* **26**, 974–983 (2014).
139. M Chen, M Dong, B Liang, Multi-user Mobile Cloud Offloading Game with Computing Access Point in *Proceedings of the IEEE International Conference on Cloud Networking*. (IEEE), pp. 64–69 (2016).
140. L Villas, A Boukerche, HD Oliveira, RD Araujo, A Loureiro, A Spatial Correlation Aware Algorithm to Perform Efficient Data Collection in Wireless Sensor Networks. *Ad Hoc Networks* **12**, 69 – 85 (2014).
141. Y Yao, Q Cao, A Vasilakos, 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*. pp. 182–190 (2013).
142. Y Lin, B Kemme, M Patino-Martinez, R Jimenez-Peris, Enhancing Edge Computing with Database Replication in *Proceedings of the IEEE International Symposium on Reliable Distributed Systems*. (IEEE), pp. 45–54 (2007).
143. L Gao, M Dahlin, A Nayate, J Zheng, A Iyengar, Application Specific Data Replication for Edge Services in *Proceedings of the International Conference on World Wide Web*. (ACM), pp. 449–460 (2003).
144. Z Chen, et al., Early Implementation Experience with Wearable Cognitive Assistance Applications in *Proceedings of the Workshop on Wearable Systems and Applications*. (ACM), pp. 33–38 (2015).
145. P Liu, D Willis, S Banerjee, ParaDrop: Enabling Lightweight Multi-tenancy at the Network's Extreme Edge in *Proceedings of IEEE/ACM Symposium on Edge Computing*. pp. 1–13 (2016).
146. L Ma, S Yi, Q Li, Efficient Service Handoff Across Edge Servers via Docker Container Migration in *Proceedings of the ACM/IEEE Symposium on Edge Computing*. pp. 1–13 (2017).
147. D Roy, D De, A Mukherjee, R Buyya, Application-aware Cloudlet Selection for Computation Offloading in Multi-cloudlet Environment. *The J. Supercomput.*, 1–19 (2016).
148. X Chen, Q Shi, L Yang, J Xu, ThriftyEdge: Resource-efficient Edge Computing for Intelligent IoT Applications. *IEEE Netw.* **32**, 61–65 (2018).
149. S Ma, S Guo, K Wang, W Jia, M Guo, A Cyclic Game for Service-Oriented Resource Allocation in Edge Computing. *IEEE Transactions on Serv. Comput.* (2020).
150. B Jia, H Hu, Y Zeng, T Xu, Y Yang, Double-matching Resource Allocation Strategy in Fog Computing Networks Based on Cost Efficiency. *J. Commun. Networks* **20**, 237–246 (2018).
151. X Ma, et al., Cost-efficient Workload Scheduling in Cloud Assisted Mobile Edge Computing in *Proceedings of the IEEE/ACM International Symposium on Quality of Service*. (IEEE), pp. 1–10 (2017).
152. X Chen, L Jiao, W Li, X Fu, Efficient Multi-user Computation Offloading for Mobile-edge Cloud Computing. *IEEE/ACM Transactions on Netw.* **24**, 2795–2808 (2015).
153. L Ni, J Zhang, C Jiang, C Yan, K Yu, Resource Allocation Strategy in Fog Computing Based on Priced Timed Petri Nets. *IEEE Internet Things J.* **4**, 1216–1228 (2017).
154. Y Gu, Z Chang, M Pan, L Song, Z Han, Joint Radio and Computational Resource Allocation in IoT Fog Computing. *IEEE Transactions on Veh. Technol.* **67**, 7475–7484 (2018).
155. A Majeed, P Kilpatrick, I Spence, B Varghese, Modelling Fog Offloading Performance in *Proceedings of the IEEE International Conference on Fog and Edge Computing*. (2020).
156. J Zhao, Q Li, Y Gong, K Zhang, Computation Offloading and Resource Allocation for Cloud Assisted Mobile Edge Computing in Vehicular Networks. *IEEE Transactions on Veh. Technol.* **68**, 7944–7956 (2019).
157. Y Xiao, M Krunz, QoE and Power Efficiency Tradeoff for Fog Computing Networks with Fog Node Cooperation in *Proceedings of the IEEE Conference on Computer Communications*. (IEEE), pp. 1–9 (2017).
158. M Satyanarayanan, P Bahl, R Caceres, N Davies, The Case for VM-based Cloudlets in Mobile Computing. *IEEE Pervasive Comput.* **8**, 14–23 (2009).
159. W Felter, A Ferreira, R Rajamony, J Rubio, An Updated Performance Comparison of Virtual Machines and Linux Containers in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*. pp. 171–172 (2015).

160. B Ismail, et al., Evaluation of Docker as Edge Computing Platform in *Proceedings of the IEEE Conference on Open Systems*. (IEEE), pp. 130–135 (2015).
161. L Yin, J Luo, H Luo, Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing. *IEEE Transactions on Ind. Informatics* **14**, 4712–4721 (2018).
162. C Dupont, R Giaffreda, L Capra, Edge Computing in IoT Context: Horizontal and Vertical Linux Container Migration in *Proceedings of the Global Internet of Things Summit*. (IEEE), pp. 1–4 (2017).
163. A Madhavapeddy, et al., Unikernels: Library Operating Systems for the Cloud. *ACM SIGARCH Comput. Archit. News* **41**, 461–472 (2013).
164. R Morabito, V Cozzolino, A Ding, N Beijar, J Ott, Consolidate IoT Edge Computing with Lightweight Virtualization. *IEEE Netw.* **32**, 102–111 (2018).
165. C Kaner, WP Bond, Software Engineering Metrics: What Do They Measure and How Do We Know? in *Proceedings of the 10th International Software Metrics Symposium*. (2004).
166. D Bermbach, Ph.D. thesis (Karlsruhe Institute of Technology) (2014).
167. D Bermbach, J Kuhlenkamp, A Dey, S Sakr, R Nambiar, Towards an Extensible Middleware for Database Benchmarking in *Proceedings of TPC Technology Conference on Performance Evaluation and Benchmarking*. pp. 82–96 (2014).
168. D Bermbach, S Maghsudi, J Hasenburg, T Pfandzelter, Towards Auction-Based Function Placement in Serverless Fog Platforms in *Proceedings of the Second IEEE International Conference on Fog Computing*. (2020).
169. J Scheuner, P Leitner, J Cito, H Gall, Cloud WorkBench – Infrastructure-as-Code Based Cloud Benchmarking in *Proceedings of the IEEE 6th International Conference on Cloud Computing Technology and Science*. pp. 246–253 (2014).
170. J Scheuner, P Leitner, Performance Benchmarking of Infrastructure-as-a-Service (IaaS) Clouds with Cloud WorkBench in *Companion of the 2019 ACM/SPEC International Conference on Performance Engineering*. pp. 53–56 (2019).
171. A Lenk, M Menzel, J Lipsky, S Tai, P Offermann, What are you Paying for? Performance Benchmarking for Infrastructure-as-a-service Offerings in *Proceedings of the IEEE International Conference on Cloud Computing*. pp. 484–491 (2011).
172. J Kuhlenkamp, K Rudolph, D Bermbach, AISLE: Assessment of Provisioned Service Levels in Public IaaS-Based Database Systems in *Proceedings of the 13th International Conference on Service-Oriented Computing*. pp. 154–168 (2015).
173. J Kuhlenkamp, S Werner, Benchmarking FaaS Platforms: Call for Community Participation in *Proceedings of the 4th International Workshop on Serverless Computing*. (2018).
174. J Kuhlenkamp, S Werner, MC Borges, KE Tal, S Tai, An Evaluation of FaaS Platforms as a Foundation for Serverless Big Data Processing in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*. (2019).
175. W Lloyd, S Ramesh, S Chinthalapati, L Ly, S Pallickara, Serverless computing: An Investigation of Factors Influencing Microservice Performance in *Proceedings of the IEEE International Conference on Cloud Engineering*. pp. 159–169 (2018).
176. AV Hoorn, J Waller, W Hasselbring, Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*. pp. 247–248 (2012).
177. M Grambow, F Lehmann, D Bermbach, Continuous Benchmarking: Using System Benchmarking in Build Pipelines in *Proceedings of the 1st Workshop on Service Quality and Quantitative Evaluation in new Emerging Technologies*. (2019).
178. C Binnig, D Kossmann, T Kraska, S Loesing, How is the Weather Tomorrow? Towards a Benchmark for the Cloud in *Proceedings of the 2nd International Workshop on Testing Database Systems*. pp. 1–6 (2009).
179. T Rabl, et al., Solving Big Data Challenges for Enterprise Application Performance Management. *Proc. VLDB Endow.* **5** (2012).
180. S Islam, K Lee, A Fekete, A Liu, How a Consumer Can Measure Elasticity for Cloud Platforms in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*. p. 85–96 (2012).
181. J Kuhlenkamp, S Werner, MC Borges, D Ernst, D Wenzel, 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*. (2020).
182. D Bermbach, E Wittern, Benchmarking Web API Quality in *Proceedings of the 16th International Conference on Web Engineering*. pp. 188–206 (2016).
183. L Shao, et al., User-perceived Service Availability: A Metric and an Estimation Approach in *Proceedings of the IEEE International Conference on Web Services*. pp. 647–654 (2009).
184. M Toeroe, F Tam, *Service Availability: Principles and Practice*. (John Wiley & Sons), (2012).
185. T Hauer, P Hoffmann, J Lunney, D Ardelean, A Diwan, Meaningful Availability in *17th USENIX Symposium on Networked Systems Design and Implementation*. pp. 545–557 (2020).
186. MR Ulbricht, F Pallas, YaPPL - A Lightweight Privacy Preference Language for Legally Sufficient and Automated Consent Provision in IoT Scenarios in *Proceedings of Data Privacy Management*, LNCS, eds. G. Livraga, R. Rios. (Springer International Publishing), Vol. 11025, (2018).
187. F Pallas, P Raschke, D Bermbach, Fog Computing as Privacy Enabler. *IEEE Internet Comput.* (2020).
188. D Bermbach, E Wittern, Benchmarking Web API Quality-Revisited. *arXiv preprint arXiv:1903.07712* (2019).
189. F Pallas, D Bermbach, S Müller, S Tai, Evidence-Based Security Configurations for Cloud Datastores in *Proceedings of the the 32nd ACM Symposium on Applied Computing*. (2017).
190. F Pallas, J Günther, D Bermbach, Pick your Choice in HBase: Security or Performance in *Proceedings of the IEEE International Conference on Big Data*. (2017).
191. F Pallas, M Grambow, Three Tales of Disillusion: Benchmarking Property Preserving Encryption Schemes in *Proceedings of the 15th International Conference on Trust, Privacy and Security in Digital Business*. pp. 39–54 (2018).
192. G Apostolopoulos, V Peris, D Saha, 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, pp. 717–725 (1999).
193. K Kant, R Iyer, P Mohapatra, Architectural Impact of Secure Socket Layer on Internet Servers in *Proceedings of the International Conference on Computer Design*. pp. 7–14 (2000).
194. C Coarfa, P Druschel, DS Wallach, Performance Analysis of TLS Web Servers. *ACM Transactions on Comput. Syst.* **24**, 39–69 (2006).
195. S Shirasuna, A Slominski, L Fang, D Gannon, Performance comparison of security mechanisms for grid services in *Fifth IEEE/ACM International Workshop on Grid Computing*. pp. 360–364 (2004).
196. S Shastri, V Banakar, M Wasserman, A Kumar, V Chidambaram, Understanding and Benchmarking the Impact of GDPR on Database Systems. *arXiv preprint arXiv:1910.00728* (2019).
197. W Vogels, Eventually Consistent. *ACM Queue* **6**, 14–19 (2008).
198. D Bermbach, J Kuhlenkamp, Consistency in Distributed Storage Systems: An Overview of Models, Metrics and Measurement Approaches in *Networked Systems*, Lecture Notes in Computer Science, eds. V Gramoli, R Guerraoui. (Springer Berlin Heidelberg) Vol. 7853, pp. 175–189 (2013).
199. K Zellag, B Kemme, How Consistent is Your Cloud Application? in *Proceedings of the 3rd Symposium on Cloud Computing*. pp. 6:1–6:14 (2012).
200. H Wada, A Fekete, L Zhao, K Lee, A Liu, 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*. pp. 134–143 (2011).
201. W Golab, X Li, MA Shah, Analyzing Consistency Properties for Fun and Profit in *Proceedings of the 30th Symposium on Principles of Distributed Computing*. pp. 197–206 (2011).
202. E Anderson, X Li, MA Shah, J Tucek, JJ Wylie, What Consistency Does Your Key-value Store Actually Provide? in *Proceedings of the 6th Workshop on Hot Topics in System Dependability*. pp. 1–16 (2010).
203. B Varghese, S Ray, BN Gohil, S Vega, Research Challenges in Query Processing and Data Analytics on the Edge in *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*. p. 317–322 (2019).
204. D Abadi, Consistency Tradeoffs in Modern Distributed Database System Design: CAP is Only Part of the Story. *IEEE Comput.* **45**, 37–42 (2012).
205. A Burns, AJ Wellings, *Real-time Systems and Programming Languages: Ada 95, Real-time Java, and Real-time POSIX*. (Pearson Education), (2001).
206. J Wang, et al., Towards Scalable Edge-native Applications in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. pp. 152–165 (2019).
207. Y Kang, et al., 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*. p. 615–629 (2017).
208. BP Rimal, M Maier, M Satyanarayanan, Experimental Testbed for Edge Computing in Fiber-Wireless Broadband Access Networks. *IEEE Commun. Mag.* **56**, 160–167 (2018).
209. R Muñoz, et al., 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*. pp. 1–5 (2017).
210. FP Tso, DR White, S Jouet, J Singer, DP Pezaros, The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures in *Proceedings of the First International Workshop on Resource Management of Cloud Computing*. pp. 108–112 (2013).
211. J Hasenburg, M Grambow, E Grunewald, S Huk, D Bermbach, MockFog: Emulating Fog Computing Infrastructure in the Cloud in *Proceedings of the First IEEE International Conference on Fog Computing*. (2019).
212. H Gupta, AV Dastjerdi, SK Ghosh, R Buyya, iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments. *Software: Pract. Exp.* **47**, 1275–1296 (2017).
213. MM Lopes, WA Higashino, MAM Capretz, LF Bittencourt, MyiFogSim: A Simulator for Virtual Machine Migration in Fog Computing in *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*. pp. 47–52 (2017).