Beyond Microbenchmarks: The SPEC-RG Vision for a Comprehensive Serverless Benchmark

Beyond Microbenchmarks: The SPEC-RG Vision for A Comprehensive Serverless Benchmark

Erwin van Eyk
e.vaneyk@atlarge-research.com
Vrije Universiteit Amsterdam

Joel Scheuner
scheuner@chalmers.se
Chalmers | University of Gothenburg

Simon Eismann
simon.eismann@uni-wuerzburg.de
University of Wuerzburg

Cristina L. Abad
cabad@fiec.espol.edu.ec
Escuela Superior Politecnica del Litoral

Alexandru Iosup
A.Iosup@atlarge-research.com
Vrije Universiteit Amsterdam

ABSTRACT

Serverless computing services, such as Function-as-a-Service (FaaS), hold the attractive promise of a high level of abstraction and high performance, combined with the minimization of operational logic. Several large ecosystems of serverless platforms, both open- and closed-source, aim to realize this promise. Consequently, a lucrative market has emerged. However, the performance trade-offs of these systems are not well-understood. Moreover, it is exactly the high level of abstraction and the opaqueness of the operational-side that make performance evaluation studies of serverless platforms challenging. Learning from the history of IT platforms, we argue that a benchmark for serverless platforms could help address this challenge. We envision a comprehensive serverless benchmark, which we contrast to the narrow focus of prior work in this area. We argue that a comprehensive benchmark will need to take into account more than just runtime overhead, and include notions of cost, realistic workloads, more (open-source) platforms, and cloud integrations. Finally, we show through preliminary real-world experiments how such a benchmark can help compare the performance overhead when running a serverless workload on state-of-the-art platforms.

KEYWORDS

serverless computing, function-as-a-service, performance

ACM Reference Format:
Erwin van Eyk, Joel Scheuner, Simon Eismann, Cristina L. Abad, and Alexandru Iosup. 2020. Beyond Microbenchmarks: The SPEC-RG Vision for A Comprehensive Serverless Benchmark. In *ACM/SPEC International Conference on Performance Engineering Companion (ICPE '20 Companion), April 20–24, 2020, Edmonton, AB, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3375555.3384381

1 INTRODUCTION

Serverless computing and its ecosystem are rapidly evolving [4, 12, 14], with an increasing number of open-source and managed serverless platforms available to cloud users. AWS, Google, Microsoft, and other tech giants compete in this space, and tens of open- and closed-source serverless platforms already exist [10]. Yet the performance of these systems is poorly understood. Although prior studies target specific aspects of serverless platforms, no comprehensive benchmark exists. In this work, we propose our vision for a serverless benchmark that covers all facets of serverless function executions.

The core aim of serverless computing is to abstract away the operational complexity of distributed systems. More concretely, *serverless computing* is a form of cloud computing that allows users to run event-driven applications with fine-grained billing, without having to address the operational logic [11]. Within this broad domain, we focus specifically on *Function-as-a-Service (FaaS)*: a form of serverless computing where the cloud provider manages the resources, lifecycle, and event-driven execution of user-provided functions and, more recently, function compositions (workflows).

Serverless computing and FaaS have seen rapid adoption owing to their promise of fast time-to-market, delegated operational logic, and autoscaling. Since the release of AWS Lambda in late 2014, the serverless market has grown to be worth around $5 billion, and is estimated to grow to $15 billion by 2023.[1] Similarly, The serverless ecosystem has evolved into a vast landscape of tools and platforms [10, 14]. For the FaaS model specifically, major cloud providers now offer their own serverless platform [14], e.g., AWS Lambda, Microsoft Azure Functions, and Google Cloud Functions. Simultaneously, many open-source platforms emerged from industry and academia, e.g., OpenWhisk, Fission, and OpenLambda.

Despite the overall interest in the domain, the performance of serverless platforms is not well-understood. Serverless and FaaS raise important new performance challenges [1], and their salient features—fine granularity of the function abstraction and the lack of insight in the operational parts that enable lifecycle management by the cloud operator—further make it challenging to understand the key performance trade-offs of current platforms. As argued recently [3], not being able to overcome these challenges could limit system designers and hamper the progress of serverless technology.

Understanding performance is also important for users of serverless platforms. However, it is challenging for users to evaluate the various platforms and determine which one suits their requirements best. Whereas in established domains users can deploy benchmarks to improve their decision-making, performance evaluation in serverless computing remains largely an open problem.

Addressing the dearth of performance information and knowledge in the field, performance-evaluation studies have started to emerge. However, as we elaborate further on in Section 2.2, these

[1] https://www.marketsandmarkets.com/Market-Reports/serverless-architecture-market-64917099.html

ICPE '20 Companion, April 20–24, 2020, Edmonton, AB, Canada

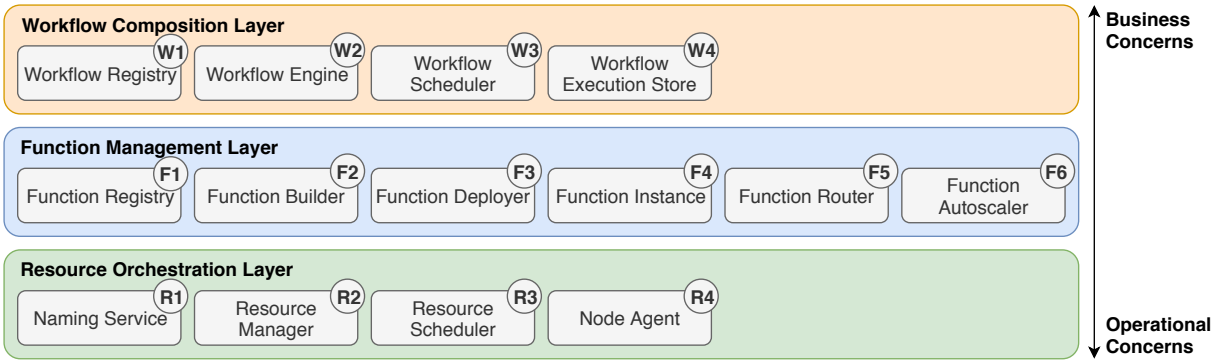

Figure 1: The SPEC-RG reference architecture for FaaS platforms (reproduced from [10]).

studies typically only focus on narrow aspects of these platforms, such as runtime overhead or duration of cold starts, and prevalently focus on single applications or a narrow subset of microbenchmarks that mimic specific application-behavior.

In this work, we envision a comprehensive serverless benchmark. Toward this end, our contribution is three-fold:

(1) We analyze the challenges of benchmarking serverless platforms (Section 2). We argue that a comprehensive benchmark will need to take into account diverse metrics and include notions of cost, focus on more realistic workloads and thus go far beyond mere microbenchmarks, and support more platforms and cloud integrations. Furthermore, there is a practical requirement that a serverless benchmark must offer support for leading serverless platforms, including both closed-source cloud platforms and open-source platforms.
(2) We present a route towards the design of a comprehensive serverless benchmark (Section 3). The route defines and limits the scope of the benchmark, an overview of its key features, implementation details, and ideas for the longer-term future. The key feature of the current design is that it addresses the aspects raised by our first contribution.
(3) We present real-world experimental results (Section 4). The experiments cover some basic performance aspects of platforms of AWS, Google, and Microsoft, and of an open-source platform, Fission. Although preliminary, these results indicate a comprehensive serverless benchmark can help with comparing the performance of serverless platforms. The results of Fission further indicate that benchmarking open-source platforms can lead to a better understanding than benchmarking closed-source platforms.

2 WHAT ARE THE NEEDS OF SERVERLESS BENCHMARKS? WHAT HAS BEEN DONE?

There have been attempts to address the need for a serverless benchmark. Although prior studies provide useful insights,[2] we argue that a comprehensive serverless benchmark is still unavailable.

[2]In future work, we plan to give an extensive and systematic survey of existing serverless benchmarks. The topic is too complex and technical to be introduced here.

2.1 What are the challenges specific to benchmarking serverless platforms?

With the large and dynamic serverless ecosystem, proper benchmarks are sorely needed. The heterogeneous architectures of these platforms and their implications are poorly understood, especially for the proprietary serverless platforms of major cloud vendors.

Although benchmarks exist for other cloud models (e.g., IaaS), we argue that benchmarks are needed that specifically target serverless platforms. The reason for this is that there are challenges specific to benchmarking the performance of these platforms, of which we describe the following five:

(1) **Performance requirements.** Compared to traditional cloud models (such as scheduling VM-based workloads), serverless computing workloads have more stringent performance requirements. Instead of overheads of minutes, for user-facing functions in FaaS, the deployment, and execution overheads are measured in milliseconds. Moreover, not only have performance requirements for the individual serverless services become detailed and increasingly more sophisticated, the workloads are far more fine-grained in nature—leading to more pressure on the scheduler's performance.
(2) **System opaqueness.** Serverless platforms are opaque by design, attempting to abstract away from the cloud user as much of the operational logic as possible. Despite the benefits of this model, the higher level of abstraction impedes our understanding of what and how internal and external factors influence the performance and other characteristics.
(3) **System heterogeneity.** The ecosystem consists of widely heterogeneous systems, which have different approaches to how functions are built, deployed, scaled, upgraded, and executed. Although we proposed a reference architecture (see Figure 1) for these platforms, we found that the reference architecture had to be formulated on a high-level, to capture the heterogeneity of the systems. Similarly, an ecosystem-wide benchmark must consider this constraint as well.
(4) **Complex ecosystems.** Serverless platforms are, in most cases, not intended as standalone systems. Instead, they provide deep integrations with other cloud services, such as integrations with event sources. To comprehensively evaluate a serverless platform, the performance and implications of these integrations need to be taken into account.

(5) **Multi-tenancy and dynamic deployments.** The short-lived and ephemeral nature of serverless services enables cloud providers to dynamically schedule and consolidate the workloads on multiplexed resources. The performance of serverless platforms varies [13] due to co-located workloads and overall resource demands. These time- and location-related variances need to be considered by a sound serverless benchmarking methodology.

2.2 What do existing benchmarks provide?

Most existing work focuses on performance requirements (point 1 in Section 2.1), while the remaining four challenges receive limited attention. In the following, we summarize what existing FaaS-specific benchmarks primarily focus on:

(1) **Hardware resource microbenchmarking.** Classic microbenchmarking of underlying hardware resource is the most studied aspect; especially CPU performance and, to a lesser extent, memory, disk and network performance.
(2) **Startup latency.** The serverless-specific aspects of cold vs warm starts for varying configurations (e.g., different runtimes, function sizes, package sizes) receive quite some attention from industrial[3] and academic studies [2, 13]
(3) **Concurrency and elasticity.** Some studies investigate how well providers fulfill the promise of fast elasticity and discovered profound differences in scaling behavior [7].
(4) **Trigger latency.** Little work exists that examines the propagation delay of triggers and it is partially limited because specific trigger types are unavailable across multiple providers [5].

3 TOWARD A COMPREHENSIVE SERVERLESS BENCHMARK DESIGN

We describe the current design of a comprehensive serverless benchmark, as created by the authors and the SPEC-RG Cloud group.[4]

3.1 Goal and key insight

Our *goal* is to create a *comprehensive serverless benchmark*.

The *key insight* that started our design is an understanding of what elements should be part of a benchmark to enable it to be comprehensive. Based on our prior work of surveying existing serverless platforms [12], we mapped the lifecycle of a function execution to multiple aspects, grouped in three operational layers—the Resource Orchestration, Function Management, and Workflow Composition layers in Figure 1. We want to capture and characterize the interplay between the various components depicted in the figure. For example, besides the function runtime overhead itself, there are other aspects that influence the performance of the overall function execution: event-trigger propagation, function configuration, code propagation, and the overhead of function deployment.

[3] https://mikhail.io/serverless/coldstarts/big3/

[4] Established in 2011, the Cloud Group of the SPEC Research Group focuses on the general and specific performance issues associated with cloud operation, from traditional to new performance metrics, from workload characterization to modeling, from concepts to tools, from performance measurement processes to benchmarks. The work presented here is part of a large activity within this group, focusing on serverless and FaaS. The activity has started in 2017 and has resulted in several publications, which are available online and as open science artifacts. The group agrees with the publication of this article under the current co-authorship.

3.2 Scope of the serverless benchmark

Although the scope of this benchmark is limited to FaaS platforms, we explicitly extend its scope to include all major facets influencing performance in FaaS. Concretely, we aim to include the following:

(1) **Function runtime.** Although the runtime performance of functions has been evaluated by several studies (see Section 2), we do plan to also include this in this benchmark. The motivation for this is two-fold: (1) being able to compare our results to existing work, to help validate our approach; and (2) our exact set of target platforms and configurations have not been evaluated by prior work.
(2) **Event propagation.** Besides focusing on the function runtime, preliminary experiments (see Section 4) show that the propagation time of events from their source to the FaaS platform can vary widely. Since the performance of the event propagation affects the overall function execution overhead, we argue it is a key aspect to incorporate into the benchmark.
(3) **Cost.** Serverless computing is inherently about balancing the cost-performance trade-off. For this reason, a comprehensive serverless benchmark should not only take performance into account, but also the associated cost. During the preliminary results (Section 4) we found that both the performance and the cost can vary widely among the evaluated cloud providers.
(4) **Software flow.** A consequence of the granular nature of serverless functions is that larger numbers of functions are needed to represent a traditional monolithic application. Each of these must be deployed, upgraded, and scaled on a regular basis, which requires the function code to be transferred in and across data centers. With serverless, the performance of orchestrating this *software flow* can significantly impact the overall performance of the function execution.
(5) **Realistic Applications.** Although the benchmark will also consist of microbenchmarks, these typically do not consider the interplay between the components of a serverless platform. Moreover, it is non-trivial for cloud users to extrapolate the results of microbenchmarks to the performance of their full applications. We argue that there is a need to go beyond microbenchmarks and evaluate realistic applications.
(6) **Support for open-source platforms.** A consistent absence in most existing benchmarks is the lack of support for, and consequently, the evaluation of, open-source serverless platforms. Although a more complex benchmark is needed to manage the deployment, configuration, and cleanup of such platforms, exactly these open-source, self-deployed systems can provide us with a deeper insight into the performance of serverless architectures.

3.3 Overview of the preliminary design

The (preliminary) design of the benchmark is depicted in Figure 2. Based on the observation that each FaaS platform and each cloud require mostly similar components but use significantly different logic for their inter-operation, the design starts from the principle of *a small core to contain the main benchmark workflow, augmented with drivers for each platform.*

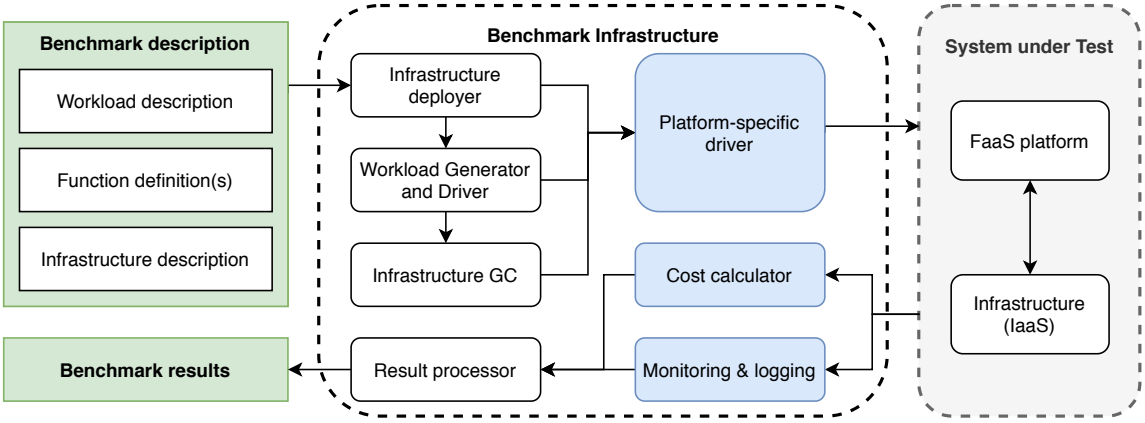

Figure 2: Overview of the serverless benchmark architecture. The rectangular green boxes indicate the static artifacts necessary for and generated by the benchmark, and the blue rounded boxes indicate the platform-specific components.

Each benchmark is represented by a *benchmark description*, which consists of a workload, function deployment, and infrastructure description. Based on this description, first the *infrastructure deployer* uses the *infrastructure description* and the *function definitions* to configure the cloud *infrastructure* and (optionally) orchestrate the deployment of a self-managed *FaaS platform*. Once the infrastructure is ready, the *workload generator and driver* pushes workload to the system according to the *workload description*. After the benchmark has been completed, the *benchmark GC* will ensure that the benchmark resources are pruned from the cloud environment.

The *monitoring & logging* component collects metrics during a benchmark run. This component is also platform-specific, as the way monitoring data is stored and retrieved differs widely between platforms. Alongside, the *cost calculator* retrieves and calculates the cost that has been incurred during the benchmark execution. Together the monitoring and cost data are post-processed by a *result processor* and stored as the final *benchmark results*.

3.4 Implementation

The development of the benchmark remains an ongoing process. However, we can already share highlights on the current status of the implementation of the benchmark.

We use Go for the framework, due to its focus on networking and its popularity within the (distributed) systems community. For most experiments, we use NodeJS or Python for the serverless functions, as they are widely supported by existing serverless platforms [10].

The field of serverless computing evolves rapidly, so—without updates—benchmark results become outdated quickly. For this reason, we focus on the implementation of reproducibility and workflow automation. This will allow us to routinely rerun the benchmark and maintain an overview of up-to-date results.

Our goal is to make the benchmark, experiments, and results open-source.[5] Moreover, beyond making the project open-source, we specifically aim for developer experience; it should be straightforward to deploy and run the benchmark, as well as easy to contribute to. With these aims, we hope to foster a long-lived, community-wide serverless benchmark.

3.5 Ideas for the longer-term future

Finally, there are several topics which we deem interesting yet out-of-scope for the near-future of the current project.

First, we expect that privacy and other data regulations, such as GDPR,[6] will increasingly affect the performance characteristics of cloud computing. For serverless computing specifically, the dynamic and ephemeral nature makes it an appropriate model for privacy-sensitive workloads, allowing platforms to schedule the execution on a granular level.

Second, as serverless computing becomes increasingly relevant for data-intensive workloads—e.g., in graph processing [9] and object storage [8]—additional experiments targeting these kinds workloads will be needed. A benchmark will need to evaluate the (possible) interplay between the storage and execution platforms, and explore how data-intensive serverless applications are designed.

4 PRELIMINARY RESULTS

We have run preliminary experiments before and during the development of our work-in-progress benchmark. The results of these experiments serve as an initial validation of the benchmark design.

We describe the results of two real-world experiments we have performed on closed- and open-source serverless platforms: (1) a basic evaluation of event propagation using a state-of-the-art message queue as proxy for a event management system in a cloud, and (2) an evaluation of the performance and cost of several FaaS platforms running the same workload.

4.1 Event propagation

The goal of the preliminary evaluation of the event propagation was to explore the performance difference between event management system configurations and to serve as a reference for the results of the different cloud providers.

[5] Due to the prototype state of the benchmark, it is currently closed source.

[6] https://eur-lex.europa.eu/eli/reg/2016/679/oj

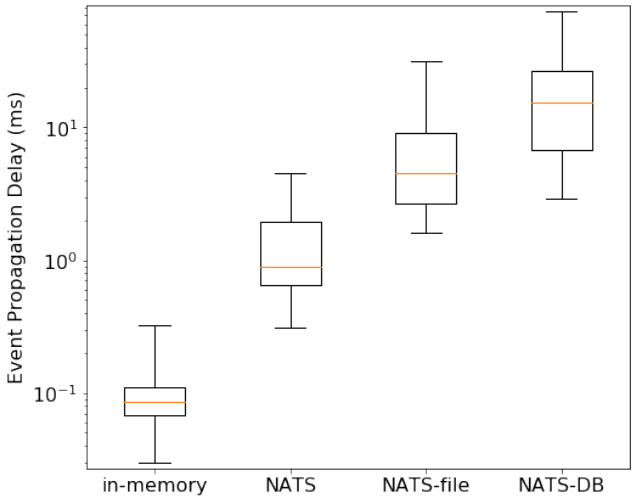

Figure 3: The impact of different event-queue configurations on the delay of event propagation.

For the event management system, we chose to evaluate configurations of an open-source, state-of-the-art message queue. These systems are typically used for event management throughout complex ecosystems (such as clouds), and can therefore be viewed as a reasonable proxy. We settled on evaluating configurations of NATS Streaming,[7] which is comparable in functionality and performance to other state-of-the-art alternatives, such as RabbitMQ and Kafka. The configurations we evaluate are the modes of persistence, which specify how the messages (or events) are persisted. We choose this parameter because it is unknown what persistence model is used within the major clouds, and it was technically feasible for a time-constrained preliminary experiment.

For this *real-world experiment*, we used a setup of a single driver VM (8 Intel Xeon CPUs, 32 GB RAM, 512 GB SSD) submitting a workload to a single VM (4 Intel Xeon CPUs, 16 GB RAM, 256 GB SSD) operating a NATS Streaming server. The VMs were deployed in the same datacenter, connected to each other by a 1 Gbit/s Ethernet link. The event propagation delay is the duration between the driver sending the message and the driver receiving the same message while subscribed to the message queue.

For the workload we used the 10-minute Chronos trace [6]. This workload trace is an ETL workload, which originates from an industrial process use case. It consists of submissions of a 3-task workflow, with on average 5.4 tasks being submitted per second with peaks of 22 events per second. For the event propagation experiment, we assumed that each task maps to one event.

Figure 3 shows the impact of the configurations of the message queue on the event propagation. As a baseline, *in-memory* skips the entire message queue and immediately returns the result to the subscriber. The *NATS* configuration is the default configuration of the system, which stores messages solely in-memory. *NATS-file* stores the messages on the local filesystem (SSD). The *NATS-DB* uses a SQL database—Postgres in this case—to persist the messages.

[7]https://github.com/nats-io/nats-streaming-server

In the context of serverless functions, the difference between these delays would likely have a significant impact on the performance (see Figure 4). Since there are numerous parameters to configure for these types of systems, these results hint that the internally-built event management systems in the major clouds will differ in performance.

4.2 Function runtime overhead

In the second experiment, our goal was to perform an initial evaluation of the overall runtime overhead of several serverless platforms. Although this type of experiment has been included in some existing benchmarks, we additionally focused on exploring the evaluation of an open-source FaaS platform and workflow orchestration systems.

We evaluated the serverless platforms of the three major cloud providers (Azure, AWS, and Google), and a state-of-the-art, open-source FaaS platform, called Fission.[8] As a baseline, we additionally include SimFaaS,[9] which is a simple FaaS simulator.

For the managed FaaS platform, we evaluated AWS Lambda, Google Cloud Functions, and Azure Functions. For each of the platforms, we used similar configurations (e.g., 128 MB RAM) and deployed the driver machine (1 CPU, 4 GB RAM) in the same region as the functions. For this experiment we again used the Chronos workload, described in the event propagation experiment. We used the function execution runtimes reported by the FaaS platforms themselves to eliminate network latency impact—although this does require us to trust the self-reported metrics of the platforms. The costs of the workloads are calculated post facto, ignoring any promotional pricing. The costs are separated into costs directly attributable to the FaaS function (*function execution costs*) and costs related to the orchestration of the Chronos workflows (*workflow orchestration costs*). For the self-deployed platform (Fission), we calculated the costs (*self-management costs*) by amortizing the total machine costs incurred during the experiment over the Chronos runs. The runtime overhead is calculated by subtracting the minimum task runtime from the actual time spend on the executing task. We deployed Fission on a cluster of 3 small VMs (1 CPU, 3.75 GB RAM), and SimFaaS on a larger VM (1 CPU, 8 GB RAM).

In Figure 4 the boxplots show that from the managed serverless platforms, Google has the lowest overhead in this experiment, achieving overheads between 50 ms and 18 ms. AWS Lambda and Azure functions have slightly higher overheads, but lower performance variability.

Despite the preliminary nature of these results, we observe that there are cases in which the performance of the serverless platforms of major clouds can differ wildly for the same workload. The costs show similar variability in the costs per platform (see Figure 5). The results also highlight the challenge of comparing self-deployed serverless platforms to managed serverless platforms; although Fission technically costs significantly less than the managed platforms for equal performance, it does not account for the cost of operating these self-deployed platforms. We consider exploring how to compare these approaches fairly a key part of our future work.

[8]https://github.com/fission/fission
[9]https://github.com/erwinvaneyk/simfaas

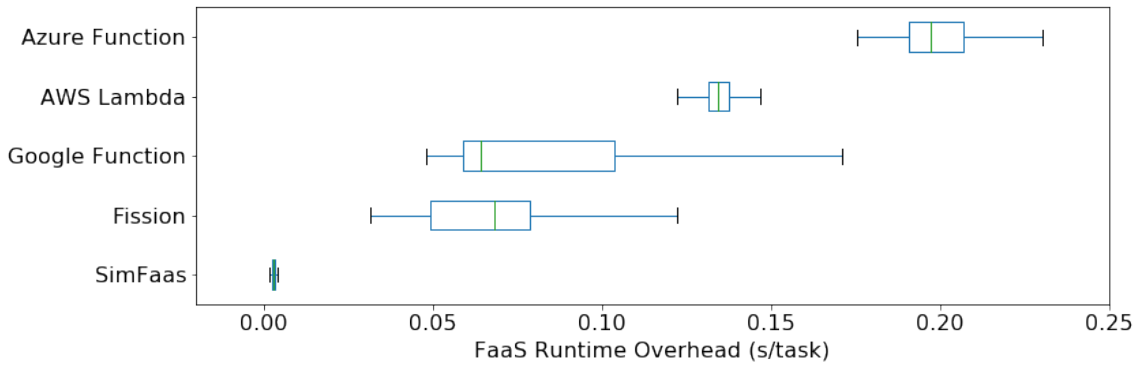

Figure 4: The performance overhead of FaaS platforms when executing the Chronos workload.

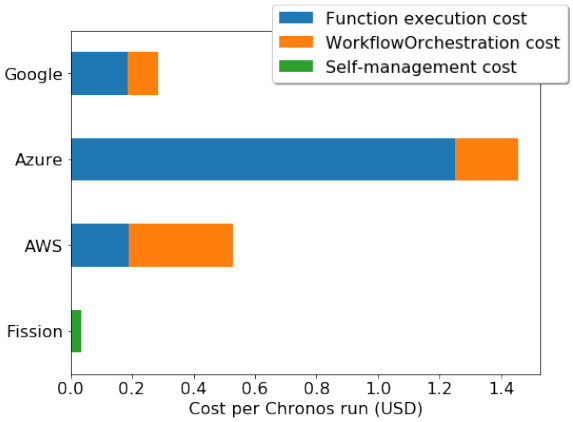

Figure 5: The cost of running the same workload, Chronos (see text), on different serverless platforms.

5 CONCLUSION

The increasingly popular serverless industry, and especially its function-based approach (FaaS), is based on emerging technology. Learning from history, we argued for the need to develop comprehensive benchmarking tools for serverless and FaaS technology.

We envisioned a benchmark designed with a structured, principled approach, aiming to evaluate the performance of the typical components of serverless platforms. Going beyond microbenchmarks, the benchmark evaluates the broader serverless ecosystem, the events that trigger functions, the overhead introduced by fetching function code, and how realistic serverless applications use the platforms. We also presented preliminary, real-world, experimental results across several closed- and open-source serverless platforms.

Our future work consists broadly of two stages. The first stage is to complete the benchmark implementation and perform a comprehensive investigation of managed serverless platforms. In the second stage, we will iterate on the existing benchmark, further analyze self-deployed serverless platforms, and evaluate the performance of more complex applications in more data-intensive domains—such as machine learning, and graph processing.

Finally, we want to end this vision with a call to action: this benchmark already is a collaborative effort of universities across the globe, and we invite the community to join this effort.

ACKNOWLEDGMENTS

The work presented in this article has benefited from discussions within the SPEC-RG Cloud Group, and further in the Cloud Control Workshop series organized by Erik Elmroth and his team.

REFERENCES

[1] 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 *ICPE WS 2018*. 21–24.

[2] Kamil Figiela, Adam Gajek, Adam Zima, Beata Obrok, and Maciej Malawski. 2018. Performance evaluation of heterogeneous cloud functions. *Concurrency and Computation: Practice and Experience* 30, 23 (2018).

[3] Joseph M. Hellerstein, Jose M. Faleiro, Joseph Gonzalez, Johann Schleier-Smith, Vikram Sreekanti, Alexey Tumanov, and Chenggang Wu. 2019. Serverless Computing: One Step Forward, Two Steps Back. In *9th Biennial Conference on Innovative Data Systems Research (CIDR)*.

[4] Eric Jonas et al. 2019. Cloud Programming Simplified: A Berkeley View on Serverless Computing. *CoRR* abs/1902.03383 (2019). arXiv:1902.03383 http://arxiv.org/abs/1902.03383

[5] Hyungro Lee, Kumar Satyam, and Geoffrey C Fox. 2018. Evaluation of Production Serverless Computing Environments. In *Third International Workshop on Serverless Computing (WoSC)*.

[6] Shenjun Ma, Alexey Ilyushkin, Alexander Stegehuis, and Alexandru Iosup. 2017. Ananke: A q-learning-based portfolio scheduler for complex industrial workflows. In *2017 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE.

[7] G. McGrath and P. R. Brenner. 2017. Serverless Computing: Design, Implementation, and Performance. In *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. 405–410.

[8] Josep Sampé, Marc Sánchez-Artigas, Pedro García-López, and Gerard París. 2017. Data-driven serverless functions for object storage. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*. 121–133.

[9] Lucian Toader, Alexandru Uta, Ahmed Musaafir, and Alexandru Iosup. 2019. Graphless: Toward serverless graph processing. In *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE, 66–73.

[10] Erwin van Eyk, Johannes Grohmann, Simon Eismann, André Bauer, Laurens Versluis, Lucian Toader, Norbert Schmitt, Nikolas Herbst, Cristina Abad, and Alexandru Iosup. 2019. The SPEC-RG Reference Architecture for FaaS: From Microservices and Containers to Serverless Platforms. *IEEE Internet Comp.* (2019).

[11] Erwin van Eyk, Alexandru Iosup, Simon Seif, and Markus Thömmes. 2017. The SPEC cloud group's research vision on FaaS and serverless architectures. In *Proceedings of the 2nd International Workshop on Serverless Computing*. 1–4.

[12] Erwin van Eyk, Lucian Toader, Sacheendra Talluri, Laurens Versluis, Alexandru Uță, and Alexandru Iosup. 2018. Serverless is more: From PaaS to present cloud computing. *IEEE Internet Computing* 22, 5 (2018), 8–17.

[13] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. 2018. Peeking Behind the Curtains of Serverless Platforms. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 133–146. https://www.usenix.org/conference/atc18/presentation/wang-liang

[14] CNCF Serverless WG. 2018. CNCF WG-Serverless Whitepaper v1.0. https://github.com/cncf/wg-serverless/blob/master/whitepaper/cncf_serverless_whitepaper_v1.0.pdf. Accessed 2018-07-29.