Towards Edge Benchmarking: A Methodology for Characterizing Edge Workloads

Klervie Toczé
Linköping University, Sweden
klervie.tocze@liu.se

Norbert Schmitt
University of Würzburg, Germany
norbert.schmitt@uni-wuerzburg.de

Ivona Brandic
Vienna University of Technology, Austria
ivona.brandic@tuwien.ac.at

Atakan Aral
Vienna University of Technology, Austria
atakan.aral@tuwien.ac.at

Simin Nadjm-Tehrani
Linköping University, Sweden
simin.nadjm-tehrani@liu.se

***Abstract*—The edge computing paradigm has recently attracted research efforts coming from different application domains. However, evaluating an edge platform or algorithm is impeded by the lack of suitable benchmarks.**

We propose a methodology for characterizing edge workloads from different application domains. It is a first step towards defining workloads to be included in a future edge benchmarking suite. We evaluate the methodology on three use cases and find that defining a common and standard set of workloads is plausible.

I. Introduction

The edge computing paradigm, where resources are moved closer to the users, is currently subject to significant research effort, e.g. with regards to resource management [1]. However, there is so far no common way for comparing different approaches and solutions due to lack of edge benchmarks, as opposed to the case with cloud evaluations [2].

Recently, Das et al. [3] proposed a suite called EdgeBench for benchmarking edge computing platforms such as Amazon AWS Greengrass or Microsoft Azure IoT Edge. Olguín Muñoz et al. [4] proposed a suite called EdgeDroid for benchmarking augmented reality or wearable cognitive assistance applications. However, those first benchmarks only address a limited part of the need for edge benchmarking, since they focus on platforms or a specific application domain.

In order to provide a benchmark to enable comparison of application-agnostic edge computing *techniques*, we, within the SPEC Edge activity, started working on characterizing edge workloads. Our goal is to investigate whether the numerous edge use cases could be evaluated using a limited set of standard but representative workloads.

This article presents our current progress. We propose a methodology for characterizing edge use cases with regard to relevant characteristics, and present the outcome of applying the methodology on a first set of three use cases.

II. Edge use cases considered

In this section, we briefly present the three edge use cases that will be characterized in Section V.

Klervie Toczé is supported by the Swedish national graduate school in computer science (CUGS).

Augmented/Mixed Reality: In an augmented/mixed reality application (AR/MR), virtual objects are integrated into the real world and the user is able to interact with them. Such an application requires using considerable amounts of sensor data, as well as a lot of computation to render the graphics of the virtual objects. Those requirements both limit the kind of device that can run AR/MR applications, and impact the remaining charge for the device battery. As a consequence, AR/MR applications are good candidates for being moved to the edge, where more powerful devices can take care of the heavy computation. However, the applications should still be responsive enough for maintaining quality of service (QoS).

Vehicular Data Analytics: Next generation transportation systems with electric powered, connected and/or autonomous vehicles generate immense quantities of streaming data. Data analytics on the vehicular setting range from in-vehicle information systems such as range and charge planning, to mission-critical applications such as collision avoidance. Since the conclusions reached through data analytics remain relevant only for a short time, aggregating the data at a centralized server (e.g. cloud) is infeasible and local processing is imperative. Even if the vehicles are equipped with sufficient computing power, many applications depend on effective communication/synchronization with the roadside units, (smart) power grid, and other vehicles. Edge computing can efficiently provide the required computation capacity and connectivity, while meeting real-time requirements.

Code Offloading: This usually means dynamically moving computationally intensive tasks from end devices to edge devices, allowing the network of different devices to be in an overall energy-efficient state. If the state changes due to higher load or new / upgraded devices, a system might enter an energy inefficient state. If the load reaches a limit at which a powerful edge device can be used efficiently, the application can be moved to the edge while the remaining computations are distributed among the local devices to conserve the maximum amount of energy across all devices, including the edge. Offloading must consider the abilities of devices, requirements of the different applications, their possible tradeoffs and resilience to latency to maintain QoS.

TABLE I
CHARACTERIZATION OF THE CONSIDERED USE CASES

Use case	Computation demand	Communication demand	Storage demand	Deadline	Arrival type	Interarrival time
Augmented/Mixed reality	High	High	Low	Close	Periodic	Short
Vehicular Data Analytics	High	High	Low	Close	Aperiodic	Short
Code Offloading	High	High	Low	Close	Periodic	Long

III. CHARACTERISTICS

By analyzing the above use cases, as well as the type of workload used in recent edge computing research (see Section 6 in a recent survey [1]), we isolate four main workload characteristics that are of interest when describing edge workloads. Those characteristics are: resource demand, deadline, arrival type, and interarrival time.

For the resource demand characteristic, we choose to include the three most common resources considered in recent research, i.e. computation, communication and storage [1].

IV. METHODOLOGY

We define a methodology to analyze edge use-cases with regards to a certain characteristic. The methodology is purposefully not based on absolute values, as the demand should be seen relative to the available resources. We consider the weakest part of the system as the reference for our relative characterization to account for edge devices with different capabilities.

For each characteristic we define two categories: high/low for resource demand, close/far, aperiodic/periodic and short/long for the deadline, arrival type and interarrival respectively. A further division into more fine-grained categories is not considered helpful with the current knowledge base.

The proposed characterization works as follows:

Computation Demand: Reasons for a *high* computational demand are complex computations or large datasets, like rendering or training a machine learning algorithm. The computational demand is also seen as *high* if a computation often or with long interrupts stops other tasks from running simultaneously on the same edge device. Hence it will result in a degradation or error in other tasks or itself.

Communication Demand: If the constant or peak communication demand closely or fully saturates the available bandwidth, the demand is set to *high*. The demand is also *high*, if the communication uses a shared medium and is currently impairing other services or is impaired itself by others.

Storage Demand: A use case has a *high* storage demand in case a considerable share of the available storage is used, or a task or service saturates the available read/write rates of the storage system.

Any resource demand not corresponding to the above cases is set as *low*.

Deadline: A deadline is *close* if computing the result after the deadline would impair an expected quality of experience or QoS for human- or machine-to-machine interaction. Otherwise, the deadline is *far*.

Arrival Type: An arrival type is *periodic* if the requests arrive in a regular pattern. If the requests arrive irregurarly or at random, the arrival type is set to *aperiodic*.

Interarrival Time: The interarrival time is mapped as *short* if the time between two similar requests is shorter than twice the transmission duration of the request itself. It is set to *long* if the time between requests is longer.

V. OUTCOME

Table I presents the outcome of applying the methodology of Section IV to the three uses cases of Section II. We can see that, although the three use cases consider applications from different domains, they exhibit the same characterization with regards to resource demand and deadline.

This first attempt at characterizing use cases suggest that, if an edge benchmark that has the same characterization as the one of the considered use cases is created, at least with regards to resource demand and deadline, then the benchmark workloads could be used for several use cases. It would only be need to have variations with respect to the arrival type and the interarrival time. This is an interesting insight since it implies that the large spectrum of possible edge use cases does not necessarily require a large spectrum of benchmark workloads for testing.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we present a methodology for characterizing edge workloads according to selected characteristics. When applied to a first selection of three use cases from various edge domains, we found that it seems possible to define a limited set of standard workloads to be used as an edge benchmark. Future work includes characterizing more use cases, refining the workload characteristics, and creating standard workloads.

REFERENCES

[1] K. Toczé and S. Nadjm-Tehrani, "A taxonomy for management and optimization of multiple resources in edge computing," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 7 476 201:1–7 476 201:23, 2018.

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

[3] A. Das, S. Patterson, and M. Wittie, "Edgebench: Benchmarking edge computing platforms," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Dec 2018, pp. 175–180.

[4] M. O. J. Olguín Muñoz, J. Wang, M. Satyanarayanan, and 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 (HotMobile '19)*. ACM, 2019, pp. 93–98.