2022 22nd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)

Pushing Serverless to the Edge with WebAssembly Runtimes

Philipp Gackstatter, Pantelis A. Frangoudis, and Schahram Dustdar
Distributed Systems Group, TU Wien, Vienna, Austria
Email: philipp.gackstatter@student.tuwien.ac.at, pantelis.frangoudis@dsg.tuwien.ac.at, dustdar@dsg.tuwien.ac.at

2022 22nd International Symposium on Cluster, Cloud and Internet Computing (CCGrid) | 978-1-6654-9956-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/CCGRID54584.2022.00023

***Abstract*—Serverless computing has become a popular part of the cloud computing model, thanks to abstracting away infrastructure management and enabling developers to write functions that auto-scale in a polyglot environment, while only paying for the used compute time. While this model is ideal for handling unpredictable and bursty workloads, cold-start latencies of hundreds of milliseconds or more still hinder its support for latency-critical IoT services, and may cancel the latency benefits that come with proximity, when serverless functions are deployed at the edge. Moreover, CPU power and memory limitations which often characterize edge hosts drive latencies even higher. The root of the problem lies in the de facto runtime environments for serverless functions, namely container technologies such as Docker. A radical approach is thus to replace them with a more light-weight alternative. For this purpose, we examine WebAssembly's suitability for use as a serverless container runtime, with a focus on edge computing settings, and present the design and implementation of a WebAssembly-based runtime environment for serverless edge computing. WOW, our prototype for WebAssembly execution in Apache OpenWhisk, reduces cold-start latency by up to 99.5%, can improve on memory consumption by more than 5×, and increases function execution throughput by up to 4.2× on low-end edge computing equipment compared to the standard Docker-based container runtime for various serverless workloads.**

***Index Terms*—Function-as-a-Service, edge computing, serverless, WebAssembly**

I. Introduction

When latency matters and data intensity is high, executing IoT service logic in the cloud is challenged, due to the latter's physical distance and the sheer amount of data devices at the edge of the network generate. Combined with privacy concerns, this calls for pushing services to the edge, for data processing and decision making in-place or nearby. However, applying traditional IaaS or PaaS cloud models to host such services at edge infrastructure can have adverse effects, due to resource limitations that make virtual machine or container allocation and scaling expensive [1]. This further complicates the elastic management of event-driven IoT services, which are often associated with bursty and unpredictable workloads. A model with finer grained resource elasticity is thus required.

One piece of the puzzle towards realizing this goal is serverless (edge) computing [2]. Serverless, commonly in the form of Function-as-a-Service (FaaS), allows developers to execute functions over cloud infrastructure without having to specify how the latter is set up, managed, and auto-scaled. The serverless provider ensures precise per-function provisioning and pure pay-per-use at the function level, following the *scale-to-zero* principle, i.e., de-allocating resources not in use.

To isolate function instances in a multi-tenant environment, serverless frameworks make use of OS-level virtualization through containers–usually of the Docker flavor. When a function is first invoked, its container is provisioned from scratch. This is referred to as *cold start* and can introduce latencies of hundreds of ms or more [3]. For example, median cold start latencies of 250-265 ms and 110-493 ms have been reported [4] for AWS and Google Cloud Platform, respectively. Particularly when operating at low-end edge computing equipment [5]–[7], and in the face of concurrent requests [8] typical of bursty workloads, cold start latencies are driven further up.

At the heart of the problem lies the container runtime and its expensive startup procedure. This issue is partially addressed by keeping containers warm in-between requests. For instance, Apache OpenWhisk,[1] a popular open-source serverless framework, keeps a function's container paused and ready for reuse for 10 minutes, before removing it entirely, while AWS Lambda's cold start policy keeps an instance alive for 5-7 minutes [9]. However, this is a form of over-provisioning and therefore opposed to the scale-to-zero premise.

We believe OS-level virtualization to be unsuitable for serverless edge computing. We thus follow a more radical approach, as has been recently suggested [10]–[12], by replacing the container runtime with an alternative offering more efficient cold starts. A technology that can play this role is WebAssembly (Wasm) [13], a portable, binary instruction format for memory-safe, sandboxed execution. Its portability means that a compiled Wasm function can be executed wherever a runtime exists. Wasm can be compiled with different strategies, some reaching near-native execution speeds to compete with function invocation within a Docker container. Importantly, Wasm functions can be created and destroyed in microseconds.

With this background, we set off to address our key research question: *the viability of WebAssembly in serverless edge computing and the performance benefits it can bring about*. We make the following contributions: ① We design an execution environment for Wasm serverless functions, with the requirements of multi-platform edge execution and ease of integration with existing serverless frameworks in mind (§III). ② We present WOW (§IV), a prototype for executing Wasm workloads in Apache OpenWhisk. WOW is extensible in

[1] https://openwhisk.apache.org/

978-1-6654-9956-9/22/$31.00 ©2022 IEEE
DOI 10.1109/CCGrid54584.2022.00023