M2 PDS Research Project | Starting 2026-10-12

ngFaaS: Next-generation software stack for efficient Function-as-a-Service platforms

Mathieu Bacou

SAMOVAR, Télécom SudParis, Inria, Institut Polytechnique de Paris, France
mathieu.bacou@telecom-sudparis.eu

Keywords: Serverless, FaaS, virtualization, containers, distributed systems

I. Context

Cloud computing is now a major foundation of digital infrastructures. Among its service models, *Function-as-a-Service* (FaaS) enables the automatic execution of software functions in response to events, without requiring the client to manage the allocation of the necessary hardware and software infrastructure resources.

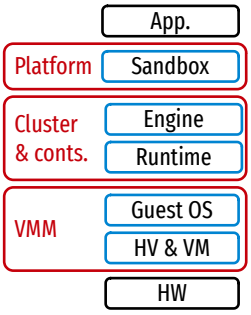


Figure 1: Current FaaS system stacks.

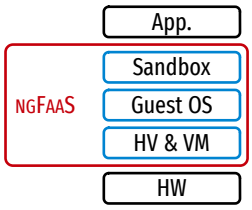


Figure 2: ngFaaS stack.

The current design of the software stack of FaaS platforms, illustrated in Figure 1, has emerged from existing container clusters orchestrators based on virtual nodes, that targeted the *model of micro-services*. It results in a mismatch between the properties of the stack, and its usage. We identify this discrepancy as a source of two main challenges faced by FaaS platforms:

Resource overhead The many layers and roles in a micro-services stack converted to a FaaS stack, which must now individually manage user function instances, have lead to huge resource overheads [4], thus reducing the cost-effectiveness of FaaS data-centers.

Startup latency If no function instance is available, a FaaS platform must first start one up, on the critical path of serving a request: this is a cold start [3]. They incur significant latency overheads [2, 6], that are amplified by every layer of the current FaaS stack, thus reducing the usefulness of the FaaS paradigm.

Those challenges are in part due to how *function sandboxes* are managed in the FaaS stack. It is based on some combination of hardware-based (HW) virtualization (e.g., virtual machines with QEMU/KVM) and software-based (SW) virtualization (e.g., containers with containerd). Existing works tried to tackle the issues of the HW+SW FaaS stack, by integrating more closely the SW into the HW virtualization for security purposes [1], or by focusing on efficient SW virtualization *despite* the HW [5].

II. Research direction

In the ngFaaS project, we seek to design the next-generation FaaS stack. Previous works missed the opportunity *to holistically rethink the FaaS stack* based on FaaS requirements and principles. Thus, resource usage overhead remains — it is only moved away from the SW+HW virtualization stack, and can only be alleviated. Moreover, the startup latency remains a critical issue because cold starts cannot be avoided entirely: either all functions benefit from the previous techniques, inflating resource usage to absurd amounts; or only the most popular ones do, but then the other ones experience cold starts [2].

Quickly put, a central point of ngFaaS's holistic design is to *put back the Operating System at the center* — to put it back in control of the local computing resources. Consequently, the project revises both the control plane for the management of function sandboxes, and the data plane to transmit requests and responses. It follows that ngFaaS targets both the Serverless distributed platform, and the sandbox software stack on the cluster workers.

III. Research project goals

A baseline Serverless distributed platform has already been implemented, and in this project, the student will work on the *sandbox stack*, local to a worker in the cluster. The imagined design of the stack is shown in Figure 2.

The student will start from preliminary work on the control and data planes in the guest OS. As an introduction to the project, their first task will be to finish the full integration of the data plane: transmitting requests and responses between the guest OS and the FaaS platform, and between the guest OS and the function sandboxes.

Then, they will work on findings answers that match ngFaaS's design tenants, to the following open research questions:

- what scheduling and resource management mechanisms are needed by the guest OS to control its sandboxes?
- what is the "sandbox" abstraction the guest OS must provide in a FaaS context?

Any work will be validated experimentally, to check for performance and resource usage efficiency.

IV. Work environment

The student will integrate into the research team Benagil of Inria, who studies how to design more efficient and secure distributed systems by focusing on their core components.

Technology-wise, the student will gain experience in bleeding-edge cloud infrastructures, from the Serverless distributed platform to the virtualization and containerization stacks of worker nodes. Tangentially, they will also learn more about networking and scheduling. They can expect development in systems languages: mostly C because they will mostly work in the Linux kernel, Go if work is needed on the experimental platform or existing containerization technologies, otherwise left to the student's preference.

Bibliography

[1] Alexandru Agache, Marc Brooker, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana- Maria Popa. 2020. Firecracker: Lightweight Virtualization for Serverless Applications. In *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*, 2020. USENIX Association. Retrieved from https://www.usenix.org/conference/nsdi20/presentation/agache

[2] Xiaohu Chai, Tianyu Zhou, Keyang Hu, Jianfeng Tan, Tiwei Bie, Anqi Shen, Dawei Shen, Qi Xing, Shun Song, Tongkai Yang, Le Gao, Feng Yu, Zhengyu He, Dong Du, Yubin Xia, Kang Chen, and Yu Chen. 2025. Fork in the Road: Reflections and Optimizations for Cold Start Latency in Production Serverless Systems. In *19th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2025, Boston, MA, USA, July 7-9, 2025*, 2025. USENIX Association. Retrieved from https://www.usenix.org/conference/osdi25/presentation/chai-xiaohu

[3] Artjom Joosen, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Luke Nicholas Darlow, Jianfeng Wang, Qiwen Deng, and Adam Barker. 2025. Serverless Cold Starts and Where to Find Them. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, 2025. ACM. Retrieved from https://doi.org/10.1145/3689031.3696073

[4] Zijun Li, Jiagan Cheng, Quan Chen, Eryu Guan, Zizheng Bian, Yi Tao, Bin Zha, Qiang Wang, Weidong Han, and Minyi Guo. 2022. RunD: A Lightweight Secure Container Runtime for High-density Deployment and High-concurrency Startup in Serverless Computing. In *Proceedings of the 2022 USENIX Annual Technical Conference, USENIX ATC 2022, Carlsbad, CA, USA, July 11-13, 2022*, 2022. USENIX Association. Retrieved from https://www.usenix.org/conference/atc22/presentation/li-zijun-rund

[5] Edward Oakes, Leon Yang, Dennis Zhou, Kevin Houck, Tyler Harter, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci- Dusseau. 2018. SOCK: Rapid Task Provisioning with Serverless-Optimized Containers. In *Proceedings of the 2018 USENIX Annual Technical Conference, USENIX ATC 2018, Boston, MA, USA, July 11-13, 2018*, 2018. USENIX Association. Retrieved from https://www.usenix.org/conference/atc18/presentation/oakes

[6] Mohammad Shahrad, Rodrigo Fonseca, Iñigo Goiri, Gohar Irfan Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. 2020. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. In *Proceedings of the 2020 USENIX Annual Technical Conference, USENIX ATC 2020, July 15-17, 2020*, 2020. USENIX Association. Retrieved from https://www.usenix.org/conference/atc20/presentation/shahrad