Benchmarking an EDGELESS Cluster for Serverless Edge Computing Applications

Claudio Cicconetti
Institute of Informatics and Telematics, National Research Council – Pisa, Italy
c.cicconetti@iit.cnr.it

***Abstract*—The EDGELESS project is set to efficiently operate serverless computing in extremely diverse computing environments, from resource-constrained edge devices to highly-virtualised cloud platforms. Automatic deployment and reconfiguration will leverage AI/ML techniques, resulting in a flexible horizontally-scalable computation solution able to fully use heterogeneous edge resources while preserving vertical integration with the cloud and the benefits of serverless and its companion programming model, i.e., Function-as-a-Service (FaaS). The system under design will be environmentally sustainable, as it will dynamically concentrate resources physically (e.g., by temporarily switching off far-edge devices) or logically (e.g., by dispatching tasks towards a specific set of nodes) at the expense of performance-tolerant applications. Evaluating the performance of such a complex system in realistic conditions is a daunting task. In the paper, we provide an overview of the ongoing efforts to achieve this goal.**

***Index Terms*—Stateful FaaS, Serverless Computing**

I. Introduction

EDGELESS is a collaborative project funded by the European Commission under the Horizon Europe program that aims to leverage the serverless concept [1] in all the layers in the edge-cloud continuum to fully benefit from diverse and decentralised computational resources available on demand close to where data are produced or consumed. In particular, we aim at realising an efficient and transparent *horizontal* pooling of the resources on edge nodes with constrained capabilities or specialised hardware, smoothly integrated with cloud resources, which is a giant leap forward compared to state-of-the-art *vertical* offloading solutions where the edge is a mere supplement of the cloud.

Benchmarking plays a vital role in understanding the effectiveness of the solutions developed in such a new and complex system. However, assessing the performance of edge computing systems, in general, is known to be a difficult task, due to the lack of significant datasets [2]. While in the literature some tools have been proposed, even in the specific case of serverless computing [3]–[6], these solutions are tailored to specific aspects that do not fit the distinguishing features of EDGELESS, including the stateful nature of functions, its flexible deployment across the edge-cloud continuum, and the use of lightweight virtualisation abstractions (instead of containers, which are far more common). Therefore, in the project, we developed a framework, called `edgeless_benchmark` that is designed specifically to assess the performance of an EDGELESS cluster in relevant conditions.

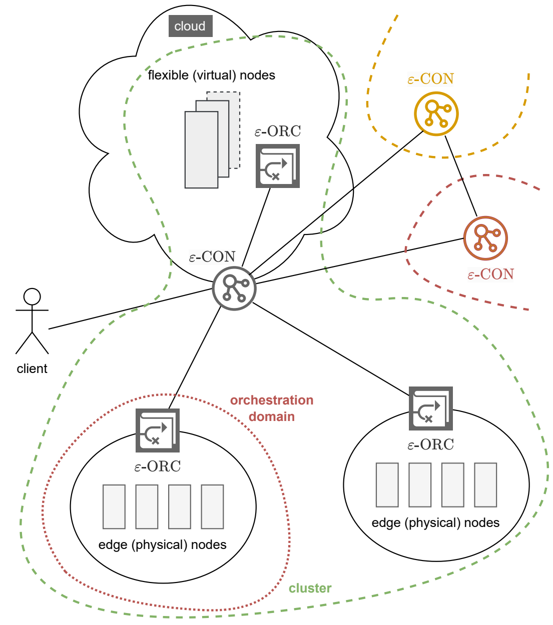

Fig. 1. EDGELESS benchmark test application.

In Section II we briefly introduce the project architecture. In Section III we introduce the local observability means, which are used in benchmarks to measure performance. In Section IV we describe some relevant features of the benchmarking framework.

II. Architecture

The EDGELESS architecture is illustrated with the help of the example deployment in Figure 1 and has been designed to support broad deployment scenarios involving edge and cloud computing infrastructures. The deployment spans two clusters, one of which shows its internal structure consisting of three orchestration domains. Two of those domains are mapped to physically (and possibly geographically) separated edge computing infrastructures, while the third uses virtual resources offered by a third-party cloud provider under an Infrastructure as a Service (IaaS) model. The cloud operators used by the clusters can be different. Each orchestration domain, at the edge or in the cloud, is managed by an ε-ORC, but the worker nodes are different between these two cases. At the edge, the physical resources available are limited by the type, capacity, and number of nodes installed at a given site (nodes may range from embedded devices to small computers to full-fledged servers), while nodes in the cloud run inside a highly virtualised environment, using containers or virtual machines.

III. Observability

The local observability metrics in EDGELESS are collected by each node and transferred to the ε-ORC in response to periodic keep-alive polls. The keep-alive response message consists of two parts. First, the node's health status information, including system information about the edge node, like CPU average load, memory occupation, and network transfers. Second, the performance samples are obtained by the node's telemetry system and include the running times of all the function invocations. Furthermore, it is possible to have application-specific metrics, which are triggered from within the functions by using a dedicated Application Programming Interface (API) made available to the developers.

IV. Benchmarking

`edgeless_benchmark` is a tool to help function developers and designers of orchestration algorithms through the automated performance evaluation of a population of workflows in controlled conditions.

Arrival models supported:

- *poisson*: inter-arrival between consecutive workflows and lifetimes are exponentially distributed.
- *incremental*: one new workflow arrives every new inter-arrival time, with constant lifetime.
- *incr-and-keep*: add workflows, with constant lifetimes, incrementally until the warm up period finishes, then keep until the end of the experiment.
- *single*: add a single workflow that lasts for the entire experiment.
- *trace*: read the arrival and end times of workflows from a file specified.

Workflow types supported:

- *single*: a single function.
- *matrix-mul-chain*: a chain of functions, each performing the multiplication of two matrices of 32-bit floating point random numbers at each invocation.
- *vector-mul-chain*: a chain of functions, each performing the multiplication of an internal random matrix of 32-bit floating point numbers by the input vector received from the caller.
- *map-reduce*: a workflow consisting of a random number of stages, where each stage is composed of a random number of processing blocks. Before going to the next stage, the output from all the processing blocks in the stage before must be received.

The duration of the experiment is configurable via a command-line option, like the seed used to generate pseudo-random numbers to enable repeatable experiments and the duration of a warm-up period.

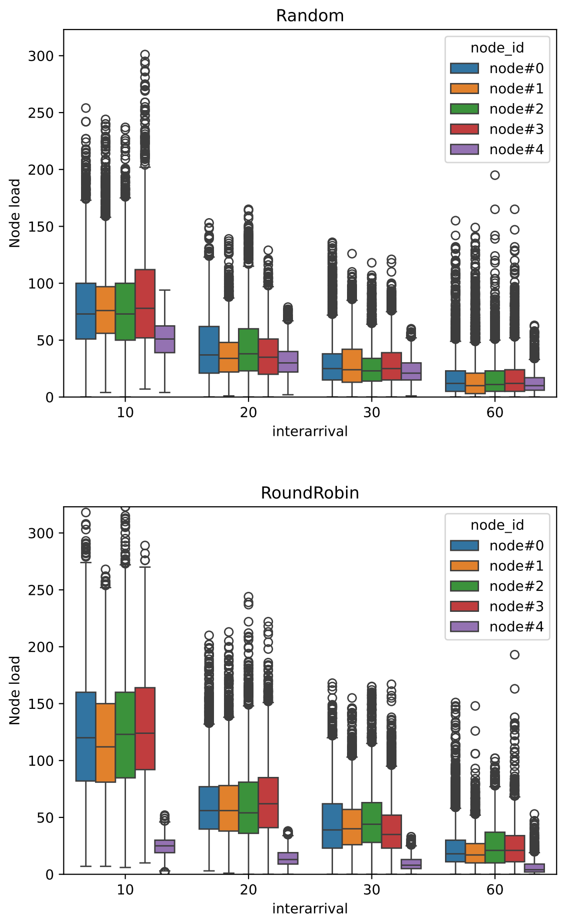

Fig. 2. EDGELESS benchmark example results.

The benchmark allows obtaining results like those in Figure 2, which were obtained in a cluster of 5 edge nodes: 4 with fewer computing capabilities (node#0 to node#3) and 1 more powerful (node#4). We used a Poisson arrival time, with average interarrival varying from 10 to 60 seconds. Each experiment lasted 1 hour and was repeated 10 times. We ran separately experiments with two orchestration policies:

- *Random*: function instances are assigned to nodes in a weighted random manner, with the weight proportional to the number of cores of each node.
- *RoundRobin*: function instances are assigned in a round-robin manner, without considering the nodes' capabilities.

As can be seen, with RoundRobin node#4 is under-utilised because it is assigned the same amount of computational burden as the others, which are less capable. On the other hand, Random yields a more balanced allocation of resources, by exploiting knowledge about the nodes' capabilities, which is advertised to the ε-ORC by the nodes upon registering to the orchestration domain.

Acknowledgment

The work was partly funded by the European Union under the project EDGELESS (GA no. 101092950).

References

[1] Y. Li, Y. Lin, Y. Wang, K. Ye, and C.-Z. Xu, "Serverless Computing: State-of-the-Art, Challenges and Opportunities," *IEEE Transactions on Services Computing*, vol. 1374, no. c, pp. 1–1, 2022.

[2] O. Kolosov, G. Yadgar, S. Maheshwari, and E. Soljanin, "Benchmarking in the dark: On the absence of comprehensive edge datasets," *HotEdge 2020 - 3rd USENIX Workshop on Hot Topics in Edge Computing*, 2020.

[3] N. Somu, N. Daw, U. Bellur, and P. Kulkarni, "PanOpticon: A Comprehensive Benchmarking Tool for Serverless Applications," *2020 International Conference on COMmunication Systems and NETworkS, COMSNETS 2020*, pp. 144–151, 2020, publisher: IEEE ISBN: 9781728131870.

[4] J. Scheuner, S. Eismann, S. Talluri, E. van Eyk, C. Abad, P. Leitner, and A. Iosup, "Let's Trace It: Fine-Grained Serverless Benchmarking using Synchronous and Asynchronous Orchestrated Applications," 2022, arXiv: 2205.07696. [Online]. Available: http://arxiv.org/abs/2205.07696

[5] F. Carpio, M. Michalke, and A. Jukan, "BenchFaaS: Benchmarking Serverless Functions in an Edge Computing Network Testbed," pp. 1–7, 2022, arXiv: 2206.02150. [Online]. Available: http://arxiv.org/abs/2206.02150

[6] S. Chaitanya Palepu, D. Chahal, M. Ramesh, and R. Singhal, "Benchmarking the Data Layer Across Serverless Platforms," *Proceedings of the 2nd Workshop on High Performance Serverless Computing (HiPS '22), June 30, 2022, Minneapolis, MN, USA*, vol. 1, 2022, iSBN: 9781450393119. [Online]. Available: https://doi.org/10.1145/3526060.3535460