
This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Rajput, Kaustubh Rajendra; Kulkarni, Chinmay Dilip; Cho, Byungjin; Wang, Wei; Kim, In Kee
EdgeFaaS Bench: Benchmarking Edge Devices Using Serverless Computing

Published in:
IEEE International Conference on Edge Computing and Communications (EDGE)

DOI:
[10.1109/EDGE55608.2022.00024](https://doi.org/10.1109/EDGE55608.2022.00024)

Published: 01/01/2022

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Rajput, K. R., Kulkarni, C. D., Cho, B., Wang, W., & Kim, I. K. (2022). EdgeFaaS Bench: Benchmarking Edge Devices Using Serverless Computing. In C. A. Ardagna, H. Bian, C. K. Chang, R. N. Chang, E. Damiani, G. Elia, Q. He, V. Puig, R. Ward, F. Xhafa, & J. Zhang (Eds.), *IEEE International Conference on Edge Computing and Communications (EDGE)* (pp. 93-103). (IEEE international conference on edge computing). IEEE. <https://doi.org/10.1109/EDGE55608.2022.00024>

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

EdgeFaaS Bench: Benchmarking Edge Devices Using Serverless Computing

Kaustubh Rajendra Rajput^{*§}, Chinmay Dilip Kulkarni^{*§}, Byungjin Cho[†], Wei Wang[‡], and In Kee Kim^{*}

^{*}University of Georgia, Computer Science, {kaustubh.rajput, chinmay.kulkarni, inkee.kim}@uga.edu

[†]Aalto University, Communications and Networking, byungjin.cho@aalto.fi

[‡]The University of Texas at San Antonio, Computer Science, wei.wang@utsa.edu

Abstract—Due to the development of small-size, energy-efficient, and powerful CPUs and GPUs for single board computers, various edge devices are widely adopted for hosting real-world applications, including real-time object detection, autonomous driving, and sensor stream processing. At the same time, serverless computing receives increasing attention as a new application deployment model because of its simplicity, scalability, event-driven processing, and short-lived computation. Therefore, there is a growing demand for applying serverless computing to edge computing environments. However, due to the lack of characterization of serverless edge computing (e.g., application performance and impact from resource heterogeneity), researchers and practitioners have to conduct tedious measurements to understand the performance of serverless applications on edge devices in non-systematic ways.

We create EdgeFaaS Bench, a novel benchmark suite for serverless computing on edge devices, to bridge this gap. EdgeFaaS Bench is developed on top of Apache OpenFaaS with Docker Swarm and can run various serverless benchmark workloads on edge devices with different hardware specifications (e.g., GPUs). EdgeFaaS Bench contains 14 different benchmark workloads running on heterogeneous edge devices and captures various system-level, application-level, and serverless-specific metrics, including system utilization, response time, cold/warm start times, and impact of concurrent function executions. Experimental studies are conducted on two widely used edge devices, Raspberry Pi 4B and Jetson Nano, to show EdgeFaaS Bench’s capabilities to benchmark serverless computing on edge devices.

Index Terms—Benchmarking and Performance Evaluation; Edge Computing; Serverless Computing;

I. INTRODUCTION

The proliferation of IoT devices has been generating an exponential amount of sensing data, often requiring real-time processing near data sources [1]–[3]. The edge computing paradigm is emerging because the edge cluster can be deployed near data sources, e.g., IoT sensors, and process the data without relying on traditional data-center computing [4]–[6]. In particular, the development of small, energy-efficient, and capable CPU and AI accelerators for edge devices facilitates the adoption of edge computing to servicing various real-world applications, e.g., autonomous driving, drone-based surveillance systems, and environmental sensing [7]–[10].

At the same time, serverless computing and Function-as-a-Service (FaaS) have gained increasing attention as the next-generation application deployment model because of its

simplicity, event-based processing, scalability, and short-lived computation [11]–[17]. Various user-facing and data-intensive applications have been converted into serverless architecture [18]–[22]. Furthermore, there are increasing attempts to apply serverless computing to edge computing environments [23]–[26], and it is widely recognized as serverless computing models are well suited for edge computing’s service offerings [27]–[32].

When serverless applications are designed and developed for edge computing environments, specifically if the serverless applications are intended to be hosted on resource-constrained edge devices, software developers wish to know the potential performances and behaviors of their serverless applications on edge devices. Given the heterogeneity and constrained capacity of resources on edge devices, understanding performance limitations and bottlenecks of serverless applications on edge devices are particularly challenging.

A standard approach to understanding an application’s performance is to use benchmark suites that are designed for a target environment [33], [34]. There are several benchmark suites available for either serverless computing [35]–[38] or edge computing [39]–[44]. Unfortunately, none of the existing benchmark suites was tailored for comprehensively benchmarking characteristics of serverless computing on edge devices. The lack of benchmarks for serverless on edge devices makes it difficult to understand the performance of the target applications and their execution environments. For example, desired serverless applications on edge devices [24], [29], [30] can be significantly different from serverless applications for cloud computing [14]–[16], and serverless edge applications’ workload arrival patterns [45], [46] may also be different from those of serverless computing in clouds [47], [48]. As a result, existing benchmark tools [35]–[44] may not correctly capture the necessary performance and characteristics for serverless deployment on edge devices, and developers and practitioners have to conduct tedious and time-consuming measurement processes to understand the performance of serverless applications on edge devices.

To address this problem, we create EdgeFaaS Bench, a novel benchmark suite for serverless computing on edge devices. EdgeFaaS Bench is developed on top of a widely used open-source FaaS framework and container orchestration tool, OpenFaaS [49] and Docker Swarm [50], and can perform various benchmark tests on edge devices with different hard-

[§]Equal contribution

ware specifications, i.e., arm64/aarch64 devices with GPUs. EdgeFaaS Bench has 14 different serverless workloads performing micro- and application-level benchmarking on edge devices. Micro-benchmark workloads focus on measuring the performance of a specific resource type on the devices, e.g., CPU, memory, network bandwidth, and disk I/O. On the other hand, application-level benchmark workloads are developed based on real-world serverless computing use-cases to capture the performances and characteristics of serverless applications on edge devices. In particular, various machine learning and AI FaaS applications, e.g., image classification, object detection, are developed for EdgeFaaS Bench. With 14 benchmark workloads, EdgeFaaS Bench can measure various system-level, application-level, and serverless-level performances, including system utilization, application response time, cold/warm start times, and impact of concurrent function executions.

To show the effectiveness of EdgeFaaS Bench, a thorough benchmark study is performed on two edge devices; Raspberry Pi 4B [51] and Jetson Nano [52]. Raspberry Pi 4B is a widely used device in edge computing settings, and Jetson Nano is an edge device equipped with small yet powerful GPU accelerators. The evaluation results report application response time, system utilization, cold and warm start times, the impact of concurrent function executions, and the performance comparison between GPU and GPU-based inferences.

The contributions of this work are as follows.

1. **Novel benchmark suite.** We present EdgeFaaS Bench, the first comprehensive benchmark suite centered on evaluating the performance and characteristics of serverless applications on heterogeneous edge devices to the best of our knowledge. EdgeFaaS Bench software is publicly available at <https://github.com/kaustubhrajput46/EdgeFaaS Bench>.

2. **Comprehensive FaaS benchmark workloads.** 14 different workloads are employed for EdgeFaaS Bench to collect various performance aspects of edge devices. EdgeFaaS Bench can measure both system-level and application-level metrics, such as utilization and response times, as well as serverless-specific metrics, such as cold/warm startup time variations and the impact of concurrent function executions.

3. **A thorough evaluation with practical edge devices.** EdgeFaaS Bench is tested on two widely used edge devices, Raspberry Pi 4B and Jetson Nano, to demonstrate the effectiveness and feasibility in performance benchmarking of serverless computing on edge devices.

We structure the rest of the paper as follows. Section II describes the design of EdgeFaaS Bench by emphasizing our benchmark focus. Section III discusses 14 benchmark workloads in EdgeFaaS Bench. Section IV reports the results from a benchmark study on two edge devices with EdgeFaaS Bench. Section V describes related work regarding this work. Finally, Section VI concludes this paper.

II. DESIGN OF EDGEFAASBENCH

A. *Edge Computing Architecture and the Focus of Benchmark Edge Computing Architecture.* Fig. 1 illustrates the general architecture of edge computing. Edge computing architecture

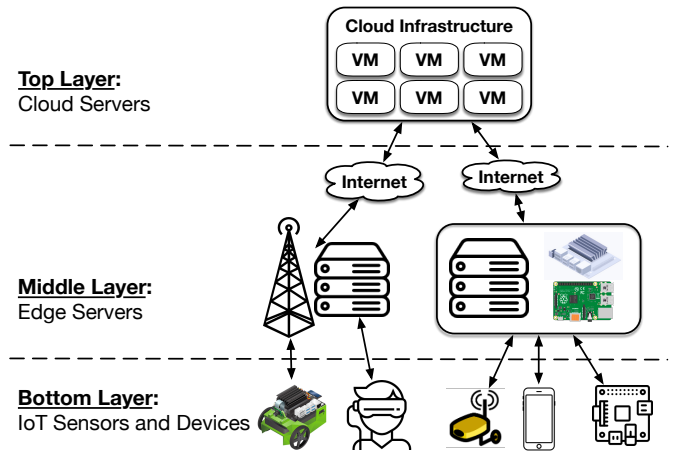


Fig. 1. Edge Computing Architecture

is commonly composed of three layers; 1) IoT sensor and device layer, 2) Edge server layer, and 3) Cloud layer.

IoT sensor and device layer (“bottom layer” in Fig. 1) has various IoT sensors and user devices, which perform diverse sensing operations, and user devices (including edge devices) are used to host lightweight applications to perform real-time on-board processing, including sensing data filtering and noise removal [53], lightweight AI inference tasks [10], [54], and stream processing [55].

Edge server layer (“middle layer” in Fig. 1) is the main computing component in the edge computing environment. Typically, the edge server layer is placed near data sources (IoT sensor and device layer) to provide low-latency computing services that process the data collected by IoT sensors. The edge servers host heavier-weight applications that cannot be executed on computing elements in the IoT sensor and device layer. The applications hosted on edge server layer perform deep learning (DL) inference [7] and big data analytics [56] operations, which tend to require high-performance processors, GPUs, and edge AI accelerators.

The last layer is *cloud server layer* (“top layer” in Fig. 1). Cloud server layer leverages the scalability and elasticity of cloud infrastructure to provide large-scale data storage and perform computation-intensive tasks with specialized HW accelerators, e.g., DL model training with TPU [57], that cannot be performed on the other two layers.

Benchmark Focus of EdgeFaaS Bench. While edge computing has three different layers, EdgeFaaS Bench focuses on benchmarking the performance of edge devices, explicitly adopting the idea of the FaaS paradigm [15], [16]. Moreover, existing edge benchmark suites, e.g., DeFog [41] and others [39], already offer the capability of end-to-end edge benchmarks. Please refer to Section V for more information.

Benchmarking the edge device performance has received significant attention from the research and development community because of the faster improvement of the small but powerful edge devices. In addition, HW developers and man-

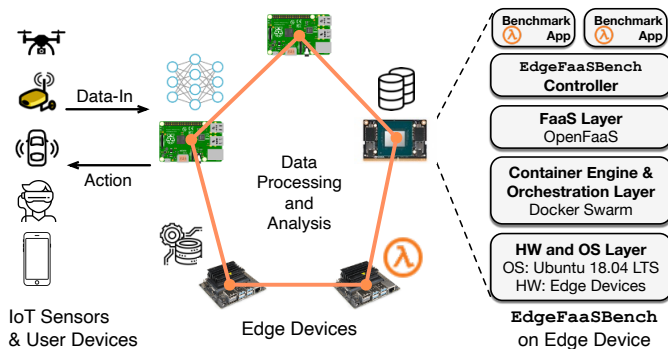


Fig. 2. The overview of EdgeFaaS Bench on edge devices

ufacturers recently released high-performance edge devices, often equipped with GPUs [52], [58], [59], and Edge TPU accelerators [60]. Thanks to this development, understanding the performance and behaviors of edge applications running on such devices is particularly challenging due to the variety of resource models and their heterogeneity, and it has become a significant research task. Moreover, the development of edge devices opens up a new opportunity to leverage application deployment models recently developed by the cloud computing community; serverless computing and FaaS models [15], [16]. Therefore, benchmarking with the FaaS applications will provide essential indications to develop more flexible and lightweight applications models, which are more suitable for edge computing [27]–[32]. It is worth noting that edge devices can be deployed in both middle (edge server) and bottom (IoT sensor and device) layers. Generic and low-end edge devices like Raspberry Pi can be used in the IoT sensor and device layer, and high-end edge devices with GPU resources are typically deployed in the edge server layer.

B. Overview of EdgeFaaS Bench

Fig. 2 illustrates the overview of EdgeFaaS Bench on edge devices. EdgeFaaS Bench on edge devices is composed of four different hierarchical components.

HW and OS Layer. This layer refers to the edge devices and OS running on the devices. While there are some differences with devices, edge devices like Raspberry Pi 4B [51] and Nvidia’s Jetson series [52], [58], [59] commonly have multi-core ARM CPU designed based on `arm64/aarch64` architecture, 4GB to 8GB of RAM, and storage with portable micro-SD card. Moreover, some edge devices, designed for facilitating AI inference tasks at the edge, are equipped with GPU accelerators. For example, Jetson Nano [52] has a 128-core Nvidia Maxwell GPU accelerator, and Jetson Xavier NX [59] is equipped with a Volta GPU with 384 CUDA cores and 48 Tensor cores. We also selected Ubuntu 18.04 LTS (64-bit) for the OS of the edge devices because this version of Ubuntu is commonly running on most edge devices.

Container Engine and Orchestration Layer. Docker [61] and Docker Swarm [50] are used for EdgeFaaS Bench. Docker is the most widely used container engine, and Docker Swarm

is the container orchestration framework for Docker. Both are used to support OpenFaaS (in the FaaS layer) and host serverless applications managed by OpenFaaS. Among the multiple capabilities Docker Swarm supports, the auto-scaling mechanism is particularly beneficial for benchmarking edge devices. For example, EdgeFaaS Bench can leverage Docker Swarm’s auto-scaling to characterize the performance with concurrent executions of serverless applications on the devices.

FaaS Layer. This layer uses OpenFaaS [49] as a primary serverless framework to manage various serverless benchmark workloads. OpenFaaS is selected for EdgeFaaS Bench for the following reasons. First, OpenFaaS offers convenient and comprehensive capabilities of managing serverless applications with minimal HW requirements, various programming languages to write serverless applications, and packaging the applications in a containerized manner. Moreover, the programmable APIs and CLI of OpenFaaS allow convenient scaling-out and scaling-in operations for serverless functions on edge devices. EdgeFaaS Bench uses `faas-cli`¹ with version 0.13.13 and OpenFaaS with version 0.20.5.

EdgeFaaS Bench Controller Layer and Benchmark Workloads. These two layers comprise the core of EdgeFaaS Bench. The controller layer is responsible for running the benchmark tasks and measuring system statistics (e.g., resource utilization) on edge devices. To start and stop a benchmark workload, EdgeFaaS Bench collaborates with APIs and CLI functionality of OpenFaaS. Moreover, EdgeFaaS Bench uses diverse system monitoring techniques, including `sysstat`² and `docker stats`³, to monitor the changes in resource utilization and application performance. The detailed information on collected metrics and procedures from EdgeFaaS Bench will be described in the following subsection (Section II-C).

EdgeFaaS Bench offers both micro-benchmark and application-level benchmark capabilities to capture diverse aspects of edge devices. All the benchmark workloads used in EdgeFaaS Bench will be discussed in Section III.

C. Benchmark Metrics and Procedure of EdgeFaaS Bench

As illustrated in Fig. 3, EdgeFaaS Bench is composed of 1) *benchmark client* and 2) *device manager*. The *benchmark client* runs on a client machine that triggers the benchmark process, analyzes performance statistics collected by both devices and the client machine, and generates the final report for the benchmark. EdgeFaaS Bench uses a workload generator (e.g., Apache bench⁴ to send requests to serverless benchmark workloads on an edge device and measure the performance statistics (e.g., application’s response time) from the client-side. The *device manager* controls the benchmark process on the device-side. The main component in the device manager is the system monitor that captures performance statistics

¹<https://github.com/openfaas/faas-cli>

²<http://sebastien.godard.pagesperso-orange.fr/>

³<https://docs.docker.com/engine/reference/commandline/stats/>

⁴<https://httpd.apache.org/docs/2.4/programs/ab.html>

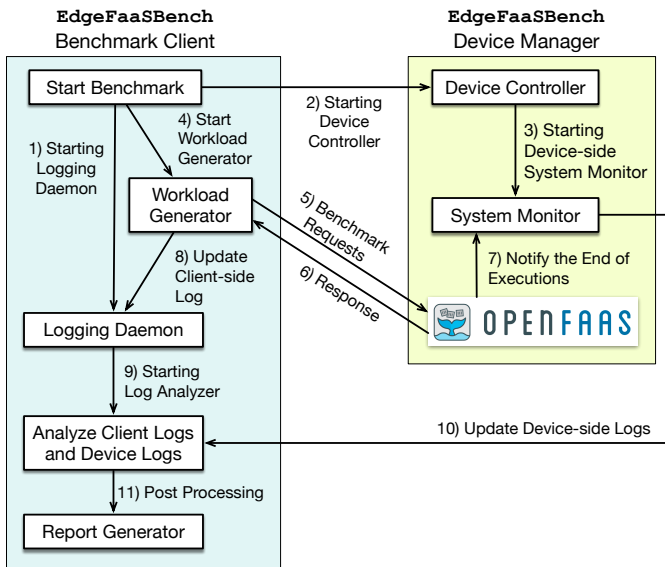


Fig. 3. Benchmark Procedure of EdgeFaaS Bench

during the benchmark execution and reports the collected metrics/statistics to the benchmark client.

Benchmark Metrics. EdgeFaaS Bench can measure three different categories of metrics for the performance benchmark.

1) *System Resource Utilization:* Four main resource utilization metrics, e.g., CPU, memory, disk I/O, and network bandwidth, can be measured by EdgeFaaS Bench. To accurately collect CPU and memory usage on edge devices, a particular challenge for EdgeFaaS Bench is to differentiate the resource consumption by FaaS containers (hosting serverless benchmark workloads) managed by OpenFaaS and that by other processes (e.g., EdgeFaaS Bench system monitor on the device). Therefore, EdgeFaaS Bench employs two different system monitoring techniques, `docker stats` and `sysstat`, to collectively measure separate resource consumption by FaaS containers and other processes. In particular, `docker stats` allows EdgeFaaS Bench to measure resource utilization per each FaaS container. Other system metrics, e.g., disk I/O and network bandwidth, are measured using micro-benchmark serverless workloads developed based on `dd` and `iPerf3`⁵.

2) *Serverless Application Performance:* The response times of the benchmark workloads are the main performance metric to understand the behavior of serverless applications. The response times are measured by both on benchmark client and device manager (edge devices) of EdgeFaaS Bench. The benchmark client measures client-side response times. i.e., the actual execution time on the devices and the transmission latency. Furthermore, the device manager can collect the execution time of serverless applications inside the OpenFaaS docker container. EdgeFaaS Bench can provide multiple types of statistics of the response time, including the average, tail (e.g., 99%ile), and distribution of the response time.

3) *Serverless-related Metrics:* To measure the performance and behavior of serverless applications on edge devices, EdgeFaaS Bench can collect *cold* and *warm start times* of serverless applications. The cold start time represents the delay when the first function (serverless application) is invoked, and it includes both container creation and application deployment overhead. The cold start time is an initialization delay caused when the first function (serverless application) is invoked, and it includes both container creation and application deployment overhead. On the other hand, the warm start time is the function invocation delay when a request arrives in an existing serverless container. In particular, the cold start time is the primary performance bottleneck of serverless applications because the cold start time can be up to an order of magnitude slower than the warm start time [62], [63]. Therefore, it is critical to measure such overhead on edge devices. To measure the cold start time, EdgeFaaS Bench leverages OpenFaaS CLI and customized `yml` files for benchmark workloads to intentionally create cold start cases (e.g., set `com.openfaas.scale.min`⁶ to zero). Moreover, the cold start time can be further amplified with concurrent function executions. To measure such impact, EdgeFaaS Bench dynamically changes the request rates in the workload generator and auto-scaling configuration in OpenFaaS to support more function executions.

Benchmark Procedure. Fig. 3 also shows the benchmark procedure of EdgeFaaS Bench. EdgeFaaS Bench begins by starting the logging daemon in the benchmark client (step #1). The logging daemon is responsible for collecting all the logs from both the benchmark client and system monitor on the target edge device. The benchmark client then notifies the device controller on the edge device about the start of the benchmark (step #2). The device controller initializes the system monitor (step #3), and the system monitor becomes ready to collect system statistics of the edge device (e.g., resource utilization).

The workload generator in the client is started (step #4), and then it sends benchmark requests to the serverless benchmark workloads in the OpenFaaS framework on the device (step #5). When the benchmark workload receives the requests from the workload generator, it starts conducting the performance benchmark on the device. At the same time, the system monitor starts data collection of the edge device's system statistics (e.g., the change of CPU usage). Once the benchmark workload processes the request from the workload generator, the application sends a response back to the workload generator (step #6). The workload generator measures the response time by calculating the difference between the time to send the request and the time to receive the response. When the benchmark workload finishes its execution, it notifies the end of the benchmark to the system monitor (step #7).

Then, the logging daemon in the benchmark client collects the client-side logs and measured statistics, including the application's response time (step #8), and starts the log analyzer

⁵<https://iperf.fr/iperf-download.php>

⁶<https://docs.openfaas.com/architecture/autoscaling/>

TABLE I
SUMMARY OF BENCHMARK WORKLOADS IN EDGEFAASBENCH. (H: HIGH, M: MEDIUM, L: LOW)

Category	Workload Name	Workload Type					Description
		CPU	MEM	I/O	NET	GPU	
Micro-Benchmark	Matrix Multiplication (MM)	H	H	-	-	-	Performing matrix multiplication of different matrix sizes multiple times.
	Fast Fourier Transform (FFT)	H	H	-	-	-	Reading a random seed number and performing its Fast Fourier Transform operations multiple times.
	Floating Point Operation Sine (FPO-SINE)	H	M	-	-	-	Calculating the sine value of all 360 degrees multiple times.
	Floating Point Operation Square Root (FPO-SQRT)	H	M	-	-	-	Calculating the square root value of random numbers, ranging from 10,000 to 30,000 multiple times
	Sorter (SORT)	M	H	L	-	-	Reading a file containing random text data and sorting the text data using the Linux <i>sort</i> command.
	dd (DD)	L	L	H	-	-	Performing random read/write operations on storage (micro-SD) on edge devices using the Linux <i>dd</i> command.
	iPerf3 (IPERF)	L	L	-	H	-	Leveraging the iPerf3 tool to measure the achievable bandwidth of the IP network of edge devices.
Application-Level Benchmark	Image Processing (IP)	M	M	L	L	-	Resizing random images (from benchmark client) to a size of 400×400 pixels.
	Sentiment Analysis (SA)	H	M	-	L	-	Downloading JSON files about different topics and calculating the ratio of positive and negative engagements about the topics.
	Speech to Text (ST)	M	M	L	M	-	Downloading a random audio file from external storage, generating translated text, and sending it back to the benchmark client.
	Image Classification on CPU (IC-CPU)	H	M-H	L	L	-	Receiving random images from the benchmark client, performing image classification tasks using pre-training CNN models on CPU, and transferring the classification results to the benchmark client.
	Image Classification on GPU (IC-GPU)	M	M-H	L	L	H	Performing a similar task with IC-CPU. But this benchmark enables GPU resources to perform the DNN inference task faster. It can be run on edge devices with GPU. i.e., Nvidia Jetson series
	Object Detection on CPU (OD-CPU)	H	H	L	L	-	Receiving random images from the benchmark client, and performing object detection tasks with YOLOv3 on CPU.
	Object Detection on GPU (OD-GPU)	M	H	L	L	H	Performing the similar task with OD-CPU, but enabling GPU for faster inference. This workload can run on GPU-equipped edge devices.

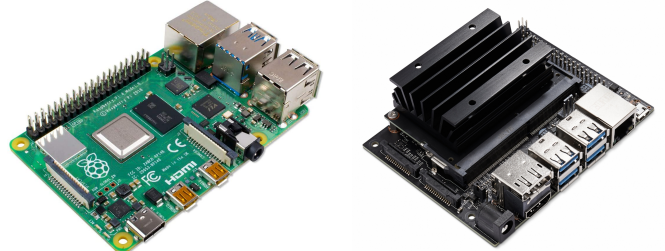
(step #9), which collects the device-side benchmark logs and statistics (step #10) and conducts further analysis of both client and device logs to understand the performance of the serverless benchmark workloads on the device. The results from the log analyzer will be given to the report generator to create the final performance benchmark reports for the user (step #10).

III. BENCHMARK WORKLOADS IN EDGEFAASBENCH

This section describes serverless benchmark workloads in EdgeFaaS Bench. The benchmark workloads can be categorized into two types; 1) micro-benchmark and 2) application-level workloads. The summary of benchmark workloads in EdgeFaaS Bench is shown in Table I.

Micro-benchmark. Micro-benchmark workloads are employed to benchmark the performance of specific resources on edge devices. i.e., CPU, memory. MM and FFT generate high workloads for both CPU and memory resources on edge devices by performing mathematical operations. MM uses three different matrix sizes (300×300 , 400×400 , and 450×450) and performs the matrix multiplication operations 300 times. FFT calculates a Fast Fourier Transform of a random seed number. Both FPO-SINE, and FPO-SQRT are workloads to generate high CPU stress and moderate memory stress. FPO-SINE calculates the sine value of all 360 degrees, and FPO-SQRT calculates square root values of random numbers (ranging from 10,000 to 30,000) multiple times to generate stress to CPU.

Moreover, the micro-benchmark workloads incorporate off-the-shelf Linux commands and tools. SORT tests the memory



(a) Raspberry Pi 4B (b) Nvidia Jetson Nano
Fig. 4. Two edge devices, Jetson Nano and Raspberry PI 4B, used for evaluation with EdgeFaaS Bench

performance on edge devices by generating high memory pressure from the sort operation of random text data with *sort* command. DD uses *dd* Linux command to benchmark Disk I/O performance of edge devices. IPERF uses the iPerf3 tool to measure the achievable network bandwidth of edge devices.

For the micro-benchmark workloads, `python3-debian` template in OpenFaaS is commonly used to implement the workloads as a form of serverless applications on edge devices.

Application-level Benchmark. Application-level benchmark workloads employ realistic serverless application models to benchmark the performance of edge devices. IP is an image processing serverless application that receives random images from EdgeFaaS Bench benchmark client and resizes them to 400×400 pixel size images. IP is implemented using the

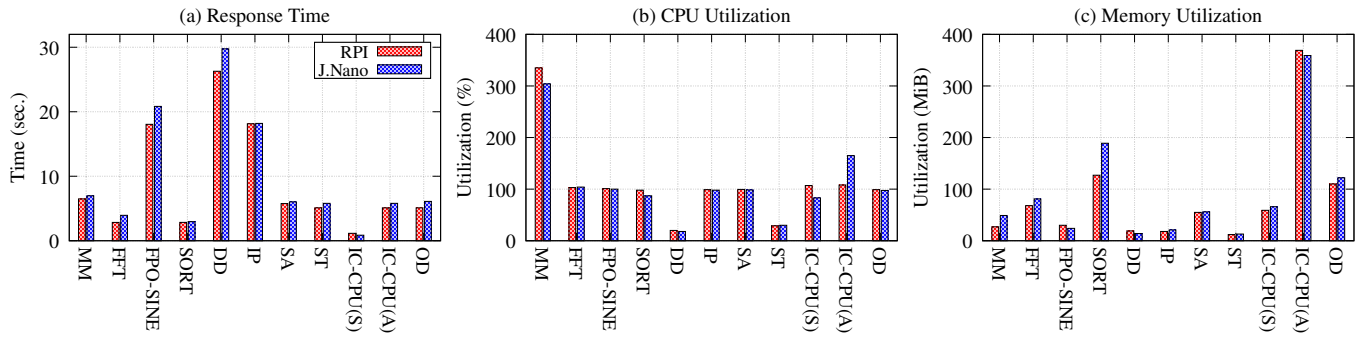


Fig. 5. Benchmark results of response time, CPU utilization, and memory utilization. Please note that IC-CPU (S) is the image classification workload with SqueezeNet, and IC-CPU (A) is the image classification workload with AlexNet.

`python-pillow`⁷ library, which provides extensive image file format support, an efficient internal representation, and powerful image processing capabilities. SA is a sentiment analysis application that detects the ratio of positive and negative engagements about multiple random topics provided by benchmark client. To implement SA, NLKT⁸ and TextBlob⁹ libraries for Python3 are used. ST is an application that generates text data by transcribing the random audio files provided from EdgeFaaS Bench benchmark client and sends it back to the client. Two Python3 libraries, `pyttsx3`¹⁰ and `SpeechRecognition`¹¹, are used to implement ST.

Moreover, since the AI at the edge is increasingly adopted in edge computing, EdgeFaaS Bench also employs four serverless benchmark workloads that perform DL inference tasks on edge devices. Both IC-CPU and IC-GPU perform image classification tasks on edge devices using two pre-trained DL models; AlexNet [64] and SqueezeNet [65]. The difference between IC-CPU and IC-GPU is as follows. IC-CPU uses the CPU of edge devices for performing image classification tasks to benchmark edge devices without GPU accelerator like Raspberry Pi. On the other hand, IC-GPU can perform the image classification tasks on GPU, which offers faster inference time. In particular, IC-GPU is designed to benchmark edge devices with GPU accelerators. i.e., Nvidia’s Jetson series [52], [58], [59]. Also, both workloads generate slightly different memory pressure on edge devices with various models. For example, SqueezeNet is a lightweight DL model, and image classification with SqueezeNet shows moderate memory utilization. On the other hand, the image classification tasks with AlexNet, which is heavier than SqueezeNet, can generate high memory pressure. For both IC-CPU and IC-GPU, we use PyTorch¹² for the main DL framework. In addition, both OD-CPU and OD-GPU generate object detection workloads on edge devices. Similar to the previous workloads, OD-GPU is to benchmark GPU performance on edge devices. The

object detection workloads use an open-source darknet¹³ framework, which is written in C and CUDA, and it employs YOLOv3 [66] for performing the object detection tasks.

All seven application-level workloads are also implemented based on a `python3-debian` template to be run as serverless applications on OpenFaaS in edge devices.

IV. BENCHMARK RESULTS

This section describes our initial benchmark results measured by EdgeFaaS Bench to show its effectiveness.

A. Benchmark Edge Devices

This benchmark study uses two widely used edge devices shown in Fig. 4. Raspberry Pi 4B (RPI, shown in Fig. 4a) [51] is a small, low-cost, representative computing board for edge and IoT devices. RPI uses Broadcom BCM2711 SoC and has a quad-core ARM Cortex-A72 (1.5 GHz) and 4 GB LPDDR4 RAM. RPI can be deployed for various use cases, such as IoT sensor control, sending data collection and filtering, and lightweight data processing. RPI devices are typically deployed in the bottom (IoT sensor and device) layer in Fig. 1 in Section II.

Jetson Nano (J.Nano, shown in Fig. 4b) has a slightly older version of ARM cores (a four-core Cortex-A57 at 1.5 GHz) and 4 GB LPDDR4 RAM. However, J.Nano is equipped with a 128-core Nvidia Maxwell GPU and can provide faster DL inference time with various DL frameworks like PyTorch and TensorFlow [10]. Because J.Nano is specialized in AI processing, the device can be deployed in both the middle (edge servers) and the bottom (IoT sensor and device) layers in Fig. 1 in Section II.

B. Benchmark Results

We report four benchmark results with two edge devices, including response time and resource utilization, cold and warm start times, concurrent function executions, and the CPU and GPU performance comparison.

Response Time and Resource Utilization. Fig. 5 shows EdgeFaaS Bench’s benchmark results on response time and

⁷<https://pillow.readthedocs.io/en/stable/>

⁸<https://www.nltk.org/index.html>

⁹<https://textblob.readthedocs.io/en/dev/>

¹⁰<https://pypi.org/project/pyttsx3/>

¹¹<https://pypi.org/project/SpeechRecognition/>

¹²<https://pytorch.org/>

¹³<https://pjreddie.com/darknet/>

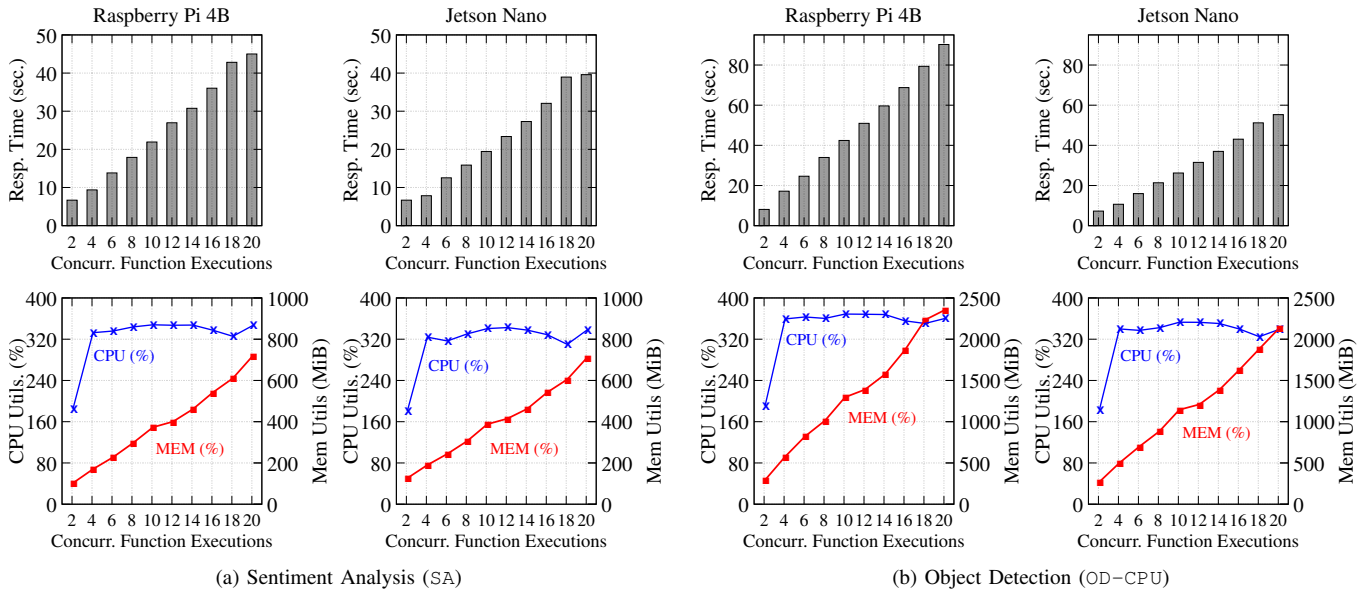


Fig. 6. Benchmark results of concurrent function executions for two EdgeFaaS Bench workloads. (a) SA workload with concurrent executions, (b) OD-CPU workload with concurrent executions

TABLE II
BENCHMARK RESULTS OF THE COLD AND WARM START TIME OF WORKLOADS ON TWO DEVICES. (SQN: SQUEEZE NET, ALN: ALEX NET)

	Raspberry Pi 4B		Jetson Nano	
	Cold Start	Warm Start	Cold Start	Warm Start
MM	7.8s	1.4s	7.3s	1.3s
FFT	9.2s	1.8s	8.0s	1.4s
FPO-SINE	7.6s	0.2s	5.6s	0.2s
SORT	7.9s	0.3s	5.7s	0.3s
DD	7.4s	0.4s	5.8s	0.4s
IP	8.5s	0.8s	8.0s	0.9s
SA	11.5s	2.1s	11.9s	2.3s
ST	7.6s	0.3s	6.4	0.3s
IC-CPU (SQN)	10.3s	1.9s	10.6s	2.3s
IC-CPU (ALN)	30.0s	5.3s	42.1s	7.7s
OD-CPU	4.0s	0.1s	3.2s	0.1s

CPU/memory utilization. In this evaluation, we ran each benchmark workload 100 times to get reliable benchmark results. The results include all benchmark workloads except for GPU-enabled workloads (IC-GPU, OD-GPU), and IPERF.

As shown in the figure, while both devices showed similar response times for CPU-based workloads, RPI provided about 10% faster response time (except for IC-CPU with SqueezeNet) than J.Nano. This is mainly because RPI has a slightly faster CPU with a newer version of ARM cores. For the CPU and memory utilization, EdgeFaaS Bench reported similar statistics for most of the workloads.

Cold and Warm Start Time. The next benchmark measures serverless functions' cold and warm start times on edge devices. The capability to measure the cold and warm start times is particularly important since the cold start overhead is recognized as the major performance overhead of serverless

applications. To measure the cold start time, EdgeFaaS Bench recorded the initial function invocation time collected by OpenFaaS when no serverless container was running. After this recording, EdgeFaaS Bench compared the time with the first recorded time internally measured in the function in the benchmark workloads. For measuring the warm start time, EdgeFaaS Bench also recorded the request invocation time in OpenFaaS and compared it with the first recorded time in a warm serverless container.

Table II shows the cold and warm start times on both devices measured by EdgeFaaS Bench. While the cold and warm start times vary with different benchmark workloads, both devices showed an order of magnitude slower cold start times compared to the warm start times. For example, RPI and J.Nano had 16.9 \times and 13.6 \times slower cold start times, respectively. Such slower cold start times are consistent with existing measurement studies on serverless platforms on clouds [62]. As identified by prior work [63], the significantly slower cold start times are mostly because of container and package initialization used in the serverless functions. In particular, benchmark workloads like IC-CPU with AlexNet and SqueezeNet and SA require various packages in the serverless functions, so that they needed more time to initialize the serverless container and showed higher cold start times.

For comparing start times between two devices, J.Nano showed equivalent and slightly faster cold start times except for two IC-CPU workloads and SA, and both devices showed similar warm start times.

Concurrent Function Executions. Next, we measured the impact of concurrent invocation of serverless functions on edge devices. In this evaluation, EdgeFaaS Bench generated the concurrent function invocations, gradually increasing from

TABLE III
DEEP LEARNING INFERENCE THROUGHPUT COMPARISON BETWEEN CPU (IC-CPU) AND GPU (IC-GPU) ON JETSON NANO (J.NANO).

	DL Inference Throughput (# Infer./sec.)	
	with SqueezeNet	with AlexNet
Jetson Nano CPU (IC-CPU)	3.06	2.78
Jetson Nano GPU (IC-GPU)	27.65	8.80

1 to 20, and measured response time and resource utilization.

Fig. 6 shows the benchmark results of two workloads, which are SA and OD-CPU. We omit the results of other benchmark workloads due to the page limitation, and the results from other workloads showed a similar pattern to the two workloads reported in the figure. As expected, the response times of concurrent functions increase with a higher level of concurrency. Also, resource utilization (CPU and memory) increases with more concurrent functions. An interesting observation is that CPU utilization was close to the maximum with a lower level of concurrency (e.g., 4 or 6 concurrent functions) on both devices. However, both devices were still able to process more functions while they showed such high CPU utilization. This is mainly because the memory usage did not reach the maximum. The devices were able to handle more concurrent functions unless the memory resources were not saturated. Based on this observation, we expect that the maximum degree of concurrent functions that can be processed by the edge devices will be determined by memory utilization, indicating that the upper bound of concurrent function executions can be determined when memory utilization becomes 100%.

Moreover, between the devices, J.Nano showed better performance to handle concurrent function executions. Interestingly, while the response time to handle a single function on J.Nano was slightly slower than RPI, J.Nano showed a 15% faster response time for SA and a 55% faster response time for OD-CPU with concurrent functions. For OD-CPU, we clearly observed that J.Nano showed 15% – 20% more efficient memory utilization with concurrency.

Comparison of CPU and GPU Performance. The last benchmark result is to show EdgeFaaS Bench’s capability to compare the DL inference performance on CPU and GPU. We performed this comparison on J.Nano executing IC-CPU and IC-GPU, by leveraging its CPU and GPU resources. In this evaluation, we report the end-to-end DL processing latency as well as DL throughput on CPU and GPU by performing multiple executions of 200 batches of images.

Fig. 7 shows the end-to-end DL processing latency of IC-CPU and IC-GPU on J.Nano. The end-to-end latency is the sum of DL loading inference (DL framework and model) and DL inference latency. IC-GPU (running on J.Nano’s GPU) showed 3.2× and 1.2× faster processing latency with SqueezeNet and AlexNet compared to IC-CPU (only running on CPU), showing the benefit of leveraging GPU accelerator for DL inference tasks. Interestingly, the loading latency (blue

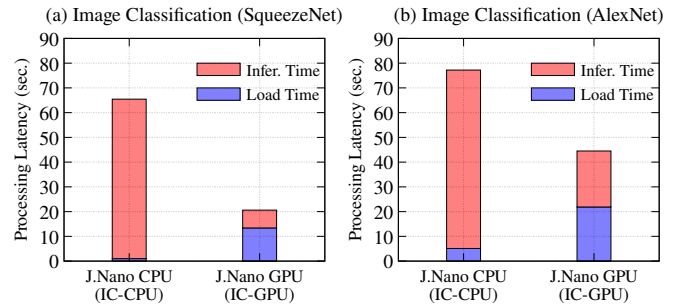


Fig. 7. End-to-end deep learning processing latency comparison between CPU and GPU on Jetson Nano (J. Nano). DL processing latency is composed of load and inference latency.

portion of Fig. 7) of IC-GPU was significantly slower than that of IC-CPU. For example, IC-GPU’s loading latency was over 100× slower than the loading latency of IC-CPU when using SqueezeNet. This is because the GPU accelerator on J.Nano was much slower to initialize and load DL models. Therefore, Pytorch’s initialization latency with GPU is much slower than that with CPU. However, once the DL model is loaded and the framework is initialized, the inference latency with IC-GPU is much faster than the inference latency of IC-CPU. The inference latency of IC-GPU was 7.1× (with SqueezeNet) and 1.8× (with AlexNet) shorter than those of IC-CPU. We also observed that the loading and inference latency could be varying with different DL models. In particular, the latency was significantly changed with DL model sizes.

We also confirm another benefit of enabling GPU on edge devices by reporting the throughput differences between IC-GPU and IC-CPU. As shown in Table III, IC-GPU processed 9× (with SqueezeNet) and 3.2× (with AlexNet) more images compared to IC-CPU, clearly indicating the benefit of GPU accelerator for the AI at the edge processing.

V. RELATED WORK

The research community has proposed various benchmark suites for edge computing environments. This section summarizes the state-of-the-art benchmark suites [39]–[44] and discusses our novelty over the state-of-the-arts.

DeFog [41] is a comprehensive fog/edge computing benchmark suite containing six different edge applications. DeFog focuses on benchmarking the performance of edge computing with three different deployment modes; edge-cloud, cloud-only, and edge-only. DeFog performed an experimental study on three deployment modes using two edge devices (e.g., Raspberry Pi 3 Model B and Odroid XU 4) and an Amazon cloud server, and the experiment results showed the performance variations of benchmark applications with different deployment modes. While DeFog has broader benchmark coverage than EdgeFaaS Bench in supporting different edge computing scenarios, EdgeFaaS Bench has the following novelty over DeFog. First, we focus on benchmarking the performance of edge devices and their diverse resource types (e.g., CPU and GPU). And, differing from DeFog, all the benchmark

TABLE IV
COMPARISON OF EDGE COMPUTING BENCHMARK SUITES

Benchmark	Edge Only	Edge + Cloud	Micro-Benchmarks (# of Apps.)	Application Benchmarks (# of Apps.)	Serverless Support on Edge Device	Serverless Metrics	GPU Support	Open Source
DeFog [41]	Yes	Yes	No	Yes (6)	No	No	-	Yes
Edge AIBench [43]	Yes	No	No	Yes (6)	No	No	Yes	No
pCAMP [54]	Yes	No	No	Yes (4)	No	No	Yes	No
Subedi et al. [10]	Yes	No	No	Yes (4)	No	No	Yes	No
EdgeBench'18 [40]	Yes	Yes	No	Yes (3)	No	No	No	Yes
EdgeBench'20 [42]	Yes	Yes	No	Yes (2)	Yes	No	No	No
Carpio et al. [44]	Yes	No	No	Yes (2)	Yes	No	-	No
EdgeFaaS Bench	Yes	No	Yes (7)	Yes (7)	Yes	Yes	Yes	Yes

workloads in EdgeFaaS Bench are serverless applications, and EdgeFaaS Bench can measure diverse serverless-oriented performance metrics. i.e., cold and warm startup times of serverless applications.

Due to the broader adoption of artificial intelligence (AI) at the edge paradigm [7], benchmark suites that are centered on measuring AI inference performance in edge computing have also been proposed [10], [43], [54]. Edge AIBench [43] is a benchmark suite with four realistic AI deployment scenarios for edge computing and offers capabilities to measure the end-to-end AI performance of various edge-computing scenarios. pCamp [43] conducted benchmarking the performance of machine learning (ML) packages and frameworks when running image classification tasks on edge devices and mobile devices. This work reported latency (including model loading time), memory usage, and energy consumption from different ML packages. Subedi et al. [10] also provided the benchmarking results of AI inference tasks on edge devices by using multiple deep neural networks (DNN) models on various combinations of four edge devices and two AI accelerators for edge devices (e.g., Coral EdgeTPU Accelerators [60]). Moreover, the work performed by Subedi et al. focused on characterizing the multi-tenancy aspects of DNN models' executions on the devices. While AI benchmark workloads are part of our benchmark suite, EdgeFaaS Bench can perform benchmarks with broader FaaS applications models, offering AI-based, traditional applications, and micro-benchmarks. EdgeFaaS Bench also focuses on measuring the feasibility and performance of FaaS offering on edge devices so that it can report various serverless-centric metrics, including cold/warm function startup times and performance variation with concurrent function executions.

EdgeBench'18 [40], EdgeBench'20 [42], and Carpio et al. [44] proposed earlier FaaS adoption to benchmark suites for edge computing. However, these works have limitations in correctly understanding the FaaS offering on edge devices because these benchmarks focused on collaborative cloud-edge environments. For example, EdgeBench'18 [40] relied on FaaS offerings from public cloud providers, such as AWS Greengrass with AWS Lambda. EdgeBench'20 [42] was designed to benchmark the efficacy of diverse cloud-edge workflows that can be used in edge computing deployment. Moreover, the benchmark suite by Carpio et al. [44] contains only two benchmark applications, hence not sufficient to capture various aspects of FaaS behaviors on edge devices.

We summarize the differences between EdgeFaaS Bench and previous benchmark suites in Table IV. As shown in the table, EdgeFaaS Bench employs more comprehensive benchmark workloads ranging from micro-benchmarks to application-level benchmarks (including AI workloads), which help edge users correctly identify the behavior of FaaS characteristics and performance on edge devices. Moreover, by supporting GPU capability on edge devices, EdgeFaaS Bench can evaluate FaaS performance variations on heterogeneous edge devices if GPU resources are available.

VI. CONCLUSION

Due to the fast development of serverless computing and edge devices (especially with AI accelerators), there is increasing demand for serverless edge computing. To evaluate the performance of serverless computing on edge devices, we present EdgeFaaS Bench, a novel benchmark suite for edge devices with serverless computing. EdgeFaaS Bench is prototyped on a modern container engine/orchestration and a widely used open-source serverless framework.

EdgeFaaS Bench comprises 14 serverless workloads to correctly benchmark diverse aspects of edge devices' performance. Among the 14 workloads, 7 of them are micro-benchmark, and the others are application-level benchmark workloads. Micro-benchmark workloads evaluate the performance of a specific resource type (e.g., CPU) on edge devices, and application-level benchmark workloads evaluate the edge devices' performance with realistic application scenarios like DL inference and sentiment analysis. EdgeFaaS Bench can collect diverse tradition and serverless-specific metrics reflecting the performance of edge devices. In particular, EdgeFaaS Bench can collect cold and warm start times of serverless applications and performance degradation with concurrent function executions, which are the important performance bottleneck of serverless applications on edge devices. In addition, EdgeFaaS Bench also enables GPU to benchmark the scenarios of AI at edge computing.

We performed a benchmark study on two widely used edge devices to show the effectiveness of EdgeFaaS Bench and reported diverse performance metrics and serverless characteristics on the edge devices. Finally, EdgeFaaS Bench is publicly available at <https://github.com/kaustubhrajput46/EdgeFaaS Bench>.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their insightful comments and constructive suggestions. This research was in part supported by the U.S. Department of Agriculture (USDA), under award number 2021-67019-34342, and the Academy of Finland (AoF), under grants 317432 and 318937. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of the USDA or AoF.

REFERENCES

- [1] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.
- [2] Rajiv Ranjan and et al. The Next Grand Challenges: Integrating the Internet of Things and Data Science. *IEEE Cloud Computing*, 5(3):12–26, 2018.
- [3] Danny Yuxing Huang, Noah J. Aporthe, Frank Li, Gunes Acar, and Nick Feamster. IoT Inspector: Crowdsourcing Labeled Network Traffic from Smart Home Devices at Scale. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, 4(2):46:1–46:21, 2020.
- [4] Mahadev Satyanarayanan. The Emergence of Edge Computing. *IEEE Computer*, 50(1):30–39, 2017.
- [5] Marcos Dias de Assunção, Alexandre Da Silva Veith, and Rajkumar Buyya. Distributed Data Stream Processing and Edge Computing: A Survey on Resource Elasticity and Future Directions. *Journal of Network and Computer Applications*, 103:1–17, 2018.
- [6] Andy Rosales Elias, Nevena Golubovic, Chandra Krintz, and Rich Wolski. Where’s The Bear?: Automating Wildlife Image Processing Using IoT and Edge Cloud Systems. In *ACM/IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI)*, Pittsburgh, PA, USA, April, 2017. ACM.
- [7] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing. *Proceedings of IEEE*, 107(8):1738–1762, 2019.
- [8] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge Computing for Autonomous Driving: Opportunities and Challenges. *Proceedings of IEEE*, 107(8):1697–1716, 2019.
- [9] Jianwei Hao, Piyush Subedi, In Kee Kim, and Lakshmi Ramaswamy. Characterizing Resource Heterogeneity in Edge Devices for Deep Learning Inferences. In *International Workshop on Systems and Network Telemetry and Analytics (SNTA@HPDC)*, Virtual Event, June, 2021.
- [10] Piyush Subedi, Jianwei Hao, In Kee Kim, and Lakshmi Ramaswamy. AI Multi-Tenancy on Edge: Concurrent Deep Learning Model Executions and Dynamic Model Placements on Edge Device. In *IEEE International Conf. on Cloud Computing (CLOUD)*, September, 2021.
- [11] AWS Lambda. <https://aws.amazon.com/lambda/>. Accessed: 2022-03-01.
- [12] Azure Functions. <https://azure.microsoft.com/en-us/services/functions>. Accessed: 2022-03-01.
- [13] Google Cloud Functions. <https://cloud.google.com/functions>. Accessed: 2022-03-01.
- [14] Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht. Occupy the Cloud: Distributed Computing for the 99%. In *ACM Symposium on Cloud Computing (SoCC)*, Santa Clara, CA, USA, September, 2017.
- [15] Paul C. Castro, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. The rise of serverless computing. *Communications of ACM*, 62(12):44–54, 2019.
- [16] Johann Schleier-Smith and et al. What serverless computing is and should become: the next phase of cloud computing. *Communications of ACM*, 64(5):76–84, 2021.
- [17] Fei Xu, Yiling Qin, Li Chen, Zhi Zhou, and Fangming Liu. Adnn: Achieving predictable distributed DNN training with serverless architectures. *IEEE Transactions on Computers*, 71(2):450–463, 2022.
- [18] Lixiang Ao, Liz Izhikevich, Geoffrey M. Voelker, and George Porter. Sprocket: A Serverless Video Processing Framework. In *ACM Symposium on Cloud Computing (SoCC)*, Carlsbad, CA, USA, October, 2018.
- [19] Ashraf Mahgoub, Karthick Shankar, Subrata Mitra, Ana Klimovic, Somali Chaterji, and Saurabh Bagchi. SONIC: Application-aware Data Passing for Chained Serverless Applications. In *USENIX Annual Technical Conference (ATC)*, Virtual Event, July, 2021.
- [20] Anirban Bhattacharjee, Yogesh D. Barve, Shweta Khare, Shunxing Bao, Aniruddha Gokhale, and Thomas Damiano. Stratum: A Serverless Framework for the Lifecycle Management of Machine Learning-based Data Analytics Tasks. In *USENIX Conference on Operational Machine Learning (OpML)*, Santa Clara, CA, USA, May, 2019.
- [21] Ingo Müller, Renato Marroquin, and Gustavo Alonso. Lambda: Interactive Data Analytics on Cold Data Using Serverless Cloud Infrastructure. In *International Conf. on Management of Data (SIGMOD)*, Portland, OR, USA, June, 2020.
- [22] Djob Mvondo, Mathieu Bacou, Kevin Nguetchouang, Lucien Ngale, Stéphane Pouget, Josiane Kouam, Renaud Lachaize, Jinho Hwang, Tim Wood, Daniel Hagimont, Noël De Palma, Bernabé Batchakui, and Alain Tchana. OFC: an opportunistic caching system for FaaS platforms. In *Sixteenth European Conference on Computer Systems (EuroSys)*, Virtual Event, April, 2021.
- [23] Stefan Nastic, Thomas Rausch, Ognjen Seckic, Schahram Dustdar, Marjan Gusev, Bojana Koteska, Magdalena Kostoska, Boro Jakimovski, Sasko Ristov, and Radu Prodan. A Serverless Real-Time Data Analytics Platform for Edge Computing. *IEEE Internet Computing*, 21(4):64–71, 2017.
- [24] Luciano Baresi and Danilo Filgueira Mendonça. Towards a Serverless Platform for Edge Computing. In *IEEE International Conference on Fog Computing (ICFC)*, Czech Republic, June, 2019.
- [25] Gareth George, Fatih Bakir, Rich Wolski, and Chandra Krintz. NanoLambda: Implementing Functions as a Service at All Resource Scales for the Internet of Things. In *IEEE/ACM Symposium on Edge Computing (SEC)*, San Jose, CA, USA, November, 2020.
- [26] Bin Wang, Ahmed Ali-Eldin, and Prashant J. Shenoy. LaSS: Running Latency Sensitive Serverless Computations at the Edge. In *International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, Virtual Event, June, 2021.
- [27] Nabil El Ioini, David Hästbacka, Claus Pahl, and Davide Taibi. Platforms for Serverless at the Edge: A Review. In *European Conference on Service-Oriented and Cloud Computing (ESOCC)*, Heraklion, Crete, Greece, September, 2021.
- [28] Zhe Zhou, Bingzhe Wu, Zheng Liang, Guangyu Sun, Chenren Xu, and Guojie Luo. Is FaaS Suitable for Edge Computing? In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, Virtual Event, June, 2020.
- [29] Mohammad Sadegh Aslanpour, Adel Nadjaran Toosi, Claudio Cicconetti, Bahman Javadi, Peter Sbarski, Davide Taibi, Marcos Dias de Assunção, Sukhpal Singh Gill, Raj Gaire, and Schahram Dustdar. Serverless Edge Computing: Vision and Challenges. In *2021 Australasian Computer Science Week Multiconference (ACSW)*, Dunedin, New Zealand, February, 2021.
- [30] Phani Kishore Gadepalli, Gregor Peach, Ludmila Cherkasova, Rob Aitken, and Gabriel Parmer. Challenges and Opportunities for Efficient Serverless Computing at the Edge. In *38th International Symposium on Reliable Distributed Systems (SRDS)*, Lyon, France, October, 2019.
- [31] Luciano Baresi and Danilo Filgueira Mendonça. Towards a Serverless Platform for Edge Computing. In *IEEE International Conference on Fog Computing (ICFC)*, Prague, Czech Republic, June, 2019.
- [32] Thomas Rausch, Waldemar Hummer, Vinod Muthusamy, Alexander Rashed, and Schahram Dustdar. Towards a Serverless Platform for Edge AI. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, Renton, WA, USA, November, 2019.
- [33] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1990.
- [34] Lizy Kurian John and Lieven Eeckhout. *Performance Evaluation and Benchmarking*. Taylor & Francis, 2005.
- [35] Jeongchul Kim and Kyungyong Lee. FunctionBench: A Suite of Workloads for Serverless Cloud Function Service. In *IEEE International Conference on Cloud Computing (CLOUD)*, Milan, Italy, July, 2019.
- [36] Pascal Maissen, Pascal Felber, Peter G. Kropf, and Valerio Schiavoni. FaaSdom: A Benchmark Suite for Serverless Computing. In *ACM International Conference on Distributed and Event-based Systems (DEBS)*, Montreal, Quebec, Canada, July, 2020.

- [37] Marcin Copik, Grzegorz Kwasniewski, Maciej Besta, Michal Podstawski, and Torsten Hoeffler. SeBS: A Serverless Benchmark Suite for Function-as-a-Service Computing. *CoRR*, abs/2012.14132, 2020.
- [38] Tianyi Yu and et al. Characterizing Serverless Platforms with Serverlessbench. In *ACM Symposium on Cloud Computing (SoCC)*, Virtual Event, October, 2020.
- [39] Blesson Varghese, Nan Wang, David Bermbach, Cheol-Ho Hong, Eyal de Lara, Weisong Shi, and Christopher Stewart. A Survey on Edge Performance Benchmarking. *ACM Computing Surveys*, 54(3):66:1–66:33, 2021.
- [40] Anirban Das, Stacy Patterson, and Mike Wittie. Edgebench: Benchmarking Edge Computing Platforms. In *IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Zurich, Switzerland, December, 2018.
- [41] Jonathan McChesney, Nan Wang, Ashish Tanwer, Eyal de Lara, and Blesson Varghese. DeFog: Fog computing benchmarks. In *ACM/IEEE Symposium on Edge Computing (SEC)*, Washington DC, USA, November, 2019.
- [42] Qirui Yang, Runyu Jin, Nabil Gandhi, Xiongzi Ge, Hoda Aghaei Khouzani, and Ming Zhao. Edgebench: A Workflow-based Benchmark for Edge Computing. *arXiv preprint arXiv:2010.14027*, 2020.
- [43] Tianshu Hao and et al. Edge AIBench: Towards Comprehensive End-to-End Edge Computing Benchmarking. In *International Symposium on Benchmarking, Measuring and Optimization (BENCH)*, Seattle, WA, USA, December, 2018.
- [44] Francisco Carpio, Marc Michalke, and Admela Jukan. Engineering and Experimentally Benchmarking a Serverless Edge Computing System. *CoRR*, abs/2105.04995, 2021.
- [45] Chanh Nguyen Le Tan, Cristian Klein, and Erik Elmroth. Multivariate LSTM-Based Location-Aware Workload Prediction for Edge Data Centers. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, Larnaca, Cyprus, May, 2019.
- [46] Bin Cheng, Jonathan Fuerst, Gurkan Solmaz, and Takuya Sanada. Fog Function: Serverless Fog Computing for Data Intensive IoT Services. In *IEEE International Conference on Services Computing (SCC)*, Milan, Italy, July, 2019.
- [47] Mohammad Shahrad, Rodrigo Fonseca, Iñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. In *USENIX Annual Technical Conference (ATC)*, Virtual Event, July 2020.
- [48] Shivani Arbat, Vinodh Kumar Jayakumar, Jaewoo Lee, Wei Wang, and In Kee Kim. Wasserstein Adversarial Transformer for Cloud Workload Prediction. In *The Thirty-Fourth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-22)*, Vancouver, BC, Canada, February, 2022.
- [49] OpenFaaS. <https://www.openfaas.com/>. Accessed: 2022-03-01.
- [50] Swarm mode overview. <https://docs.docker.com/engine/swarm/>. Accessed: 2022-03-01.
- [51] Raspberry Pi 4. <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/>, 2022. Accessed: 2022-03-01.
- [52] Jetson Nano — Nvidia Developer. <https://developer.nvidia.com/embedded/jetson-nano>, 2022. Accessed: 2022-03-01.
- [53] Omid Setayeshfar, Karthika Subramani, Xingzi Yuan, Raunak Dey, Dezhi Hong, Kyu Hyung Lee, and In Kee Kim. ChatterHub: Privacy Invasion via Smart Home Hub. In *IEEE International Conference on Smart Computing (SMARTCOMP)*, Virtual Event, August, 2021.
- [54] Xingzhou Zhang, Yifan Wang, and Weisong Shi. pCamp: Performance Comparison of Machine Learning Packages on the Edges. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*, Boston, MA, USA, July, 2018.
- [55] Pinchao Liu, Dilma Da Silva, and Liting Hu. DART: A Scalable and Adaptive Edge Stream Processing Engine. In *USENIX Annual Technical Conference (ATC)*, Virtual Event, July, 2021.
- [56] Gaurav Somani, Xinghui Zhao, Satish Narayana Srirama, and Rajkumar Buyya. Integration of Cloud, Internet of Things, and Big Data Analytics. *Software: Practice and Experience*, 49(4):561–564, 2019.
- [57] Norman P. Jouppi and et al. In-Datcenter Performance Analysis of a Tensor Processing Unit. In *44th International Symposium on Computer Architecture (ISCA)*, Toronto, ON, Canada, June, 2017.
- [58] Nvidia Jetson TX2. <https://developer.nvidia.com/embedded/jetson-tx2>, 2022. Accessed: 2022-03-01.
- [59] NVIDIA Jetson Xavier NX. <https://developer.nvidia.com/embedded/jetson-xavier-nx>, 2022. Accessed: 2022-03-01.
- [60] Coral USB Accelerator datasheet. <https://coral.ai/docs/accelerator/datasheet/>, 2022. Accessed: 2022-03-01.
- [61] Docker. <https://docs.docker.com/>. Accessed: 2022-03-01.
- [62] Anirban Das, Shigeru Imai, Stacy Patterson, and Mike P Wittie. Performance Optimization for Edge-Cloud Serverless Platforms via Dynamic Task Placement. In *IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, Melbourne, Australia, May, 2020.
- [63] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael M. Swift. Peeking Behind the Curtains of Serverless Platforms. In *USENIX Annual Technical Conference (ATC)*, Boston, MA, USA, July, 2018.
- [64] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Annual Conference on Neural Information Processing Systems (NIPS)*, Lake Tahoe, Nevada, USA, December, 2012.
- [65] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *CoRR*, abs/1602.07360, 2016.
- [66] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *CoRR*, abs/1804.02767, 2018.