# Towards a Methodology for Benchmarking Edge Processing Frameworks

Pedro Silva, Alexandru Costan, Gabriel Antoniu
*Univ Rennes, Inria, CNRS, IRISA*
Rennes, France
pedro.silva@irisa.fr, alexandru.costan@irisa.fr, gabriel.antoniu@inria.fr

***Abstract*—With the spectacular growth of the Internet of Things, edge processing emerged as a relevant means to offload data processing and analytics from centralized Clouds to the devices that serve as data sources (often provided with some processing capabilities). While a large plethora of frameworks for edge processing were recently proposed, the distributed systems community has no clear means today to discriminate between them. Some preliminary surveys exist, focusing on a feature-based comparison. We claim that a step further is needed, to enable a performance-based comparison. To this purpose, the definition of a benchmark is a necessity. In this short paper, we make a step towards the definition of a methodology for benchmarking Edge processing frameworks.**

***Index Terms*—Edge Computing, Edge Analytics, Fog Computing, Cloud Computing, Benchmarking**

## I. Introduction

Stream processing is a new model for handling continuous flows of data in real-time, which has recently been receiving a lot of attention from the distributed systems community. This is due to the inherent challenges related to its real-time and low-latency requirements, and also to the underlying infrastructure, which breaks a standard that has been established over the last twenty years.

Strongly emerged after the IBM-Google announcement in 2007, *Cloud computing* became the preferred option for deploying distributed applications thanks to features like reduced upfront investment, fast scalability, availability and pay-as-you-go cost model. Applications deployed on the Cloud follow a rather centralized processing model: they typically collect data from the *"edge"* of the network [1] and ship it through the Internet for processing on centralized Cloud datacenters. Lately, with the emergence of the Internet of Things (IoT) and the increasing capabilities becoming available at the edge of the network, a shift was initiated with the goal of leveraging such edge resources *for decentralizing processing*. For example, a connected heat sensor may filter temperatures under 50 degrees Celsius before transmitting data to the Cloud for further processing; or a home assistant may perform a first lexical analysis before requesting a translation to the Cloud.

*Edge computing* is the new paradigm which enables data processing at the Edge. It brings advantages such as: *(i)* the exploitation of unused computing resources on the Edge and their implicit parallelism, *(ii)* reduction of data sent to core machines and consequently transmission costs, and *(iii)* data locality in order to satisfy privacy constraints. However, it may also have downsides such as: *(i)* raising energy consumption on Edge machines or *(ii)* data loss due to limited fault tolerance.

Several frameworks emerged recently to enable Edge computing (cf. Section II-D). Their functionality is twofold: they allow for execution and infrastructure management (e.g., handling the deployed code versions on the machines or data transmission) and data processing (e.g., implementation of analytics algorithms). Our focus is on the latter. Despite the increasing number of available Edge data processing frameworks, there are only a few publications that evaluate and compare them. This makes it hard for system architects to choose the best solution for their scenarios. Most often, this choice is simply made based on arbitrary trade-offs and on empirical, small-scale evaluations.

In this work, our objective is to discuss **a preliminary methodology for comparing Edge processing frameworks through benchmarking**. While the definition and implementation of such a benchmark is still work in progress, this paper discusses existing Edge processing frameworks and preliminary benchmarking efforts (Section II). Furthermore, we propose an initial version of a benchmarking methodology for Edge processing and discuss the main challenges posed by its implementation (Section III).

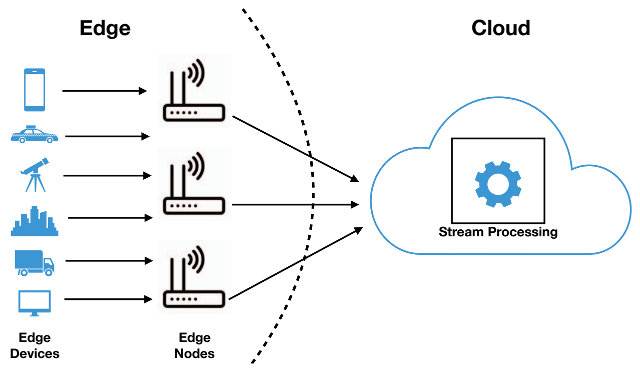

Fig. 1. Edge computing infrastructure. Note that the area where "Edge nodes" are depicted is commonly defined as the Fog.

This work is supported by the ANR OverFlow project (ANR-15-CE25-0003).

## II. Background and Related Work

In this section, we first introduce stream processing applications, which are today the primary users of Edge processing frameworks. Then, we define the concept of Edge computing and discuss the state of the art on Edge processing frameworks.

### A. Stream processing applications

A *data stream*, or simply a *stream*, is an unbounded, continuous flow of data. Stream processing applications typically collect data from sources (e.g., mobile devices, home computers, sensors, etc.) located at the Edge of the networks. Some pre-processing can be done on Edge nodes located close to the data sources, to have fast, real-time results, while most of the data is sent to centralized Cloud datacenters for more complex data analytics (e.g., using batch processing).

### B. Edge computing and edge processing

*Edge computing* is a collection of technologies that allow computation to be performed at the Edge of a network [2], i.e., on the set of processing resources located *between* the devices acting as data sources and the Cloud datacenters. Some authors make a distinction between processing data on the *devices* themselves (e.g., sensors, actuators, mobile phones, etc.) and processing data on *intermediary nodes* (e.g., outpost servers, cell phone base stations, routers, etc.). The former is categorized as Edge computing, and the latter is defined as *Fog computing* (cf. Figure 1). In this work, we consider that the Fog is part of the Edge and ignore this distinction.

*Edge processing* is a particular type of edge computing, whose goal is to process data in support of a data analytics process. Examples of edge processing scenarios are numerous. [3] presents an approach for locally processing events dispatched by sensors installed on trucks in order to reduce event processing latency and transmission costs. In that work, data is processed by a device, installed in the trucks, which is also responsible for emitting events. In [4], local data processing is used to detect anomalies (e.g. leaks) in water transmission pipes in order to reduce latency and to overcome limitations related to the connectivity inside of tunnels. Finally, in [5], the computation of image and video processing from a mobile app is offloaded to servers located near the smart phones, with the objective of improving application performance.

*Enabling Edge processing.* Edge environments are different from more traditional environments such as Clouds in several aspects. In short, Edge processing leverages highly distributed and heterogeneous small machines (e.g., thousands of devices or sensors) connected to core datacenters through limited networks (e.g., with high latency or low bandwidth). Such particular features need to be specifically addressed to enable Edge processing, especially in large-scale scenarios. In the following paragraphs, we highlight three important dimensions of *Edge processing management*.

(a) **Management of software deployment, update and configuration on Edge machines.** In highly distributed scenarios, manually installing and configuring data processing software on each machine may be unfeasible due to issues such as difficult access to the machines, heterogeneity of hardware and number of devices.

(b) **Execution of data processing algorithms.** The resource capacities available to Edge machines are usually restricted compared to Clouds or other computing environments. Hence, tools that are resource consuming (e.g., in terms of storage, processing power or energy consumption) may not be suited for the Edge computing.

(c) **Management of execution and data transmission.** In an environment where connections are poor and machines are highly distributed, not robust (e.g. cheap sensors / devices) and may, sometimes, be physically accessed, it is essential to have an execution management software layer. Besides basic features such as starting and stopping services, it should be also responsible for other more complex tasks like monitoring, fault tolerance and cryptography.

In this work, we focus on dimension (b), hence, we aim at understanding the ecosystem of data processing tools specially adapted for the Edge environment, in support of data analytics. Currently, in the related literature, the notions of "Edge processing" and "Edge analytics" are often used interchangeably (even if, semantically, data analytics refers to higher-level extraction knowledge, while data processing concerns lower-level management of data in motion and of data at rest). In the remaining of this paper we will focus on Edge processing tools, whose aim is to support higher-level data analytics.

### C. Edge processing frameworks

Relevant and important *open-source* research prototype frameworks include Nebula [6] (for generic computing and data processing at the edge), or Geelytics [7] and SpanEdge [8] (focusing on stream processing at the edge). We will however rather focus on production-ready, stable software backed by an open-source community. We find these characteristics in Apache Edgent [9], "an open-source programming model and runtime for edge devices" that enables data and event analysis.

Large technology companies lead the way on private solutions, such as Amazon Greengrass [10], Azure Stream Analytics [11], IBM Watson IoT [12], Intel IoT [13], and Oracle Edge Analytics [14].

With such a rich landscape of private and open-source Edge processing frameworks solutions, it becomes a challenge to a system architect to choose the solution that best adapts to its needs and budget. The problems becomes even more complex if we take into consideration the heterogeneous nature of those tools which may have distinct supported programming languages and protocols, resource requirements, system integration capabilities, and configuration complexities.

### D. Benchmarking Edge processing frameworks

Surveys and benchmarks can bring light into the main characteristics of computer systems and compare them following a scientific methodology.

In the survey of [15], authors present details about many different frameworks acting in multiple dimensions of Edge processing management (cf. Section II-B), including the support for execution of data processing algorithms. Although established tools like Amazon Greengrass and Apache Edgent are not present, the proposed literature review is still interesting. The survey of [16] does not mention tools like Intel IoT and Apache Edgent, however, they present even more frameworks than [15] and also compares their features. In particular, authors propose an interesting state of the art review on less stable academic/research solutions.

Edge computing benchmarks are a bit scarcer in the state of the art, except for device benchmarks. In [17], a simple methodology of benchmark is proposed and three different physical machines are benchmarked. [18] presents a very detailed description of a device benchmark methodology validated and standardized by TPC [1]. In spite of focusing on the device benchmark methodology, there are ideas that can be incorporated in other types of Edge computing benchmark tools, such as infrastructure configuration and workload definition.

Finally, a methodology for performing IoT benchmarks which includes application, workloads and data generation is proposed in [19], but it is only validated on Azure IoT. In [20] a well defined benchmark methodology of Edge processing frameworks is implemented, nevertheless they only compare Azure Streams against AWS Greengras, letting aside other open-source and private solutions.

### E. Discussion

The contributions of the aforementioned works are very important. However, we consider that an Edge processing benchmark of well-established private and open-source tools would be strongly useful to the distributed systems community, as long as it is founded on a well-defined methodology, on realistic use-case applications, and on realistic workloads and infrastructures. To the best of our knowledge, there is no published work covering all these aspects. Our work aims to make a step towards this goal.

## III. Preliminary Benchmarking Methodology

We divide our benchmark methodology in seven elements: *(i)* **benchmark objectives**, *(ii)* **Edge processing frameworks**, *(iii)* **infrastructure**, *(iv)* **scenario applications and input data**, *(v)* **experiment parameters**, *(vi)* **evaluation metrics**, and *(vii)* **benchmark workflow**. In the next subsections, we discuss the main challenges associated to each of those elements and detail our approach to address them.

### A. Benchmark objectives

Benchmarks are often associated with measuring or comparing the *performance* of a computer program or device. However, as the notion of performance is ambiguous, it must be clearly defined. In this work, we call benchmark objectives the characterization of performance, i.e., the definition of the aspects of the tools being benchmarked that will be taken into consideration during evaluation, such as computing power, energy consumption or fault tolerance, for example.

[1]http://www.tpc.org

Our proposed methodology defines a threefold benchmark objective: we evaluate *processing power*, *supported programming languages* and *development easiness*.

### B. Edge processing frameworks

A set of potential candidates was described in Section II-C, namely Apache Edgent, Amazon Greengrass, Azure Stream Analytics, IBM Watson IoT, Intel IoT and Oracle Edge Analytics. It is also important to define *baseline* data processing programs to be used for comparison to the other frameworks. For this purpose, we developed two Edge processing programs, one in C++ (BLC++) and another in Java (BLJava). In short, differently from the benchmarked tools which are generic and optimized to the Edge computing, they hardcode data processing applications and their connections to other software.

### C. Infrastructure

The specification of the machines on which the benchmarked tools will be deployed is a crucial component of a benchmark methodology because it may impact on their performance. In particular, Edge environments are often very heterogeneous, hence, they may have machines with different resource profiles (e.g., sensors, gateways, routers, etc.).

To add this variety of machines to the benchmark, we use a Raspberry Pi 2 (RPI2), bare metal machines from Grid5000[2], a French experimental testbed, and virtual machines deployed on top of Grid5000. Raspberry Pi 2 devices are interesting because, besides their widespread utilization in IoT scenarios and their limited resource profile (1GB RAM and 900MHz quad-core ARM Cortex-A7 CPU), they also allow testing the frameworks on ARM processors. In contrast, we also test the benchmarked frameworks on the robust Dell PowerEdge R630 (Intel Xeon, 2.40GHz, 2 CPUs/node, 8 cores/CPU and 128GB RAM). Machines with resource profiles smaller than and in between those two are simulated with virtual machines.

### D. Use-case applications and input data

Realistic scenario applications and representative data bring credibility to the benchmark. In the proposed methodology, we use two scenarios: taxi ride data from the New York City Taxi and Limousine Commission (TLC) [21] and CCTV footage from the University of California San Diego (CCTV) [22]. The TLC data are used to calculate various aggregated metrics in real-time (e.g., identifying the busiest drivers for some specific time periods). The operations performed in the Edge are the filtering of spurious data and the transformation of data to a specific format. The CCTV scenario can serve for various video data analytics (e.g., identifying the busiest monitored place, in terms of number of people, over a period of time). The operation executed in the Edge is the processing of video frames in order to count the number of persons per frame.

[2]https://www.grid5000.fr

TABLE I
PRELIMINARY LIST OF PARAMETERS OF INTEREST AS DISCUSSED IN SECTIONS III-D AND III-E

| Parameters | Value |
|---|---|
| EPF | {Azure, Amazon, Edgent, IBM, Intel, Oracle, BLC++, BLjava} |
| HM | {Nano, RPI2, Mini, Medium, Large, Dell PE R630} |
| AS | {TLC, CCTV} |
| DT (msg/s) | {100, 1000, 10000} |
| Total executions | $8 \times 6 \times 2 \times 3 = 288$ |

### *E. Parameters of interest*

A key point of a benchmark methodology is the designation of parameters of interest (or knobs) whose values variation and combination creates different scenario configurations. It is important to keep a balance between having as many representative parameters as possible without generating an excessive number of scenarios.

Currently, we consider four benchmark parameters: **Edge processing framework** (EPF), **hosting machine** (HM), **application scenario**(AS) and **input data throughput** (DT). Their preliminary values are described in Table I.

### *F. Evaluation metrics*

In order to understand the results of the experiments and to compare the performance of each benchmarked framework, it is necessary to define performance metrics. In summary, they describe how to process results, partial results and other indicators with the objective of measuring performance. Hence, evaluation metrics and benchmark objectives must be in phase.

In this work, taking into consideration that our benchmark objectives are processing power, supported programming languages and development easiness, we consider five metrics: **message processing throughput** (number of messages processed per second), **average message processing latency** (average of time taken to process all messages), **quantity of supported programming languages**, **quantity of supported connectors to other frameworks**, and **number of lines of code** needed to implement each scenario.

### *G. Benchmark workflow*

The benchmark workflow is a *plan* which describes the necessary steps for performing the execution of the frameworks using combinations of the values of parameters of interest. It describes the deployment and configuration of the infrastructure, the installation and configuration of the Edge processing frameworks, the setup and upload of data to the host machines, the execution control of the benchmarked tools (i.e., when to start and when to end the execution), and the collection and transmission of the results.

## IV. CONCLUSION

In this short paper, we make a step towards the definition of a methodology for benchmarking Edge processing frameworks. We discuss state-of-the art Edge processing tools that enable data analytics and point out the absence of benchmarks for Edge processing frameworks. We present a preliminary methodology that we are developing as the foundation of an Edge processing framework benchmark that we are currently implementing. We divide our methodology into six parts and discuss associated challenges such as defining sufficiently heterogeneous host machines, choosing representative applications and input data, and choosing the right set of benchmark parameters and evaluation metrics. Once implemented, this benchmark would make a step beyond existing feature-based comparisons available in the related work, typically based on their documentation: it will serve to compare a large number of private and open-source Edge processing frameworks in a performance-oriented fashion.

## REFERENCES

[1] M. Satyanarayanan, "The emergence of edge computing," *Computer*, 2017.
[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, 2016.
[3] R. Young, S. Fallon, and P. Jacob, "An architecture for intelligent data processing on iot edge devices," *UKSim-AMSS*, 2017.
[4] S. Kartakis, W. Yu, R. Akhavan, and J. A. McCann, "Adaptive edge analytics for distributed networked control of water systems," in *IEEE IoTDI*, 2016.
[5] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan, "Scalable crowd-sourcing of video from mobile devices," in *Proceeding of MobiSys '13*, 2013.
[6] M. Ryden, K. Oh, A. Chandra, and J. Weissman, "Nebula: Distributed edge cloud for data intensive computing," in *Proceedings of the 2014 IEEE International Conference on Cloud Engineering*, 2014.
[7] B. Cheng, A. Papageorgiou, F. Cirillo, and E. Kovacs, "Geelytics: Geo-distributed edge analytics for large scale iot systems based on dynamic topology," in *IEEE WF-IoT*, 2015.
[8] H. P. Sajjad, K. Danniswara, A. Al-Shishtawy, and V. Vlassov, "Spanedge: Towards unifying stream processing over central and near-the-edge data centers," in *IEEE/ACM SEC*, 2016.
[9] Apache, "Apache edgent." [Online]. Available: https://edgent.incubator.apache.org
[10] Amazon, "Aws iot greengrass." [Online]. Available: https://aws.amazon.com/greengrass/
[11] Microsoft, "Azure stream analytics." [Online]. Available: https://azure.microsoft.com/en-us/services/stream-analytics/
[12] IBM, "Ibm watson iot." [Online]. Available: https://www.ibm.com/internet-of-things
[13] Intel, "Intel iot." [Online]. Available: https://software.intel.com/en-us/iot/home
[14] Oracle, "Oracle edge analytics." [Online]. Available: https://www.oracle.com/middleware/technologies/complex-event-processing.html
[15] M. A. A. da Cruz, J. J. P. C. Rodrigues, J. Al-Muhtadi, V. V. Korotaev, and V. H. C. de Albuquerque, "A reference model for internet of things middleware," *IEEE Internet of Things Journal*, April 2018.
[16] A. J. Ferrer, J. M. Marquês, and J. Jorba, "Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing," *ACM Comput. Surv.*, Jan. 2019.
[17] C. P. Kruger and G. P. Hancke, "Benchmarking internet of things devices," in *2014 12th IEEE INDIN*, 2014.
[18] M. Poess, R. Nambiar, K. Kulkarni, C. Narasimhadevara, T. Rabl, and H.-A. Jacobsen, "Analysis of tpcx-iot: The first industry standard benchmark for iot gateway systems," 2018.
[19] A. Shukla, S. Chaturvedi, and Y. Simmhan, "Riotbench: A real-time iot benchmark for distributed stream processing platforms," *CoRR*, 2017.
[20] A. Das, S. Patterson, and M. Wittie, "Edgebench: Benchmarking edge computing platforms," in *2018 IEEE/ACM UCC Companion*, 2018.
[21] B. Donovan and D. B. Work, "Using coarse gps data to quantify city-scale transportation system resilience to extreme events." in *Transportation Research Board 94th Annual Meeting*, 2015.
[22] A. B. Chan and N. Vasconcelos, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.