M2 internship (pre-PhD) | March - September 2026 (6 months)

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

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

Keywords: Serverless, Function-as-a-Service, virtualization, containers, resource efficiency

1 Context

Cloud, i.e., renting remote computing resources, is the main method for deploying applications at scale. It has eventually reflected on the architecture of the applications: we observe a trend of "cloud native" designs. Focusing on *Serverless*, applications are designed to deployed on *Function-as-a-Service* (FaaS) platforms. It means that their features are served by composing and replicating simple functions.

The current design of the *software stack* of FaaS platforms 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 [7], 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 the issue of *cold starts* [5]. They incur significant latency overheads [9, 3], 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 [2, 6]) and software-based (SW) virtualization (e.g., containers with containerd [4]). 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 [8].

2 Goals

In the ngFaaS project, we have the goal of designing 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 [3].

With ngFaaS, we will start from the latest developments concerning the cluster orchestrator, to include these advances into a novel design for the FaaS stack that provides function sandboxes by using HW and SW virtualization in *active collaboration*. The work will be to study existing solutions in HW and SW virtualization, identify opportunities to improve the state of function sandboxes, and propose and implement new designs to tackle the issues of resource usage and startup latency.

References

[1] Alexandru Agache et al. "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*. Ed. by Ranjita Bhagwan and George Porter. USENIX Association, 2020. URL: https://www.usenix.org/conference/nsdi20/presentation/agache.

[2] Fabrice Bellard. "QEMU, a fast and portable dynamic translator." In: *USENIX annual technical conference, FREENIX Track*. Vol. 41. 46. California, USA. 2005.

[3] Xiaohu Chai et al. "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*. Ed. by Lidong Zhou and Yuanyuan Zhou. USENIX Association, 2025. URL: https://www.usenix.org/conference/osdi25/presentation/chai-xiaohu.

[4] containerd. *containerd*. [Online; access 1-September-2025]. 2025. URL: https://containerd.io/.

[5] Artjom Joosen et al. "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*. ACM, 2025. URL: https://doi.org/10.1145/3689031.3696073.

[6] KVM. *KVM*. [Online; accessed 1-September-2025]. 2023. URL: https://linux-kvm.org/index.php?title=Main_Page&oldid=174096.

[7] Zijun Li et al. "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*. Ed. by Jiri Schindler and Noa Zilberman. USENIX Association, 2022. URL: https://www.usenix.org/conference/atc22/presentation/li-zijun-rund.

[8] Edward Oakes et al. "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*. Ed. by Haryadi S. Gunawi and Benjamin C. Reed. USENIX Association, 2018. URL: https://www.usenix.org/conference/atc18/presentation/oakes.

[9] Mohammad Shahrad et al. "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*. Ed. by Ada Gavrilovska and Erez Zadok. USENIX Association, 2020. URL: https://www.usenix.org/conference/atc20/presentation/shahrad.