Research project

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, as illustrated in fig. 1.

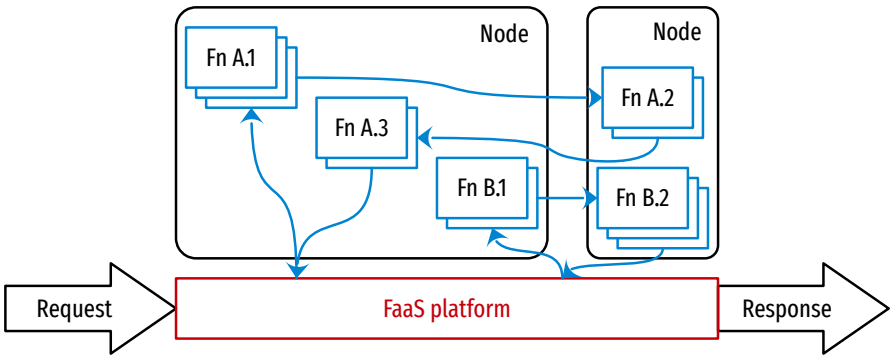

Figure 1: Serving a request with a FaaS application.

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 [8], thus reducing the cost-effectiveness of FaaS data-centers:

- CPU and memory overheads, as costly abstractions are stacked without concert [5];
- disk overheads, because sandboxes are based on badly-shared container images [13];
- networking overheads, due to bad scheduling and costly virtual networking [12].

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* [6]. They incur significant latency overheads [11, 3], that are amplified by every layer of the current FaaS stack, thus reducing the usefulness of the FaaS paradigm:

- each layer gets in the way of provisioning the sandbox and loading the user workload;
- the stack lacks active collaboration, both vertically (i.e., across layers) and horizontally (i.e., in every layer throughout the cluster).

Those challenges cover two components: the cluster orchestrator, and function sandboxes.

Cluster orchestrator. Previous works [5, 9] have established that cluster orchestration made for microservices is a very bad fit: it is too complex (in architecture and abstractions), and it offers too many guarantees (e.g., for fault tolerance) that are unnecessary for FaaS. That makes them very slow due to I/O and synchronization operations on the critical paths.

Function sandboxes. The FaaS stack is based on some combination of hardware-based (HW) virtualization (e.g., virtual machines with QEMU/KVM [2, 7]) 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 [10].

2 Goals

In the ngFaaS project, we have the goal of designing the next-generation FaaS stack.

Previous works missed the opportunity to *holistically redesign 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.

In details, the student will be tasked with:

1. experimenting with state-of-the-art container engines (yet to be determined);
2. experimenting with Firecracker [1], a state-of-the-art hypervisor for functions;
3. experimenting with Dirigent [5], the state-of-the-art Serverless cluster orchestrator, and analyzing its impact on the function sandboxes stack;

4. analyzing opportunities to build a more efficient sandboxing stack by making both virtualization layers actively collaborate;
5. implementing and evaluating proposals to employ those opportunities.

3 Work environment

The student will work under the direct supervision of Mathieu Bacou. They will integrate into the research team Benagil of Inria and the Parallel and Distributed Systems (PDS) team of the CS department of Télécom SudParis.

Technology-wise, the student(s) will gain experience in bleeding-edge system components of cloud and FaaS platforms, including virtualization layers (hypervisor and container runtimes). Development can be expected in systems language (mostly C/C++, possibly Go or Rust, otherwise left to the students' preference).

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] Lazar Cvetkovic et al. "Dirigent: Lightweight Serverless Orchestration". In: *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles, SOSP 2024, Austin, TX, USA, November 4-6, 2024*. Ed. by Emmett Witchel et al. ACM, 2024. URL: https://doi.org/10.1145/3694715.3695966.

[6] 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.

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

[8] 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.

[9] Qingyuan Liu et al. "The Gap Between Serverless Research and Real-world Systems". In: *Proceedings of the 2023 ACM Symposium on Cloud Computing, SoCC 2023, Santa Cruz, CA, USA, 30 October 2023 - 1 November 2023*. ACM, 2023. URL: https://doi.org/10.1145/3620678.3624785.

[10] 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.

[11] 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.

[12] Shelby Thomas et al. “Particle: ephemeral endpoints for serverless networking”. In: *SoCC ’20: ACM Symposium on Cloud Computing, Virtual Event, USA, October 19-21, 2020*. Ed. by Rodrigo Fonseca, Christina Delimitrou, and Beng Chin Ooi. ACM, 2020. URL: https://doi.org/10.1145/3419111.3421275.

[13] Hanfei Yu et al. “RainbowCake: Mitigating Cold-starts in Serverless with Layer-wise Container Caching and Sharing”. In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1, ASPLOS 2024, La Jolla, CA, USA, 27 April 2024- 1 May 2024*. Ed. by Rajiv Gupta et al. ACM, 2024. URL: https://doi.org/10.1145/3617232.3624871.