Proceedings Paper:
Djemame, K orcid.org/0000-0001-5811-5263 (2023) Serverless Computing: Introduction and Research Challenges. In: Bañares, JÁ, Altmann, J, Agmon Ben-Yehuda, O, Djemame, K, Stankovski, V and Tuffin, B, (eds.) Economics of Grids, Clouds, Systems, and Services. 19th International Conference, GECON 2022, 13-15 Sep 2022, Izola, Slovenia. Lecture Notes in Computer Science, 13430 . Springer , pp. 15-23. ISBN 978-3-031-29315-3

https://doi.org/10.1007/978-3-031-29315-3_2

Serverless Computing: Introduction and Research Challenges

Karim Djemame[1][0000−0001−5811−5263]

School of Computing, University of Leeds, UK
K.Djemame@leeds.ac.uk
https://eps.leeds.ac.uk/computing/staff/187/professor-karim-djemame

Abstract. Serverless computing is a technology that offers the ability to create modular, highly-scalable, fault-tolerant applications, leveraging container-based virtualisation to deploy applications and services. It is revolutionising the way we think about application development, and serverless platforms are already ubiquitous in the public and private sectors. Commercial solutions dominate the market with widespread adoption from industry giants such as Amazon, Google and Microsoft, though open-source solutions do exist such as Apache OpenWhisk, Fission and OpenFaaS. This tutorial will present the state-of-the-art in serverless computing research, and provide useful insights into the main challenges that motivate researchers to work on this topic. It will also identify research gaps for future research.

Keywords: Serverless computing · · Cloud Computing · Containerisation · Performance Evaluation.

1 Introduction

The goal of serverless computing is to provide isolated environments that abstract underlying technologies and expose small runtime containers for users to run functions as code [13]. It provides a resource-efficient, low overhead alternative to Virtual Machines (VMs) and containers. Serverless computing simply means that the serverless platform allows a developer to build code and deploy it without ever needing to configure or manage underlying servers. The unit of deployment is the code; not the container that hosts the code, or the server that runs the code, but simply the code itself. However, a major requirement for writing serverless code is the ability to express the logic as *functions* that are instantiated to process a single event triggered by a service.

With zero infrastructure and maintenance costs and little-to-no operating expense, a serverless computing platform is an ideal solution to build and optimise any Internet of Things (IoT) operation as it allows IoT businesses to offload all of a server's typical operational backend responsibilities [20]. Moreover, such system is a natural fit for edge computing applications as serverless computing also supports the protocols which IoT devices require in actual deployment conditions.

For a map of state-of-the-art research on the topic of FaaS platform and tooling engineering together with analysis of relations of the proposed concepts to existing solutions, the reader is referred to [35]: the mapping study on engineering FaaS platforms and tools provides insights on publication trends, the common challenges and drivers for research as well as information on industry participation in research publications.

2 Service Models

Serverless computing is linked to mainly two service models [35], similar to the ones that originally emerged with the rise of cloud computing:

1. *Backend as a Service(BaaS)*: This refers to services that offer features traditionally implemented by back-end applications such as databases or API servers. Users can incorporate them in their front-end web applications without the provisioning, setup and management of servers [11]. Although similar to Platform as a Service (PaaS), they are more full-featured, implementing server side logic such as user authentication or push/pull notifications which PaaS offerings forego in favor of more flexibility.
2. *Function as a Service(FaaS)*: This model allows users to develop their application logic by compositing event-driven executable elements called functions. These functions are executed inside ephemeral containers, taking advantage of container virtualization to quickly provision resources for the duration of the execution. Most notably these containers are managed by the providers and scale automatically in number based on demand. FaaS is the most prominent model of serverless computing and has seen widespread

3 Commercial Offering

Serverless computing has seen widespread adoption from tech industry giants such as Amazon [1], Microsoft [7] and Google [5]. *Amazon AWS Lambda* is a service for executing stateless functions in the cloud, triggered by events, with transparent provisioning and no infrastructure management from users [1]. It has since grown into one of the most widely used and researched FaaS platforms, as evidenced by the extensive literature on performance evaluation [22] and investigations into its viability in various application domains [18].

Similarly, Microsoft have general availability of their own FaaS service, *Azure Functions*, citing a much wider service-function binding capability, allowing functions to be triggered by events from external services. Azure Functions is built on top of Microsoft's PaaS offering, Azure App Service, and theWebJobs SDK with a runtime layer on top to support multiple programming languages. Google also have *Cloud Functions*, their own brand of serverless, and so does IBM which developed of what would eventually become *OpenWhisk*, the open-source serverless platform that embraces community collaboration. OpenWhisk was released to the general public and admitted into the Apache Foundation and was the basis for IBM's commercial FaaS product, *Cloud Functions*.

4 Open Source Solutions

Serverless computing has also seen adoption from the public domain, with open-source projects like *Apache OpenWhisk* [2], *Fission* [4], *OpenFaas* [8], *IronFunctions* [6] and more.

OpenLambda [21] is an open-source platform for building web services applications using the serverless computing model. OpenWhisk [2] follows a simple event-driven architecture – functions are being triggered in response to events originating from direct invocations to the OpenWhisk API, or external services that are integrated in the platform through specialized packages. These community-developed packages allow functions to be triggered when an external action is being performed, such as when a new commit is pushed to a GitHub repository, or a file has finished uploading to a Dropbox folder. Every action in OpenWhisk is transformed and routed through its RESTful HTTP-based API that serves as the platform's single point of entry.

5 Research Categories

An investigation of the challenges and drivers that motivate researchers to engineer new or extend existing FaaS platforms and platform-specific tools is found in [35]. The state of the art on the topic of developing new or enhancing existing FaaS platforms and tools engineering focuses on the following: 1) Function Execution; 2) Platform deployment environment; 3) Testing and observability; 4) Benchmarking; 5) Costs optimisation; 6) Programming models; 7) Research-centric platforms; 8) Deployment automation; 9) Migration, and 10) Continuous integration / Continuous delivery pipeline.

One of the findings in [35] is that the challenges tackled by researchers are heterogeneous and target different layers of the FaaS platform, with a bulk of the work focusing on optimizing the performance of function execution aspects using various strategies, e.g., optimize or replace function runtime, improve function scheduling and resources allocation.

Moreover, underlying *metrics* that are considered in the literature include [34]:

- *Communication performance*: this is important in function composition, and is particularly used for complex serverless applications such as sequence and nested chain;
- *Startup latency*: although function execution time is usually short, a 15× factor differential is noted between *cold* and *warm* times [26]. To avoid start-up latency, the underlying virtualisation technology must be optimised to decrease the start-up latency of *cold* start and can be tackled by suitable managing of the function instances in the serverless platform, e.g., reusing launched instances by keeping them *warm* for a period of time [25], or reduction of the container image size.

- *Stateless overhead*: serverless functions are by defnition stateless. Two options are considered for an application that wants to preserve state across function executions: 1) the state can be fully encapsulated within the event message passing between the function executions. If there are multiple functions executing in a workflow, each function will find the full application state it needs in the event message it receives, operate on it, and pass it on to the next function, and 2) persist state across function executions is to utilise a storage system, e.g. AWS S3.
- *Resource efficiency*: from the platform perspective, this is the ability to colocate serverless functions with other workloads for utilisation improvement. For users, the aim is to provision resources for Quality of Service and economy benefit.

6 Research Challenges

There are a number of research challenges in serverless computing such as unreliability, large overheads and an absence of benchmarks [16]. Investigations into various aspects of serverless architectures are therefore required to guide the decision making process. In the following, some of these challenges are discussed.

6.1 Performance

There has been extensive research around factors affecting function execution performance [31] as well as some evaluations of commercial and open-source serverless frameworks. A comprehensive evaluation of Apache OpenWhisk is evaluated in [14], with a series of experiments designed and implemented to assess its performance in terms of effectiveness and efficiency. Two metrics were of interest in experimentation: function runtime and resource utilisation. Experiments also involved creation of two alternate solutions used as benchmarks for the results produced by OpenWhisk to provide some context and means for comparison: *Docker* [3] and *native*. The results of experiments showed that OpenWhisk could outperform a solution which employed similar functionality, through use of container-based virtualisation.

Another important point to consider in the context of performance is virtualisation technology. For example, AWS Lambda uses Firecracker micro-VMs [17] which provide enhanced security and workload isolation over traditional VMs, while enabling the speed and resource efficiency of containers.

6.2 Serverless Composition

Serverless applications can be made of multiple functions in a chain, and consequently unreasonable composition may incur high communication overhead. For example, network functions are usually short-lived and in most scenarios, they are chained together to form a Service Function Chain (SFC) [12]. The development and performance tuning of SCFs are difficult and should be considered

in designing the more complex application scenarios. Considering serverless parallelism , especially in the context of autoscaling can benefit applications with high efficiency. How to control and manage such parallelism remains an open question.

6.3 Utilisation

Serverless functions are usually scaled up and down on-demand in a serverless platform. Load balancing manages resource utilisation by distributing the function executions to available resources. Function efficiency heavily depends on the resources allocated but at the same time maximising resource utilization is a hard problem, especially in a large scale environment, e.g. cloud computing [34].

As noted previously, an application can be launched as multiple orchestrated functions. To avoid undesired latency and network overhead (and therefore energy consumption), the function executions belonging to the same session can be assigned to the same server. For latency sensitive communication, locality requirements are considered to group functions as a single application.

6.4 Programming Models

Serverless platforms are typically focused on FaaS, where applications need to be redesigned as a set of event-triggered functions implemented in a supported programming language. Consequently, this requires additional effort for the developers as many applications cannot be easily redesigned as a set of functions.

The adoption of serverless computing in Big Data has attracted the attention of researchers and developers. Examples of data analytics over serverless platforms are found in [19] to perform data processing with Spark over Apache OpenWhisk Ooso is an open source tool based on managed cloud services, Amazon S3 and AWS Lambda that allows running MapReduce jobs in a serverless way [29]. Another open-source framework for building serverless applications is AWS Serverless Application Model (SAM) [10] which provides shorthand syntax to express functions, APIs, databases, and event source mappings as well as application modelling using YAML. A programming model designed to create highly-parallel event-driven file-processing serverless applications for serverless architectures is presented in [28]. This programming model allows the user to run generic applications, including legacy ones, on serverless platforms .

6.5 Energy Efficiency

Applications' performance lies with not only efficient node-level execution but energy consumption as well as these applications, e.g. IoT, may operate in a low energy computing environment. A serverless platform does not take into account energy savings in resource management decisions, and therefore addressing performance concerns combined with availability and energy efficiency concerns in serverless computing becomes important.

Serverless platform can be extended to support resource mapping and load balancing to increase resource utilisation by distributing the function executions to available resources with the aim to minimise power consumption. A load balancing strategy should consider functions interactions by assigning the function executions belonging to the same session to the same server. Latency sensitive communication services require careful placement of functions by allowing locality requirements for grouping functions as a single application. Containers image sizes are reduced to speed up the start of a function execution thus avoiding cold start.

The work in [24] proposes RAEF, a function-level runtime system for allocating the resources required by serverless functions in order to minimize power consumption, and which consists of predictors, resource explorer, monitor and coordinator. The work in [12] describes a modular, and micro-service based Software Defined Network (SDN) architecture that applies network programmability within the context of Network Function Virtualisation (NFV) and explores how it could benefit from the serverless computing paradigm. Experimental results show that the serverless paradigm can decrease service latency for disaggregated architectures, and also provide on-demand and scalable resource management. The reduction in the execution time and the average resource usage of microservices allows for many optimizations from the resource management point of view. Follow-up work addresses performance concerns combined with energy efficiency support in serverless computing [9]. To this aim, a number of experiments are conducted to compare the power consumption of a serverless platform, OpenFaaS, against Docker containers with the consideration of applications and benchmarks, driven by SDN and NFV requirements. The experimental results show that OpenFaaS is more power efficient than Docker when the processor and memory are under stress.

6.6 Language Runtimes

One of the most detrimental factors affecting performance in serverless architectures is the notion of cold start that takes place when the first incoming request to an application leads to a time-consuming allocation of resources which delays the response and leads to bad user experience [11]. Research has shown that the choice of language runtime plays a non-trivial role in the performance of serverless applications. In particular, the cold start times differ significantly across different languages and platforms [15] . Various papers have investigated the performance impact the choice of language runtime has on function execution as well as measured runtime overhead through the consideration of different use cases, e.g. a performance and cost analysis of language choice on AWS Lambda and Azure Functions [23]; a just-in-time dynamic language outperforming the compiled alternatives [27]; identification of factors influencing function performance, interpreted languages, effect of cold and warm start [32]; effect of resource provisioning and how it affects performance on commercial platforms (AWS Lambda, Azure, Google Functions) [33]; and comparison of dynamic and compiled languages [30] .

Some areas for further research include: 1) the evaluation of more trigger types for invoking functions (e.g. database updates, timers, message queues); 2) the evaluation of additional serverless platforms (e.g. Knative, OpenFaaS, Kubeless and Iron Functions), and 3) the investigation custom runtimes when serverless platforms offer the ability for a custom executable to be used as a runtime environment.

7 Conclusion

This paper has introduced Serverless Computing, a cloud computing model where the exact amount of resources needed by applications is dynamically allocated on-demand. Function-as-a-service (FaaS) is the service model that allows developers to run code directly in the cloud without the need to build packages or maintain any infrastructure. The paper has surveyed and elaborated the research domains in the serverless context and provided useful insights into the main challenges that motivate researchers to work on this topic.

Acknowledgements

The author would like to thank the Next Generation Internet Program for Open INTErnet Renovation (NGI-Pointer 2) for supporting this work under contract 871528 (EDGENESS Project).

References

1. Amazon Web Services. AWS Lambda (2019), https://aws.amazon.com/lambda/
2. Apache Openwhisk. Open Source Serverless Cloud Platform (2019), https://openwhisk.apache.org/
3. Docker. Empowering App Development for Developers (2019), http://www.docker.com
4. Fission - Open source, Kubernetes-native Serverless Framework (2019), https://fission.io
5. Google. Google Cloud Functions (2019), https://cloud.google.com/functions
6. IronFunctions - Open Source Serverless Computing (2019), https://open.iron.io/
7. Microsoft Azure. Azure Functions (2019), https://azure.microsoft.com/en-us/services/functions/
8. OpenFaaS - Serverless Functions, Made Simple (2021), https://openfaas.com/
9. Alhindi, A., Djemame, K., Banaie, F.: On the power consumption of serverless functions: an evaluation of openfaas. In: Proceedings of the 15th IEEE/ACM International Conference on Utility and Cloud Computing (UCC'2022). IEEE, Vancouver, Washington (Dec 2022)
10. Amazon: AWS Serverless Application Model (AWS SAM) (2022), https://github.com/awslabs/serverless-application-model
11. Baldini, I., Castro, P.C., Chang, K.S., Cheng, P., Fink, S.J., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R.M., Slominski, A., Suter, P.: Serverless computing: Current trends and open problems. CoRR **abs/1706.03178** (2017)

12. Banaie, F., Djemame, K.: A serverless computing platform for software defined networks. In: Proceedings of the 19th International Conference on the Economics of Grids, Clouds, Systems and Services (GECON'2022). Springer, Izola, Slovenia (Sep 2022)
13. Castro, P., Ishakian, V., Muthusamy, V., Slominski, A.: Serverless programming (function as a service). In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). pp. 2658–2659 (2017)
14. Djemame, K., Parker, M., Datsev, D.: Open-source Serverless Architectures: an Evaluation of Apache OpenWhisk. In: 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC). pp. 329–335 (2020)
15. Djemame, K., Datsev, D., Kelefouras, V.I.: Evaluation of language runtimes in open-source serverless platforms. In: van Steen, M., Ferguson, D., Pahl, C. (eds.) Proceedings of the 12th International Conference on Cloud Computing and Services Science, CLOSER 2022,, Online Streaming, April 27-29, 2022. pp. 123–132. SCITEPRESS (2022)
16. van Eyk, E., Iosup, A.: Addressing performance challenges in serverless computing. In: Proceedings of ICT.OPEN 2018. ACM, Amersfoort, The Netherlands (March 2018)
17. Firecracker: Firecracker: Secure and fast microVMs for serverless computing (2021), https://firecracker-microvm.github.io/
18. Giménez-Alventosa, V., Moltó, G., Caballer, M.: A framework and a performance assessment for serverless MapReduce on AWS Lambda. Future Generation Computer Systems **97**, 259 – 274 (2019), http://www.sciencedirect.com/science/article/pii/S0167739X18325172
19. Glikson, A.: Transit: Flexible pipeline for iot data with bluemix and openwhisk (2018), https://goo.gl/ThN2TR
20. Großmann, M., Ioannidis, C., Le, D.T.: Applicability of Serverless Computing in Fog Computing Environments for IoT Scenarios. In: Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion. p. 29–34. UCC '19 Companion, Association for Computing Machinery, New York, NY, USA (2019)
21. Hendrickson, S., Sturdevant, S., Harter, T., Venkataramani, V., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H.: Serverless computation with openlambda. In: Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing. p. 33–39. HotCloud'16, USENIX Association, USA (2016)
22. Jackson, D., Clynch, G.: An investigation of the impact of language runtime on the performance and cost of serverless functions. In: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). pp. 154–160 (2018)
23. Jackson, D., Clynch, G.: An investigation of the impact of language runtime on the performance and cost of serverless functions. In: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion. pp. 154–160 (2018)
24. Jia, X., Zhao, L.: Raef: Energy-efficient resource allocation through energy fungibility in serverless. In: 2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS). pp. 434–441. IEEE (2021)
25. Li, Z., Guo, L., Cheng, J., Chen, Q., He, B., Guo, M.: The serverless computing survey: A technical primer for design architecture. ACM Comput. Surv. **54** (Sep 2022)
26. Lloyd, W., Ramesh, S., Chinthalapati, S., Ly, L., Pallickara, S.: Serverless computing: An investigation of factors influencing microservice performance. In: 2018 IEEE International Conference on Cloud Engineering (IC2E). pp. 159–169 (2018)

27. Manner, J., Endreß, M., Heckel, T., Wirtz, G.: Cold start influencing factors in function as a service. In: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). pp. 181–188 (2018)
28. Pérez, A., Moltó, G., Caballer, M., Calatrava, A.: A programming model and middleware for high throughput serverless computing applications. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. p. 106–113. SAC '19, ACM, New York, NY, USA (2019)
29. Platform, D.O.S.: Ooso: Serverless MapReduce (2017), https://github.com/d2si-oss/ooso
30. Sbarski, P., Cui, Y., Nair, A.: Serverless Architectures on AWS. Manning, 2nd edn. (2022)
31. Scheuner, J., Leitner, P.: Function-as-a-service performance evaluation: A multi-vocal literature review. Journal of Systems and Software **170**, 110708 (2020)
32. Vojta, R.: AWS journey — API gateway & Lambda & VPC performance (2016), https://www.zrzka.dev/aws-journey-api-gateway-lambda-vpc-performance/
33. Wang, L., Li, M., Zhang, Y., Ristenpart, T., Swift, M.: Peeking behind the curtains of serverless platforms. In: Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference. p. 133–145. USENIX ATC '18, USENIX Association, USA (2018)
34. Yu, T., Liu, Q., Du, D., Xia, Y., Zang, B., Lu, Z., Yang, P., Qin, C., Chen, H.: Characterizing serverless platforms with serverlessbench. In: Proceedings of the 11th ACM Symposium on Cloud Computing (SOCC'2020). p. 30–44. ACM, New York, NY, USA (2020)
35. Yussupov, V., Breitenbücher, U., Leymann, F., Wurster, M.: A systematic mapping study on engineering function-as-a-service platforms and tools. In: Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing. Auckland, New Zealand (Dec 2019)