SeBS 2.0: Keeping up with the Clouds

Marcin Copik
ETH Zurich
Switzerland
marcin.copik@inf.ethz.ch

Alexandru Calotoiu
ETH Zurich
Switzerland
alexandru.calotoiu@inf.ethz.ch

Torsten Hoefler
ETH Zurich
Switzerland
htor@inf.ethz.ch

Abstract

Three years ago, SeBS, the serverless benchmarking suite, was introduced to address the need for an automatic, representative, and easy-to-use benchmarking framework for FaaS applications. SeBS has been widely adopted in research projects and has evolved to incorporate new features addressing the changing landscape of FaaS platforms. As serverless workflows and services continue to grow in size and complexity, there is an ongoing need to support emerging application classes. In this paper, we outline both the progress made and the roadmap for supporting new workloads and frameworks, while identifying upcoming trends and paradigm shifts to provide researchers with reliable and reproducible benchmarks. The serverless community needs an open and portable benchmarking framework to drive future progress, and SeBS aims to fulfill this crucial role.

CCS Concepts

• **Computer systems organization** → **Cloud computing**; • **Software and its engineering** → **Cloud computing**; • **General and reference** → *Metrics*; *Performance*; *Evaluation*; *Measurement*.

Keywords

Serverless, Function-as-a-Service, FaaS, Benchmark

ACM Reference Format:
Marcin Copik, Alexandru Calotoiu, and Torsten Hoefler. 2025. SeBS 2.0: Keeping up with the Clouds. In *The 3rd Workshop on SErverless Systems, Applications and MEthodologies (SESAME' 25), March 30-April 3, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3721465.3721867

SeBS implementation: https://github.com/spcl/serverless-benchmarks

1 Past

The Serverless Benchmark Suite (SeBS) was introduced in 2020 as a comprehensive framework for evaluating Function-as-a-Service (FaaS) platforms [5], supporting three major commercial serverless systems: AWS Lambda, Azure Functions, and Google Cloud Functions. In addition to providing a collection of real-world serverless functions, SeBS offered an integrated solution for automatic benchmarking: the framework allows users to automatically build deployment artifacts of selected functions, deploy them to the selected cloud platform, invoke functions, and conduct performance analysis. This approach helps to ensure **reproducible** experiments, as deployment artifacts are consistently built following standardized recipes, regardless of the packaging requirements or cloud platform specifications. Among the core design principles were **modularity** and focus on the **cloud-agnostic** approach. The modular architecture allows seamless integration of new serverless systems, benchmarks, and programming languages through a plugin-like system. By implementing cloud-agnostic benchmarks with a unified API that abstracts away provider-specific details, SeBS enables fair comparisons across different serverless platforms.

In recent years, the serverless computing landscape has changed significantly, with new types of workloads, a focus on data movement, and increased complexity of serverless applications. To maintain relevance and continue serving the research community effectively, SeBS needs to evolve to address these emerging trends.

2 Present

Since its initial benchmark release, SeBS has significantly expanded its capabilities, platform support, and measurement framework. In this section, we summarize the major updates and current work in progress, focusing on the new types of experiments and measurements that these changes enabled.

Languages. SeBS has been extended to support two new programming languages: C++ and Java. These additions allow benchmarking across different cold start scenarios: from the minimal overhead of compiled languages in C++ to Java's heavy runtime, which prompted cloud providers to introduce function snapshotting. Additionally, C++ support enables future benchmarking of high-performance and GPU-accelerated functions, while Java support will facilitate the inclusion of microservice benchmarks like TeaStore.

Deployment. SeBS now supports a wider range of serverless platforms, including OpenWhisk, Fission, and Knative. While initially, all functions were deployed as ZIP code packages to provide a shared and fair evaluation baseline among cloud providers, SeBS now includes container support. This allows users to circumvent code size limits and evaluate how different deployment methods affect performance. Containers are now the default deployment method for both Knative and OpenWhisk. Finally, benchmark functions have been adapted to support the `aarch64` architecture, reflecting the growing adoption of ARM architecture in clouds.

Cloud Services. The initial SeBS release provided a unified interface for object storage, where each benchmark configuration only has to define the data that needs to be uploaded to the cloud. SeBS has now expanded to support additional cloud service types, specifically NoSQL databases and queues. For NoSQL, SeBS implements a unified interface of key-value storage with primary and sorting keys, using underneath DynamoDB on AWS, CosmosDB

SESAME' 25, Rotterdam, Netherlands

ACM ISBN 979-8-4007-1557-0/2025/03
https://doi.org/10.1145/3721465.3721867

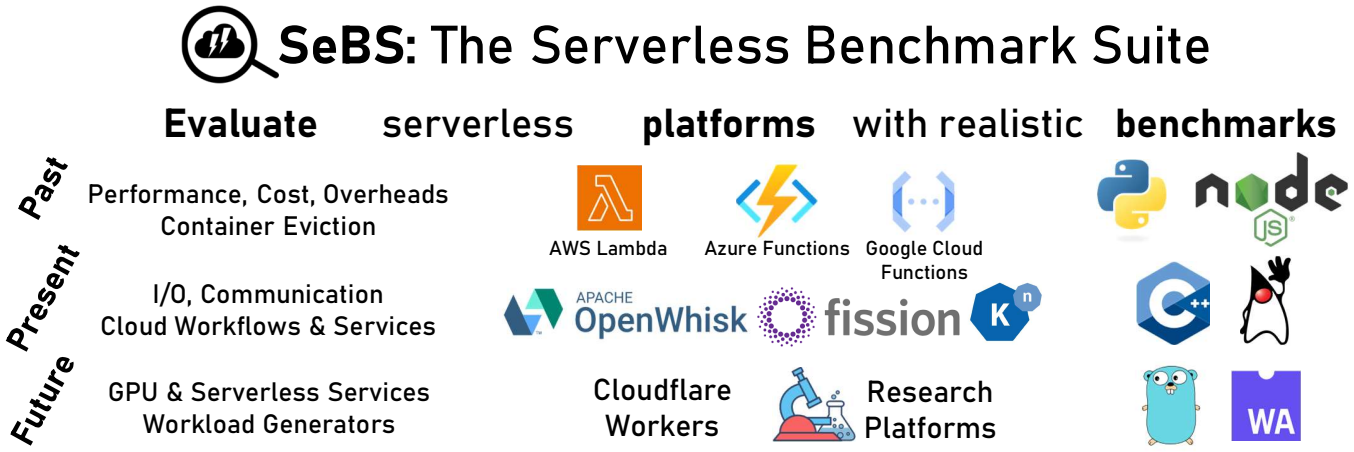

Figure 1: Past, present, and proposed future of SeBS: addressing the ever-changing serverless with new features and benchmarks.

on Azure, and Firestore in Datastore mode on Google Cloud. On open-source platforms, it can deploy ScyllaDB with its DynamoDB adapter to provide similar semantics. This integration of serverless key-value storage enables the representation of workloads where dynamic components of cloud services are offloaded to serverless functions.

Similarly, we have been working on supporting cloud queues in SeBS to enable data exchange and invoking functions by pushing new messages to the queue. This new trigger capability allows us to include dynamic applications that do not fit into the static form of a workflow, such as microservices. With queues triggering function invocations, these applications can be built in a cloud-native fashion while maintaining reliable execution. Furthermore, queues can guarantee the FIFO ordering of invocations, which is needed in applications like serverless ZooKeeper [4]. Finally, queues will allow for future support of asynchronous function invocations by providing a channel to return execution results back to the user. SeBS currently supports SQS on AWS, StorageQueue on Azure, and PubSub on Google Cloud, with plans to incorporate RabbitMQ for open-source platforms in the future.

Benchmarks. Contributions to SeBS include new benchmarks and applications that cover a broader spectrum of serverless use cases and scenarios. These additions include website backends: a web application that uses NoSQL storage underneath, utility functions performing optical character recognition with Tesseract and PDF generation with Puppeteer, and an ML inference performing image captioning. These benchmarks provide greater diversity in common workloads that use serverless as a scalable backend for web and cloud services.

Experiments. Additionally, SeBS has been extended with new experiments to measure the latency and bandwidth of various serverless communication modes, including cloud storage, in-memory caches, and direct communication through NAT hole punching. These experiments provide crucial performance data needed to understand how the reduced I/O performance impacts data-intensive workloads, such as data analytics, high-performance computing, and machine learning training [3, 15].

Workflows. A significant enhancement to SeBS is the inclusion of a whole new category of serverless applications: workflows [13]. Serverless workflows are built as a graph of functions with standard control-flow operations, allowing to represent more complex computing pipelines and even entire applications. SeBS now provides a unified and cloud-agnostic workflow specification model that is translated into a vendor-specific representation for AWS Step Functions, Azure Durable Functions, and Google Cloud Workflow. Experiments measure the latency, cost, and scalability of serverless workflows, using six different workflows that span web applications, machine learning, data analytics, and high-performance computing.

3 Future

We will continue development of SeBS by supporting more language runtimes, such as Golang and WebAssembly (WASM), and adding serverless new platforms, e.g., funcX [2] or rFaaS [6]. To accommodate ongoing changes in serverless and meet the new needs of the research community, we propose five major extensions that will increase the usability of the benchmark suite and future-proof it for new classes of workloads.

Workload Generators. With the increasing availability of serverless traces from various cloud providers, researchers can now benchmark serverless applications using realistic workloads. We plan to integrate trace-driven execution alongside standard invocation policies in SeBS, and we believe that it might become the default invocation method in the future. Since most traces are too large in scope and size, we will rely on existing approaches that provide downscaling of invocation rates [11, 14].

Bring-Your-Own-Function. We plan to extend support for benchmarking custom applications by allowing users to provide their own function code, dependencies, and input. This will enable the execution of all SeBS experiments, letting users benefit from fully automated testing while evaluating their own applications.

Profiling Serverless observability is another example of vendor lock-in, as functions typically depend on cloud services to locate performance bottlenecks and estimate the impact of external services. To understand the end-to-end latency of large serverless applications, it is necessary to combine profiling data from different functions to create a single application graph. We plan to include an existing FaaS profiler to analyze performance at the level of functions and whole applications [17]. Since the profiler does not offload tracing tasks to an existing cloud service, such as X-Ray on AWS, it can also be used with open-source and research platforms.

GPU Benchmarks. Accelerators are becoming increasingly important, with machine learning inference being one particular workload that benefits the most. However, practical deployments of GPU-accelerated functions run into many performance issues of cold starts and device sharing [7, 16]. We plan to enhance SeBS with GPU functions, focusing on common machine learning inference kernels and applications from scientific computing. Furthermore, we want to focus on experiments targeting GPU-specific challenges, such as the impact of co-location and overheads of checkpointing.

Serverless Services. Serverless applications have already been noticed as an important workload for benchmarking in FaaS [1, 9, 12]. We plan to include new microservice-style workloads [8, 10] to test the full spectrum of cloud services in a single benchmark. This will support both complete invocations and more granular benchmarking of individual system components. Furthermore, comparing with already supported workflows will highlight the strengths and weaknesses of both programming models.

4 Conclusions

SeBS has evolved significantly since its inception, and the ongoing work on hardware acceleration, programming models, and benchmark diversity addresses the new trends in serverless development. With the proposed changes, we expect the tool to make serverless research more productive by eliminating the custom and often manual work needed to prepare experiments. We anticipate that SeBS will remain a valuable tool for both researchers and practitioners in the serverless computing community.

Acknowledgments

This work would not have been possible without the contributions of our students and collaborators: Laurent Brandner, Nico Graf, Sascha Kehrli, Prajin Khadka, Abhishek Kumar, Oana Rosca, Larissa Schmid, Mahla Sharifi, Malte Wächter, and Paweł Żuk. Abhishek Kumar and Prajin Khadka have been supported through the Google Summer of Code program. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 program (grant agreement PSAP, No. 101002047), and from the Swiss State Secretariat for Education, Research and Innovation (SERI) under the SwissTwins project. We thank Amazon Web Services for supporting the development of SeBS projects with credits through the AWS Cloud Credit for Research, and Google Cloud Platform through the Google Cloud Research Credits program with the award GCP19980904.

References

[1] [Online; accessed 6. March 2025]. vSwarm - Serverless Benchmarking Suite. https://github.com/ease-lab/vSwarm.

[2] Ryan Chard, Yadu Babuji, Zhuozhao Li, Tyler Skluzacek, Anna Woodard, Ben Blaiszik, Ian Foster, and Kyle Chard. 2020. funcX: A Federated Function Serving Fabric for Science. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing* (Stockholm, Sweden) *(HPDC '20)*. Association for Computing Machinery, New York, NY, USA, 65–76. doi:10.1145/3369583.3392683

[3] Marcin Copik, Roman Böhringer, Alexandru Calotoiu, and Torsten Hoefler. 2023. FMI: Fast and Cheap Message Passing for Serverless Functions. In *Proceedings of the 37th International Conference on Supercomputing* (Orlando, FL, USA) *(ICS '23)*. Association for Computing Machinery, New York, NY, USA, 373–385. doi:10.1145/3577193.3593718

[4] Marcin Copik, Alexandru Calotoiu, Pengyu Zhou, Konstantin Taranov, and Torsten Hoefler. 2024. FaaSKeeper: Learning from Building Serverless Services with ZooKeeper as an Example. In *Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing* (Pisa, Italy) *(HPDC '24)*. Association for Computing Machinery, New York, NY, USA, 94–108. doi:10.1145/3625549.3658661

[5] Marcin Copik, Grzegorz Kwasniewski, Maciej Besta, Michal Podstawski, and Torsten Hoefler. 2021. SeBS: A Serverless Benchmark Suite for Function-as-a-Service Computing. In *Proceedings of the 22nd International Middleware Conference* (Québec city, Canada) *(Middleware '21)*. Association for Computing Machinery, New York, NY, USA, 64–78. doi:10.1145/3464298.3476133

[6] Marcin Copik, Konstantin Taranov, Alexandru Calotoiu, and Torsten Hoefler. 2023. rFaaS: Enabling High Performance Serverless with RDMA and Leases. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 897–907. doi:10.1109/IPDPS54959.2023.00094

[7] Henrique Fingler, Zhiting Zhu, Esther Yoon, Zhipeng Jia, Emmett Witchel, and Christopher J Rossbach. 2022. DGSF: Disaggregated GPUs for serverless functions. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 739–750.

[8] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, et al. 2019. An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 3–18.

[9] Martin Grambow, Tobias Pfandzelter, Luk Burchard, Carsten Schubert, Max Zhao, and David Bermbach. 2021. BeFaaS: An Application-Centric Benchmarking Framework for FaaS Platforms. In *2021 IEEE International Conference on Cloud Engineering (IC2E)*. 1–8. doi:10.1109/IC2E52221.2021.00014

[10] Zhipeng Jia and Emmett Witchel. 2021. Nightcore: efficient and scalable serverless computing for latency-sensitive, interactive microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 152–166.

[11] Christos Katsakioris, Chloe Alverti, Konstantinos Nikas, Dimitrios Siakavaras, Stratos Psomadakis, and Nectarios Koziris. 2024. FaaSRail: Employing Real Workloads to Generate Representative Load for Serverless Research. In *Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing* (Pisa, Italy) *(HPDC '24)*. Association for Computing Machinery, New York, NY, USA, 214–226. doi:10.1145/3625549.3658684

[12] Joel Scheuner, Simon Eismann, Sacheendra Talluri, Erwin van Eyk, Cristina Abad, Philipp Leitner, and Alexandru Iosup. 2022. Let's Trace It: Fine-Grained Serverless Benchmarking using Synchronous and Asynchronous Orchestrated Applications. arXiv:2205.07696 [cs.DC] https://arxiv.org/abs/2205.07696

[13] Larrisa Schmid, Marcin Copik, Alexandru Calotoiu, Laurin Brandner, Anne Koziolek, and Torsten Hoefler. 2025. SeBS-Flow: Benchmarking Serverless Cloud Function Workflows. In *Proceedings of the Twentieth European Conference on Computer Systems* (Rotterdam, Netherlands) *(EuroSys'25)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3689031.3717465

[14] Dmitrii Ustiugov, Dohyun Park, Lazar Cvetković, Mihajlo Djokic, Hongyu Hè, Boris Grot, and Ana Klimovic. 2023. Enabling In-Vitro Serverless Systems Research. In *Proceedings of the 4th Workshop on Resource Disaggregation and Serverless* (Koblenz, Germany) *(WORDS '23)*. Association for Computing Machinery, New York, NY, USA, 1–7. doi:10.1145/3605181.3626191

[15] Michael Wawrzoniak, Ingo Müller, Rodrigo Fraga Barcelos Paulus Bruno, and Gustavo Alonso. 2021. Boxer: Data analytics on network-enabled serverless platforms. In *11th annual conference on innovative data systems research (CIDR 2021)*. ETH Zurich.

[16] Hao Wu, Yue Yu, Junxiao Deng, Shadi Ibrahim, Song Wu, Hao Fan, Ziyue Cheng, and Hai Jin. 2024. {StreamBox}: A Lightweight {GPU}{SandBox} for Serverless Inference Workflow. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 59–73.

[17] Malte Wächter, Marcin Copik, Alexandru Calotoiu, and Torsten Hoefler. 2022. FaaS-Profiler: Serverless Tracing and Profiling. https://www.serverlesscomputing.org/wosc8/demos/d2. Demo at 8th Workshop on Serverless Computing.