An Approach for Energy-Efficient, Fault Tolerant FaaS Platforms

Yasmina Bouizem

An Approach for Energy-Efficient, Fault Tolerant FaaS Platforms

Yasmina Bouizem
Univ Rennes, Inria, Univ Tlemcen, LRIT

ABSTRACT

Recent years have seen the emergence of serverless computing and, in particular, the "Function as a Service" (FaaS) paradigm, which lets users run arbitrary functions without being concerned about operational issues.

This paper describes several research questions about how to design better Function-as-a-Service platforms in term of performance, fault tolerance and energy consumption. I have started tackling these open questions, starting with an evaluation of Kubernetes-native FaaS frameworks.

1 INTRODUCTION

The FaaS model promises reduced operational costs, reduced packaging and deployment complexity, high availability, and automated elasticity [15], and has been applied in diverse domains, such as IoT, machine learning inference, and image processing. All the main cloud providers have embraced the FaaS model with offerings such as AWS Lambda [1], Google Cloud Functions [4], and Azure Functions [2]. A growing number of open-source platforms enable implementing FaaS systems on private clouds [3, 5, 12]. Nevertheless, current FaaS platforms are immature and cannot effectively meet the needs of their stakeholders, that is, the users that deploy applications, and the provider that owns and operates the platform. The aim of this work is to develop an automated resource management solution for FaaS platforms that satisfies the performance and availability requirements of users, while reducing the energy consumption for the provider.

The paper is structured as follows. Section 2 presents a survey of related reseach in serverless computing. Section 3 describes challenges and gaps identified in FaaS frameworks. Section 4 proposes an approach to improve FaaS platforms. Finally, in section 5, a conclusion is drawn.

2 BACKGROUND AND RELATED WORK

Recently several works related to the FaaS paradigm have been proposed.

In [14] authors evaluated serverless computing environments with respect to concurrent invocations by deploying a series of functions for large distributed data processing. They compared the performance of CPU, memory and disk-intensive functions and measured throughput to understand the function behaviour on serverless computing environments. [15] focused on identifying factors which influence the performance of microservices deployed on serverless platforms. The authors investigated hosting implications related to infrastructure elasticity, load balancing, provisioning variation, infrastructure retention, and memory reservation size. [17] proposed a novel prototype of a queuing scheme implemented in. NET, deployed in Microsoft Azure, and utilizing Windows containers as function execution environments. The authors proposed metrics, such as throughput, to evaluate the performance of the proposed prototype as well as the following serverless platforms: AWS Lambda, Azure Functions, Google Cloud Functions, and Apache OpenWhisk. [19] evaluated OpenWhisk performance with different numbers of concurrent functions. The author plans to investigate isolation within the language runtime itself and to build a new multi-tenant serverelss language runtime to reduce latency seen by users while improving resource efficiency for the provider. [8] introduced a novel, high-performance serverless platform, SAND, that provides lower latency, better resource efficiency and more elasticity than existing serverless platforms. The authors implemented and deployed a complete SAND system. Their results show that SAND outperforms state-of-the-art serverless platforms such as OpenWhisk. [9] identified key characteristics and use cases for FaaS and described technical challenges and open problems. [16] presented the major enterprise serverless cloud computing platforms and a comparison based on general and high-level technical features such as scalability, log management, orchestration, dependencies and maximum number of functions. [20] analysed the current serverless scheduling problem and introduced the new allocation heuristic NOAH to solve it. [21] identified some performance challenges that arise specifically in the FaaS model.

There are further challenges identified in related papers: scheduling policies, cold starts, session management, code and data locality, load balancing and resource allocation among others [7, 10, 13].

The focus so far has been on analysing the benefits and limitations of current serverless technology, identifying factors which influence serverless computing performance, and trying to come up with solutions to improve performance. There is currently no work that addresses performance concerns combined with availability and energy efficiency concerns in FaaS. In this work, we are going to tackle this challenge and study the trade-offs between the three parameters.

3 CHALLENGES FOR FAAS FRAMEWORKS

Despite the attractive features promised by FaaS, current platforms tend to be immature and suffer from multiple limitations with regards to resource management. First, the platforms provide no support for managing the quality of service experienced by FaaS user. For instance, users have no influence on the latency of function execution. Second, the platforms provide no fault-tolerance mechanisms beyond a basic retry mechanism that brings up a new container when a failure is detected. Implementing further mechanisms, such as active or semi-active replication, would allow greater flexibility in the way that performance and availability requirements are managed. Third, the platforms do not take into account energy savings in resource management decisions. Finally, the platforms provide no integrated management system that takes into account the objectives of users and the provider in a coordinated manner. For example, most platforms offer auto-scaling capabilities, but these focus on optimizing the scaling of individual functions and applications.

4 PROPOSED APPROACH

To address the open issues discussed before, we focus on FaaS frameworks based on Kubernetes [6] and deployed on private infrastructures, following the architecture shown in Figure 1. Our resource management solution will be implemented in the FaaS framework layer, exploiting the configurability and extensibility of Kubernetes. In particular, the solution will allow FaaS users to express their requirements regarding performance and availability, for example, through defining different levels of quality of function execution. The solution will then take into account these requirements along with information on the current state of the system in order to take appropriate resource management decisions. Examples of such decisions include determining the placement of functions on containers and nodes, selecting appropriate replication mechanisms, dynamically scaling functions as well as reacting to failures and workload changes. Making these decisions will be based on performance and energy consumption models which will be generated using profiling and historical information [11, 18].

As a first step towards developing the solution, we are evaluating three popular open source FaaS frameworks (Kubeless [5], Fission [3] and OpenFaaS [12]), all based on Kubernetes, which allow deploying serverless on private infrastructures without any form of vendor lock-in. In particular, we are evaluating these frameworks with respect to their performance, fault-tolerance, and energy consumption properties as well as their extensibility; the goal is to select the most appropriate framework for hosting our resource management solution.

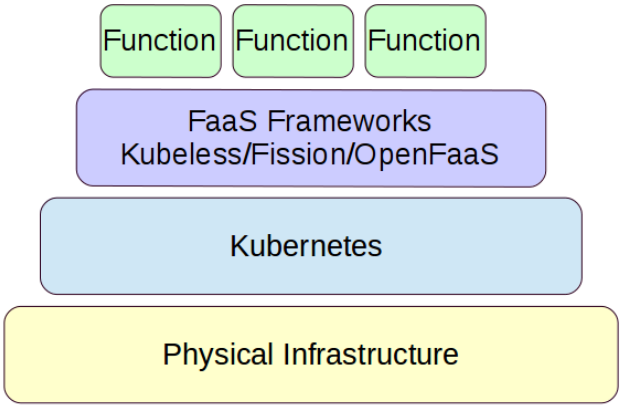

Figure 1: Kuberetes native FaaS frameworks architecture

5 CONCLUSION

The emerging FaaS model is the latest step in the long trend towards higher levels of abstraction in cloud programming models. The model enables developers to write stateless code without worrying about provisioning, configuring, and managing servers. However, current FaaS platforms provide little support for managing performance, availability, and energy efficiency requirements, which reduces the value of the platforms, and impedes their adoption in private clouds.

This paper highlighted the key challenges in addressing this problem and discussed related work in industry and research. The paper also gave an outline of our approach and the first steps in developing a solution to the problem. We hope that our solution can help bring the many benefits of serverless computing to private clouds.

ACKNOWLEDGMENTS

I would like to thank Christine Morin, Djawida Dib and Nikos Parlavantzas for their valuable comments and helpful suggestions.

REFERENCES

[1] [n. d.]. AWS Lambda – Serverless Compute - Amazon Web Services. https://aws.amazon.com/lambda/
[2] [n. d.]. Azure Functions—Serverless Architecture - Microsoft Azure. https://azure.microsoft.com/en-us/services/functions/.
[3] [n. d.]. Fission - Serverless functions for kubernetes. https://fission.io/.
[4] [n. d.]. Google Cloud Functions. https://cloud.google.com/functions/.
[5] [n. d.]. What is Kubeless. https://kubeless.io/.
[6] 2018. Kubernetes. https://kubernetes.io/.
[7] Gojko Adzic and Robert Chatley. 2017. Serverless computing: economic and architectural impact. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 884–889.
[8] Istemi Ekin Akkus, Ruichuan Chen, Ivica Rimac, Manuel Stein, Klaus Satzke, Andre Beck, Paarijaat Aditya, and Volker Hilt. 2018. SAND: towards high-performance serverless computing. In *Proceedings of the USENIX Annual Technical Conference (USENIX ATC)*.
[9] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, et al. 2017. Serverless computing: Current trends and open problems. In *Research Advances in Cloud Computing*. Springer, 1–20.
[10] Rajkumar Buyya, Satish Narayana Srirama, Giuliano Casale, Rodrigo Calheiros, Yogesh Simmhan, Blesson Varghese, Erol Gelenbe, Bahman Javadi, Luis Miguel Vaquero, Marco AS Netto, et al. 2017. A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade. *arXiv preprint arXiv:1711.09123* (2017).
[11] Djawida Dib, Nikos Parlavantzas, and Christine Morin. [n. d.]. SLA-based PaaS profit optimization. *Concurrency and Computation: Practice and Experience* 29, 21 ([n. d.]), e4251. https://doi.org/10.1002/cpe.4251 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4251 e4251 cpe.4251.
[12] A. Ellis. [n. d.]. OpenFaaS. https://docs.openfaas.com/.
[13] Scott Hendrickson, Stephen Sturdevant, Tyler Harter, Venkateshwaran Venkataramani, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. 2016. Serverless computation with openlambda. *Elastic* 60 (2016), 80.
[14] Hyungro Lee, Kumar Satyam, and Geoffrey C Fox. [n. d.]. Evaluation of Production Serverless Computing Environments. ([n. d.]).
[15] Wes Lloyd, Shruti Ramesh, Swetha Chinthalapati, Lan Ly, and Shrideep Pallickara. 2018. Serverless computing: An investigation of factors influencing microservice performance. In *Cloud Engineering (IC2E), 2018 IEEE International Conference on*. IEEE, 159–169.
[16] Theo Lynn, Pierangelo Rosati, Arnaud Lejeune, and Vincent Emeakaroha. 2017. A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms. In *Cloud Computing Technology and Science (CloudCom), 2017 IEEE International Conference on*. IEEE, 162–169.
[17] Garrett McGrath and Paul R Brenner. 2017. Serverless computing: Design, implementation, and performance. In *Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference on*. IEEE, 405–410.
[18] N. Parlavantzas, L. M. Pham, A. Sinha, and C. Morin. 2018. Cost-Effective Reconfiguration for Multi-Cloud Applications. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. 521–528. https://doi.org/10.1109/PDP2018.2018.00088
[19] Simon Shillaker. [n. d.]. A Provider-Friendly Serverless Framework for Latency-Critical Applications. ([n. d.]). http://conferences.inf.ed.ac.uk/EuroDW2018/papers/eurodw18-Shillaker.pdf.
[20] Manuel Stein. 2018. The Serverless Scheduling Problem and NOAH. *arXiv preprint arXiv:1809.06100* (2018).
[21] Erwin van Eyk, Alexandru Iosup, Cristina L Abad, Johannes Grohmann, and Simon Eismann. 2018. A SPEC RG cloud group's vision on the performance challenges of FaaS cloud architectures. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*. ACM, 21–24.