Thesis for the Degree of Doctor of Philosophy

Performance Evaluation of Serverless Applications and Infrastructures

Joel Scheuner

Division of Interaction Design and Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden, 2022

Performance Evaluation of Serverless Applications and Infrastructures

JOEL SCHEUNER

ISBN 978-91-7905-677-3
Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 5143.
ISSN 0346-718X
Electronic: `https://research.chalmers.se/en/publication/531473`

Technical Report No 224D
Department of Computer Science & Engineering
Division of Interaction Design and Software Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden

Cover picture: sdecoret / stock.adobe.com

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2022.

~100% of benchmarks are wrong.

The energy needed to refute benchmarks is orders of magnitude bigger than to run them (so, no one does)

– Brendan Gregg, Senior Performance Architect

Abstract

Context Cloud computing has become the de facto standard for deploying modern web-based software systems, which makes its performance crucial to the efficient functioning of many applications. However, the unabated growth of established cloud services, such as Infrastructure-as-a-Service (IaaS), and the emergence of new serverless services, such as Function-as-a-Service (FaaS), has led to an unprecedented diversity of cloud services with different performance characteristics. Measuring these characteristics is difficult in dynamic cloud environments due to performance variability in large-scale distributed systems with limited observability.

Objective This thesis aims to enable reproducible performance evaluation of serverless applications and their underlying cloud infrastructure.

Method A combination of literature review and empirical research established a consolidated view on serverless applications and their performance. New solutions were developed through engineering research and used to conduct performance benchmarking field experiments in cloud environments.

Findings The review of 112 FaaS performance studies from academic and industrial sources found a strong focus on a single cloud platform using artificial micro-benchmarks and discovered that most studies do not follow reproducibility principles on cloud experimentation. Characterizing 89 serverless applications revealed that they are most commonly used for short-running tasks with low data volume and bursty workloads. A novel trace-based serverless application benchmark shows that external service calls often dominate the median end-to-end latency and cause long tail latency. The latency breakdown analysis further identifies performance challenges of serverless applications, such as long delays through asynchronous function triggers, substantial runtime initialization for coldstarts, increased performance variability under bursty workloads, and heavily provider-dependent performance characteristics. The evaluation of different cloud benchmarking methodologies has shown that only selected micro-benchmarks are suitable for estimating application performance, performance variability depends on the resource type, and batch testing on the same instance with repetitions should be used for reliable performance testing.

Conclusions The insights of this thesis can guide practitioners in building performance-optimized serverless applications and researchers in reproducibly evaluating cloud performance using suitable execution methodologies and different benchmark types.

Keywords

Cloud Computing, Performance, Benchmarking, Serverless, Function-as-a-Service, Infrastructure-as-a-Service

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor and long-term mentor Philipp Leitner for his advice, trust, and collaboration during the over 8 year long journey starting from my undergraduate studies towards this PhD thesis. Philipp fostered my growth in becoming an independent researcher through his right balance between guidance and autonomy. I also thank my co-supervisor Jan-Philipp Steghöfer for his valuable detailed feedback. Further, I am grateful for the freedom my examiner Robert Feldt gave me in conducting my research.

I wish to thank my amazing collaborators for all contributions, feedback, and fruitful discussions in shaping my research. I am extremely grateful to Christoph Laaber from our extended ICET-lab in Zurich, Alexandru Iosup, Cristina Abad, Simon Eismann, Sacheendra Talluri, Erwin van Eyk, and others from the international SPEC-RG Cloud group, and to my excellent master students Rui Deng, Marcus Bertilsson, Oskar Grönqvist, Henrik Tao, and Henrik Lagergren.

Thank you to all my colleagues at the Interaction Design and Software Engineering Division for shaping a great work environment and engaging in many fun social activities. My gratitude also includes the s.e.a.l. alumni in Zurich and WASP colleagues at Chalmers and from other Swedish universities. Special thanks go to my office friends in Kuggen Linda, Hamdy, Razan, Georgios, and Peter.

I appreciate the support and visits of my family and friends from Switzerland and abroad. I am eternally grateful to my wife Yao for her positivity ☼, love, care, and continuos encouragement.

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and by the Swedish Research Council VR under grant number 2018-04127 (Developer-Targeted Performance Engineering for Immersed Release and Software Engineers).

List of Publications

Appended Publications

This thesis is based on the following publications appended in Chapters α to θ:

[α] **J. Scheuner**, P. Leitner
"Function-as-a-Service Performance Evaluation: A Multivocal Literature Review"
Journal of Systems and Software (JSS), 2020.
doi:10.1016/j.jss.2020.110708
Chapter α

[β] S. Eismann, **J. Scheuner**, E. v. Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, A. Iosup
"The State of Serverless Applications: Collection, Characterization, and Community Consensus"
IEEE Transactions on Software Engineering (TSE), 2021.
doi:10.1109/TSE.2021.3113940
Chapter β

[γ] **J. Scheuner**, S. Eismann, S. Talluri, E. v. Eyk, C. L. Abad, A. Iosup, P. Leitner
"Let's Trace It: Fine-Grained Serverless Benchmarking for Synchronous and Asynchronous Applications"
Under submission to a journal.
Chapter γ

[δ] **J. Scheuner**, R Deng, JP. Steghöfer, P. Leitner
"CrossFit: Fine-grained Benchmarking of Serverless Application Performance across Cloud Providers"
Under submission to a conference.
Chapter δ

[ε] **J. Scheuner**, M. Bertilsson, O. Grönqvist, H. Tao, H. Lagergren, JP. Steghöfer, P. Leitner
"TriggerBench: A Performance Benchmark for Serverless Function Triggers"
Proceedings of the 10th IEEE International Conference on Cloud Engineering (IC2E), 2022 (to appear as short paper).
Chapter ε

[ζ] **J. Scheuner**, P. Leitner
"A Cloud Benchmark Suite Combining Micro and Applications Benchmarks"
Companion of the 9th ACM/SPEC International Conference on Performance Engineering (ICPE): 4th Workshop on Quality-Aware DevOps (QUDOS), 2018.
doi:10.1145/3185768.3186286
Chapter ζ

[η] **J. Scheuner**, P. Leitner
"Estimating Cloud Application Performance Based on Micro-Benchmark Profiling"
Proceedings of the 11th IEEE International Conference on Cloud Computing (CLOUD), 2018.
doi:10.1109/CLOUD.2018.00019
Chapter η

[θ] C. Laaber, **J. Scheuner**, P. Leitner
"Software Microbenchmarking in the Cloud. How bad is it really?"
Empirical Software Engineering (EMSE), 2019.
doi:10.1007/s10664-019-09681-1
Chapter θ

Other Publications

The following publications were published before or during my PhD studies. However, they are not appended to this thesis because they were published before my PhD studies [a-f], unrelated to the thesis, or overlapping with the thesis content.

An updated list of all my publications is available on my website[1] and Google Scholar profile[2].

[a] **J. Scheuner**, P. Leitner, J. Cito, H. Gall
"Cloud WorkBench – Infrastructure-as-Code Based Cloud Benchmarking"
Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2014.
doi:10.1109/CloudCom.2014.98

[b] **J. Scheuner**, P. Leitner, J. Cito, H. Gall
"Cloud WorkBench: Benchmarking IaaS Providers based on Infrastructure-as-Code"
Companion of the 24th International Conference on World Wide Web (WWW Demo), 2015.
doi:10.1145/2740908.2742833

[c] P. Leitner, **J. Scheuner**
"Bursting With Possibilities – an Empirical Study of Credit-Based Bursting Cloud Instance Types"
Proceedings of the 8th IEEE/ACM International Conference on Utility and Cloud Computing (UCC), 2015.
doi:10.1109/UCC.2015.39

[d] **J. Scheuner**, G. Mazlami, D. Schöni, S. Stephan, A. De Carli, T. Bocek, B. Stiller
"Probr – A Generic and Passive WiFi Tracking System"
Proceedings of the 41st IEEE Conference on Local Computer Networks (LCN), 2016.
doi:10.1109/LCN.2016.30

[e] **J. Scheuner**, G. Mazlami, D. Schöni, S. Stephan, A. De Carli, T. Bocek, B. Stiller
"Probr Demonstration – Visualizing Passive WiFi Data"
Proceedings of the 41st IEEE Conference on Local Computer Networks (LCN Demo), 2016. ***Best Demo Award LCN'16****.*
doi:10.1109/LCN.2016.135

[f] **J. Scheuner**, P. Leitner, J. Cito, H. Gall
"An Approach and Case Study of Cloud Instance Type Selection for Multi-Tier Web Applications"
Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2017.
doi:10.1109/CCGRID.2017.12

[1] `https://joelscheuner.com/`
[2] `https://scholar.google.com/citations?user=EfD_tnUAAAAJ`

[g] **J. Scheuner**, P. Leitner
"Performance Benchmarking of Infrastructure-as-a-Service (IaaS) Clouds with Cloud WorkBench"
Companion of the 10th ACM/SPEC International Conference on Performance Engineering (ICPE Tutorial), 2019.
doi:10.1145/3302541.3310294

[h] **J. Scheuner**, P. Leitner
"Transpiling Applications into Optimized Serverless Orchestrations"
*Proceedings of the 4th IEEE FAS*W: 2nd Workshop on Hot Topics in Cloud Computing Performance (HotCloudPerf) at ICAC/SASO, 2019.*
doi:10.1109/FAS-W.2019.00031

[i] **J. Scheuner**, P. Leitner
"Tutorial – Performance Benchmarking of Infrastructure-as-a-Service (IaaS) Clouds with Cloud WorkBench"
Proceedings of the 4th IEEE International Workshops on Foundations and Applications of Self Systems (FAS*W) at ICAC/SASO, 2019.*
doi:10.1109/FAS-W.2019.00070

[j] E. v. Eyk, **J. Scheuner**, S. Eismann, C. L. Abad, A. Iosup
"Beyond Microbenchmarks: The SPEC-RG Vision for a Comprehensive Serverless Benchmark"
Companion of the 11th ACM/SPEC International Conference on Performance Engineering (ICPE): 3rd Workshop on Hot Topics in Cloud Computing Performance (HotCloudPerf), 2020.
doi:10.1145/3375555.3384381

[k] S. Eismann, **J. Scheuner**, E. v. Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L Abad, A. Iosup
"A Review of Serverless Use Cases and their Characteristics"
SPEC-RG-2020-5 Technical Report, 2020. Endorsed by SPEC-RG but not formally peer reviewed.
doi:10.48550/arxiv.2008.11110

[l] S. Eismann, **J. Scheuner**, E. v. Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, A. Iosup
"Serverless Applications: Why, When, and How?"
IEEE Software, 2021.
doi:10.1109/MS.2020.3023302

[m] JC. Carver, B. Penzenstadler, **J. Scheuner**, M. Staron
"(Research) Insights for Serverless Application Engineering"
IEEE Software, 2021.
doi:10.1109/MS.2020.3028659

[n] T. Schirmer, **J. Scheuner**, T. Pfandzelter, D. Bermbach
"FUSIONIZE: Improving Serverless Application Performance through Feedback-Driven Function Fusion"
Proceedings of the 10th IEEE International Conference on Cloud Engineering (IC2E), 2022 (to appear).
doi:10.48550/arxiv.2204.11533

Personal Contribution

Following the Contributor Roles Taxonomy (CRediT)[3] as summarized in Table 1:

I was the main contributor for all papers except for Papers β and θ.

In Paper β, I was involved in *Conceptualization*, *Data curation*, *Investigation*, *Methodology*, *Validation*, *Writing – Original Draft*, *Writing – Review & Editing*, and *Supervision* of a bachelor thesis that seeded the collection of open-source applications. *Investigation* and *Data curation* was split equally among the authors. The first author contributed *Visualization* and lead the *Writing – Original Draft* based on the underlying technical report SPEC-RG-2020-5 [k], where *Writing – Original Draft* was largely split equally among the authors of this SPEC-RG Cloud[4] project.

In Paper γ, I led an international research project in the SPEC-RG Cloud group[4]. I built the core of the *Software* and coordinated with my collaborators and research assistants to integrate a suite of applications. I conducted all experiments and most of the data analysis where I received support in finalizing and improving visualizations. I wrote the majority of the publication and my collaborators contributed individual paragraphs and figures, provided continuous feedback, and supported *Writing – Review & Editing.*

In Paper δ, I was the main contributor for *Conceptualization*, *Methodology*, *Formal analysis*, *Writing – Original Draft*, *Writing – Review & Editing*, and *Visualization.* A master student implemented the *Software* based on Paper γ and collected the data under my *Supervision* but I re-wrote parts of the analysis and the entire publication from scratch.

In Paper ε, the *Software* was originally developed based on Paper γ in two master thesis projects under my *Supervision* but I re-wrote core parts of the *Software* to integrate the projects. I did the remaining work from scratch including experimentation, data analysis, and writing.

In Paper θ, I was involved in *Conceptualization*, *Investigation*, *Methodology*, *Software*, *Validation*, and *Writing – Review & Editing.* I contributed to the design and implementation of the field experiment. The execution methodology is based on my work from Paper ζ and I extended Cloud WorkBench [a] to enable large-scale experimentation across many configurations. Hence, my main writing contributions are primarily related to the approach and execution methodology, in addition to *Writing – Review & Editing* for the entire publication.

[3]`https://casrai.org/credit/`
[4]`https://research.spec.org/home.html`

Table 1: Summary of personal contributions per paper according to the Contributor Roles Taxonomy (CRediT)[3]

	Conceptualization	Data Curation	Formal Analysis	Funding Acquisition	Investigation	Methodology	Project Administration	Resources	Software	Supervision	Validation	Visualization	Writing – Original Draft	Writing – Review & Editing
Paper α	✓	✓	✓		✓	✓		✓	✓		✓	✓	✓	✓
Paper β	✓	✓			✓	✓		✓		✓	✓		✓	✓
Paper γ	✓	✓	✓		✓	✓		✓	✓		✓	✓	✓	✓
Paper δ	✓	✓	✓			✓		✓	✓	✓	✓	✓	✓	✓
Paper ε	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓	✓	✓
Paper ζ	✓	✓	✓		✓	✓		✓	✓		✓	✓	✓	✓
Paper η	✓	✓	✓		✓	✓		✓	✓		✓	✓	✓	✓
Paper θ	✓				✓	✓			✓		✓			✓

Contents

Chapter 1

Synopsis

Cloud computing [1, 2] has transformed the delivery of modern software systems. The established cloud computing paradigm Infrastructure-as-a-Service (IaaS) grows unabatedly [3–5] and the emerging paradigm Serverless computing experiences rapid adoption [6–10]. IaaS can be seen as the core of cloud environments offering low-level computing resources (e.g., CPU processing time or disk space) as self-service, prevalently in the form of virtual machines (VMs). As cloud computing evolves towards higher-level abstractions such as the serverless paradigm, it aims to liberate users entirely from operational concerns, such as managing or scaling server infrastructure. Function-as-a-Service (FaaS) is one embodiment of serverless and offers a high-level fully-managed service with fine-grained billing to execute event-triggered code snippets (i.e., functions).

The continuing growth of the cloud computing market has led to an unprecedented diversity of cloud services offered in many different configurations with varying performance characteristics. Hence, selecting an appropriate cloud service with an optimal configuration for application performance and cost-efficiency is a non-trivial challenge.

Performance evaluation is a field of research that systematically measures characteristics such as latency or throughput to build an understanding of performance in a given environment. Serverless performance evaluation is a young but very active area of research that lacks a consolidated understanding and application-level performance insights. In contrast, performance evaluation in IaaS clouds is an established area of research but requires new methods for reproducible experimentation and for understanding the relationship between different types of performance benchmarks (i.e., performance tests). Therefore, this thesis formulates the following goal:

Goal

> My PhD thesis aims to enable reproducible performance evaluation of serverless applications and their underlying cloud infrastructure.

To achieve this goal, this thesis performs empirical research on serverless applications and performance, contributes novel approaches and benchmarks for serverless and their underlying cloud infrastructure, and conducts field experiments in real cloud environments.

The remainder of this chapter is organized as follows. Section 1.1 introduces relevant background on cloud computing and the foundations of performance evaluation. Section 1.2 summarizes related work in the fields of FaaS and IaaS performance evaluation. Section 1.3 describes challenges that motivate the high-level research questions in Section 1.4. Section 1.5 summarizes the research methodology used to address the research questions. The contributions of the individual papers are summarized and linked to the research questions in Section 1.6. The research questions are then answered in Section 1.7 and discussed in a larger context in Section 1.8. Section 1.9 outlines future research directions and Section 1.10 concludes this thesis.

1.1 Background

This section defines cloud computing, serverless computing, and Function-as-a-Service (FaaS). It further introduces the foundations of performance evaluation, distinguishes between micro- and application-level benchmarks, describes distributed tracing, and discusses reproducibility in science.

1.1.1 Cloud Computing

Cloud computing [2, 11–14] is most commonly defined as:

> a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.
>
> —The NIST Definition [1]

Cloud computing continues to evolve, moving from low-level generalist services towards more specialized high-level services. Early *Infrastructure-as-a-Service (IaaS)* clouds offer a low-level abstraction of computing resources. These resources are most commonly provided in the form of self-administered *virtual machines (VMs)* where users have near full control of the software stack [15]. Cloud VMs are offered in many different sizes (also called instance types) with different performance and cost characteristics. A prominent example of an IaaS compute service is the Amazon Elastic Compute Cloud (EC2), which was initially introduced by the cloud provider Amazon Web Services (AWS) in 2006 [16].

As cloud computing matures, new services push towards more fine-grained deployment units of increasingly specialized services as depicted in Figure 1.1. VMs virtualized the hardware of bare metal machines, containers provide virtualization on top of a shared operating system, and Function-as-a-Service (FaaS) offers prepackaged runtimes for high-level application development. FaaS deployment units are small code functions written in high-level programming languages such as JavaScript or Python. Hence, FaaS allows developers to focus on business logic while abstracting away operational concerns, such as autoscaling VMs.

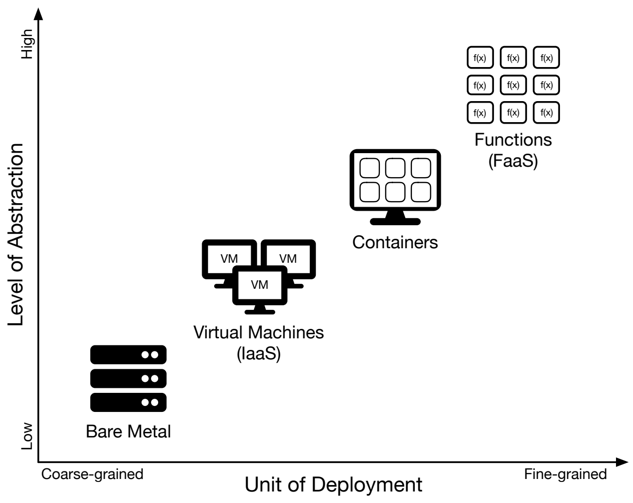

Figure 1.1: Progression of deployment options (adapted from [17–20]).

1.1.2 Serverless Computing and Function-as-a-Service

Despite several popular definitions for serverless computing and Function-as-a-Service (FaaS) [19, 21–25], both terms are often used inconsistently and sometimes even with contradicting interpretations [19]. The term *serverless* (i.e., without *managing* servers) can be considered confusing because serverless platforms are technically built on servers but they are managed by a cloud provider rather than a cloud user. Nevertheless, the term serverless is widely adopted by academics and practitioners [22, 23]. This thesis adopts an interpretation based on an accessible introduction to serverless computing [21], the vision on FaaS and serverless architectures from the SPEC Cloud research group [24], and a definition based on broad discussions in a Dagstuhl seminar on serverless computing [25].

> *Serverless computing* is a cloud computing paradigm that aims to liberate users entirely from operational concerns, such as managing or scaling server infrastructure, by offering a fully-managed high-level service with fine-grained billing.

> *Function-as-a-Service (FaaS)* is one embodiment of serverless computing and is defined through FaaS platforms (e.g., AWS Lambda) executing event-triggered code snippets (i.e., functions).

Figure 1.2 visualizes the relationship between serverless and FaaS and lists example FaaS platforms[1]. This thesis focuses on serverless applications using FaaS and does not explicitly cover serverless or event-driven computing without FaaS. For example, the performance of serverless storage (e.g., Amazon S3) can be relevant as part of serverless applications using FaaS and external services [26] but is not considered in isolation [27]. Paper α is framed as *FaaS* from that perspective, while the subsequent Papers β to ε are framed

[1]`https://landscape.cncf.io/format=serverless`

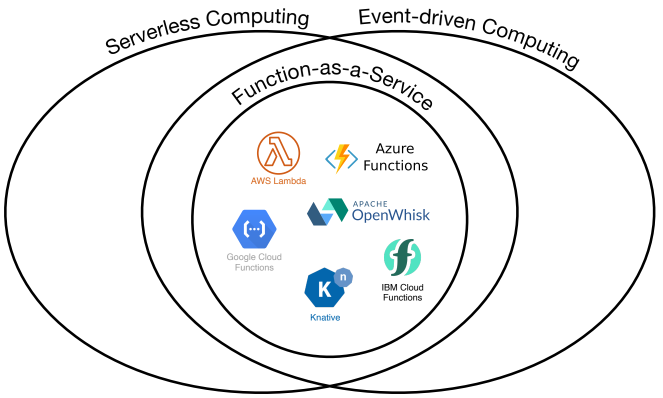

Figure 1.2: Relationship between serverless and FaaS (adapted from [19]).

as *serverless* to emphasize the tight integration with external services of FaaS. Practitioners [22, 23] define *serverless* as a combination of FaaS and Backend-as-a-Service (BaaS). BaaS refers to managed services such as Amazon S3 and is also dubbed *external services* from a FaaS perspective. In practice, the terms *FaaS* and *serverless* are often used interchangeably and therefore this thesis uses *serverless functions* to distinguish FaaS (e.g., AWS Lambda) from BaaS (e.g., Amazon S3).

1.1.3 Performance Evaluation

Performance evaluation, also known as performance benchmarking or performance testing, is the process of systematically evaluating performance features (e.g., latency or throughput [28]) of computing resources (e.g., CPU, memory) and applications (e.g., Web serving, scientific computing).

The fundamental performance testing terminology includes the following components: system under test, workload, benchmark, and benchmark suite. A *system under test (SUT)* refers to environments or components that are evaluated according to clearly defined metrics, such as response time. In the context of this thesis, the SUT is typically either a cloud environment (i.e., IaaS or FaaS) or an application within a cloud environment. A *workload* refers to the stimulation that is applied to a SUT to observe a certain effect (e.g., change in performance). This thesis distinguishes between synthetic workloads for micro-benchmarks and realistic workloads for application-benchmarks, which intend to imitate real-world scenarios. A *benchmark* tests performance in a controlled setup by applying a workload to a SUT. A *benchmark suite* groups a set of related benchmarks and defines an execution methodology for combined execution.

Concrete performance features [28], metrics [29], and evaluation methods [30, 31] are cataloged in related work and described within the thesis where relevant.

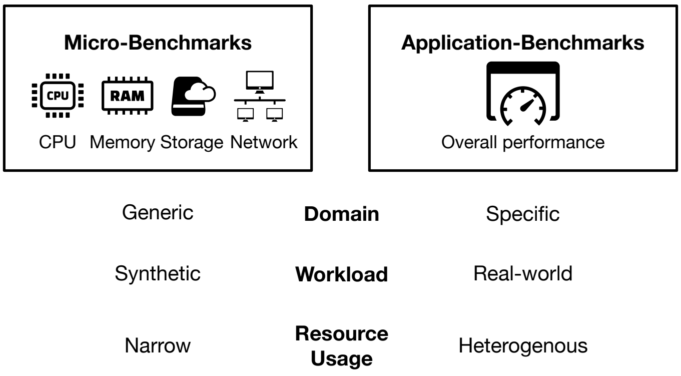

Figure 1.3: Micro- vs. application-benchmarks.

1.1.4 Micro- and Application-Benchmarks

Figure 1.3 compares two common types of benchmarks [32–34], namely micro- and application-benchmarks. *Micro-benchmarks* target a narrow performance aspect (e.g., floating-point CPU performance) with synthetic workloads. These generic benchmarks are not bound to a certain domain (e.g., Web serving) but can provide performance insights that are potentially transferable within certain execution environments (e.g., VM instance type). *Application-benchmarks*, also known as macro-benchmarks, aim to cover the overall performance of real-world application scenarios. Typical metrics are end-to-end response time or throughput. Their results are either specific to a certain application under a given workload or a domain of related applications (e.g., Web serving or scientific computing). Their resource usage profile might be complex and dynamic as they are designed to solve a real-world task rather than testing a specific resource in isolation. Application-benchmarks tend to be long-running, complex to configure, and hard to explain due to their large scope [34]. Examples of both benchmark types are described in Section ζ.3.3 for IaaS and in Section α.2 for FaaS.

For synchronously invoked applications, the overall performance can be measured as client-side response time. However, the end-to-end latency for serverless applications is hard to measure due to asynchronous call boundaries across external services. Therefore, distributed tracing is required for full observability and will be discussed in the next section.

1.1.5 Distributed Tracing

Distributed tracing [35–38] aims to achieve end-to-end observability of a request across distributed components. In 1994, Schwarz and Mattern [39] formally introduced detection models for causual relationships in distributed systems and tracing solutions started to emerge in the 2000s, for example Magpie [40] or X-Trace [36]. Google popularized distributed tracing [41] with Dapper [37] and many other companies adopted the practice as shown in an industry adoption report [42].

Figure 1.4 visualizes an end-to-end backend trace for a synchronous applica-

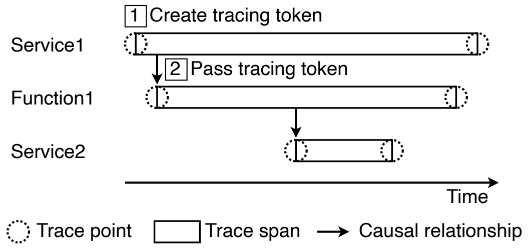

Figure 1.4: Simplified causal-time trace diagram of a synchronous invocation.

tion with a causal invocation chain starting from *Service1* over *Function1* into *Service2*. The service receiving an incoming request (e.g., *Service1*) generates a unique tracing token $\boxed{1}$ for each request. This tracing token is then used to label each timestamp captured at trace points of interest and needs to be passed $\boxed{2}$ into every downstream service along the invocation chain. Two consecutive trace points are grouped into a trace *span* if they encompass a specific operation (e.g., computation) from the same component (e.g., *Service2*). A centralized tracing service correlates spans of the same request from all components using the tracing token to build a trace graph with causal relationships.

1.1.6 Reproducibility

"Repeatability and reproducibility are cornerstones of the scientific process" [43] but often neglected in natural [44] and computer [43] science research. Repeatability refers to the extent successive measurements with the same method under *the same conditions* yield the same results [45]. A repeatability study [43] found that more than half of 601 papers from top-rated ACM systems conferences around 2012 lack functional code. Even after spending ample efforts to fix build failures, repeatability was impossible for the results of at least half of these papers. However, reproducibility could still be achieved as it refers to the extent the same results can be achieved with the same method under *changed conditions* of measurements [45].

Following these definitions, repeatability is practically impossible in public cloud environments due to the lack of control over a multi-tenant environment (i.e., shared among many users) offered by a third-party cloud provider. Therefore, this thesis focuses on *technical reproducibility* of cloud experimentation, which requires several aspects to ensure an experiment can be repeated with the *same* methodology. A complete description is often unrealistic for space-constrained research papers [46] or for blogposts that aim for a short attention span. Therefore, technical artifacts should be published as an online appendix in a usable form [47]. This might include source code, input data (e.g., workloads), and technical descriptions. Access to the same infrastructure is fundamentally given for public clouds but hampered due to their continuous evolution and potentially high costs. Due to the fast evolution of modern computational environments, Lin and Zhang [48] advocate for an understanding of reproducibility as a *process* rather than as an *achievement*.

1.2 Related Work

This section discusses related work on (i) serverless performance evaluation and application characteristics, (ii) serverless application benchmarking and distributed trace analysis, and (iii) IaaS performance evaluation and reliable performance testing in cloud environments.

1.2.1 Serverless Performance Evaluation

Performance evaluation in serverless and FaaS has a 6-year history with first studies [49, 50] appearing around 2016. The first reports followed the public release of AWS Lambda in 2015[2], which is considered the first FaaS offering by a large public cloud provider. Systematic mapping studies [51, 52] indicate that performance is the most popular research direction in the field of serverless computing. However, current reports on FaaS performance are disparate originating from different studies executed with different setups and different experimental assumptions. The serverless community is lacking a consolidated view on the state of research on FaaS performance. To the best of my knowledge, there exists no unified view on FaaS performance apart from a literature review reporting on preliminary results [53]. Kuhlenkamp and Werner [53] proposed a methodology for a collaborative literature review on FaaS performance evaluation and reported preliminary results from 10 academic studies. Otherwise, the FaaS performance evaluation landscape has only been discussed as part of limited related work sections in primary studies, most thoroughly by Somu et al. [54] across 7 studies.

1.2.2 Serverless Application Characteristics

The most extensive curated collection of real-world serverless applications lists 15 applications [55] and another collection of 13 applications summarizes how serverless is used for four common use cases [21]. Cloud providers (e.g., AWS Serverless Application Repository[3]) and FaaS frameworks (e.g., Serverless Framework[4]) publish their collections of serverless applications but these examples typically serve rather as developer documentation than real-world applications. Other studies addressed developer experience [56] and patterns for serverless functions [57]. However, the characteristics of individual serverless applications have not been systematically analyzed by prior work.

1.2.3 Serverless Application Benchmarks

Existing application-level benchmarks and empirical performance evaluations focus on the overall response times of single-function applications. ServerlessBench [58] presents a diverse application benchmark with four multi-function applications but is limited to synchronous invocations and therefore doesn't cover typical serverless applications coordinated by asynchronous function triggers. From a cloud providers perspective, vHive [59] and faas-profiler [60] evaluate server-level overheads caused by CPU branch mispredictions or hypervisor load

[2]https://aws.amazon.com/blogs/compute/aws-lambda-is-generally-available/
[3]https://aws.amazon.com/serverless/serverlessrepo/
[4]https://github.com/serverless/examples

times for coldstarts. From a developer's perspective, FunctionBench [61] and SeBS [62] offer diverse single-function applications and BeFaaS [63] presents a single multi-function application. However, none of the existing application performance studies supports diverse external services and asynchronous function coordination, which are both core premises of event-based serverless architectures.

1.2.4 Distributed Trace Analysis

Distributed tracing is common in microservice architectures but its practice and analysis are big challenges across all software engineering [64–66]. A survey among 106 practitioners working with microservices showed that distributed tracing is among the top observability challenges mentioned by 45 % of the respondents [64]. A related interview study across ten companies identified many challenges and raised intelligent trace analysis techniques as a new big data problem for software engineering [65]. Bento et al. [66] outline challenges and research directions for automated analysis of distributed traces. Current production systems such as Canopy [67] from Facebook are primarily used for ad hoc manual analysis [66, 67] but research proposed several techniques for automated trace analysis. Schwarz and Mattern [39] introduce a formal notation for causality and time and survey approaches for detecting causal relationships in distributed systems. Pivot tracing [68] introduces an efficient *happened-before join* operator to facilitate cross-component event correlation. Hendriks et al. [69] present algorithms for critical path analysis and trace graph comparison based on their generalized graph representation of execution traces [70]. FIRM [71] combines critical path and critical component analysis with machine learning models to identify and mitigate service level objective (SLO) violations. Luo et al. [72] use graph clustering to characterize the call graph dependency structure and performance of production microservice at Alibaba.

Although *tracing for serverless computing* raises several new challenges, it has received little attention. GammaRay [73] augments AWS X-Ray to track casual ordering and Lowgo [74] proposes a tracing tool for multi-cloud serverless applications. A comparison study of different serverless tracing tools investigates how well they detect different types of faults [75] and Costradamus [76] uses distributed tracing to estimate per-request costs. However, serverless tracing is still emerging and trace analysis remains a largely manual process [77]. Provider-managed infrastructure limits access to fine-grained instrumentation and developers need to rely on distributed tracing services offered by cloud providers. This leads to observability gaps and typically requires implicit tracing of downstream services due to missing tracing support. Further, the event-based nature of serverless requires adaptations to traditional critical path analysis for synchronous invocation patterns as performed in FIRM [71].

1.2.5 Infrastructure-as-a-Service Performance Evaluation

Performance evaluation in IaaS cloud environments has a 15-year history with the first reports [78–80] appearing around 2007. The first reports followed the beta release of Amazon EC2 in 2006 [16], which is considered to be the first

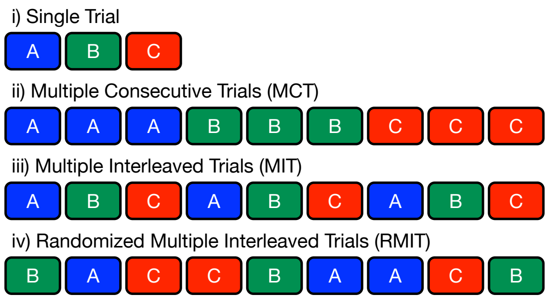
i) Single Trial
A B C
ii) Multiple Consecutive Trials (MCT)
A A A B B B C C C
iii) Multiple Interleaved Trials (MIT)
A B C A B C A B C
iv) Randomized Multiple Interleaved Trials (RMIT)
B A C C B A A C B

Figure 1.5: Different execution methodologies for three alternatives `A`, `B`, `C` (reproduced from [93]).

commercially available IaaS cloud provider. Since then, cloud performance evaluation has become a popular research area with hundreds of papers published on topics such as benchmarking expectations [81, 82], performance metrics [28, 29], benchmarking approaches [30, 31], performance benchmarks [83], performance experiments [84–87], or hardware heterogeneity [88, 89]. Secondary studies classified existing research [90] and experimentally validated hypotheses derived by codifying primary studies [91]. Unfortunately, the rapid evolution of cloud systems requires continuous re-evaluation [91] and new methods towards reproducible experimentation in inherently unstable cloud environments [91, 92].

1.2.6 Cloud Benchmarking Execution Methodology

Existing measurement methodology often makes incorrect assumptions about the underlying system under test when combining multiple performance benchmarks. Abedi and Brecht [93] proposed a new execution methodology called Randomized Multiple Interleaved Trials (RMIT). Figure 1.5 visualizes RMIT with three alternatives, which could represent different benchmarks. Single-trial and multiple consecutive trials (MCT) are currently the most common methodologies in practice but could lead to erroneous conclusions. Therefore, RMIT should be used to account for potential periodic effects in cloud environments beyond the control of experimenters. RMIT was evaluated through simulation based on measurements of micro-benchmarks collected by other researchers [87].

Several IaaS cloud experiment automation frameworks have been proposed [94–97] but only IBM's Cloud Rapid Experimentation and Analysis Toolkit (CBTOOL)[5] described by Silva et al. [94] and Google's PerfKitBenchmarker[6] are still actively maintained. None of the existing frameworks provide execution methodologies beyond serial trials. Hence, I am not aware of any IaaS benchmark suite that systematically combines multiple benchmarks using a state-of-the-art execution methodology.

[5]`https://github.com/ibmcb/cbtool`
[6]`https://github.com/GoogleCloudPlatform/PerfKitBenchmarker`

1.2.7 Cloud Application Performance Prediction

Application performance prediction for optimizing cloud service selection is a common area of research, especially in the context of cloud migration. Initial prediction methods, such as CloudProphet [98], primarily focused on predicting application performance in cloud environments when migrating an application from an on-premise application, for example through trace-and-replay. As cloud offerings started to become more diverse, holistic methods and tools for cloud rightsizing [99, 100] have been proposed to support cloud migration and optimal service selection. Optimization methods based on micro-benchmarking were proposed and validated for scientific applications [101]. So far, these methods are typically limited to few service types and applications from a single domain. Further, training and validation of existing studies might be negatively impacted by the lack of a state-of-the-art execution methodology.

Three of the most related studies were published shortly before and after Paper ζ. Yadwadkar et al. [102] predict the performance of video-encoding and Web serving applications with diverse resource profiles for 11 to 15 VM instance types in two cloud providers using hybrid online and offline data collection and modeling. Their profiling benchmarks are limited to cherry-picked workload requirements, described in insufficient detail, and unavailable, neither as code nor dataset. Wang et al. [103] use two micro-benchmarks to predict the performance of seven programs from a CPU-intensive benchmark suite for three different VM instance types across two cloud providers. Baughman et al. [104] predict the performance of bioinformatic workflows for 14 VM instance types by combining historical resource traces with online profiling. None of the three studies use interleaved or randomized trials.

1.2.8 Performance Testing in Cloud Environments

Traditional performance testing is conducted as a laboratory experiment in a contrived setting using self-managed bare metal hardware for maximum precision of the measurements. Cloud environments have become attractive testbeds for long-running test performances test suites due to their rapid availability of seemingly unlimited computational resources. Further, with cloud environments becoming the deployment target of many applications, cloud-specific performance characteristics might only be observable in real cloud environments. However, performance fluctuations (i.e., unstable or variable performance) are common in cloud environments [87, 91, 92, 105–107] due to virtualization [108], noisy neighbors [109], or hardware heterogeneity [88, 89].

Software microbenchmarks are a type of performance test where source code at method-level is used as a workload and repeatedly executed to obtain a performance distribution. A benchmarking harness such as JMH for Java orchestrates the testing process and reports summary statistics such as average execution time, throughput, or resource utilization. They are sometimes referred to as *unit tests* for performance [110, 111] but are seldomly used in open source projects according to Github mining studies [111, 112] due to challenges related to automation [111] and implementation [112]. An empirical study of 123 open source Java microbenchmarks has shown that bad practices can severely impact the outcome of these tests [113]. Chen and Shang [114] execute software microbenchmarks in a cloud environment and found that most code changes lead to both performance improvements and performance regressions at the same time. Hence, the unstable nature of cloud-based execution environments for software microbenchmarks might affect their reliability.

1.3 Challenges

This section describes six challenges that motivate my research based on gaps outlined in related work.

Challenge 1 (C1): No consolidated view on serverless performance evaluation Previous research has indicated performance-related challenges common to many FaaS platforms such as slow coldstarts, unpredictable performance, or substantial platform overheads. So far, reports about performance-related challenges in FaaS are disparate and originate from different studies (see Section 1.2.1), executed with different setups and different experimental assumptions. The serverless community is lacking a consolidated view on the state of research on FaaS performance.

Challenge 2 (C2): No consolidated view on serverless application characteristics Current reports about serverless applications regarding their motivation, context, and implementation are scattered and sometimes conflicting. Cloud developers seek guidance on questions such as why developers build serverless applications, when are they well-suited, or how are they implemented. However, there are currently no systematic studies about serverless applications and their common characteristics (see Section 1.2.2).

Challenge 3 (C3): Insufficient application benchmarks Existing application benchmarks are typically limited to single-function applications and integrated with at most a single type of external service. Most importantly, no prior work covers asynchronous applications although serverless architectures are inherently event-driven, and most event-based function triggers behave asynchronously. Hence, the serverless community lacks a realistic application benchmark designed based on real-world application characteristics (see Section 1.2.3).

Challenge 4 (C4): No fine-grained performance characterization of common serverless applications Existing serverless performance studies typically report the overall response time and derive insights through extensive experimentation and sensitivity analysis of several factors. Such coarse-grained results are hardly actionable and current approaches for distributed trace analysis are primarily manual (see Section 1.2.4). Further, solely focusing on synchronous response times ignores an important class of applications given the asynchronous nature of many serverless applications. Therefore, serverless studies should provide fine-grained insights into asynchronously coordinated applications.

Challenge 5 (C5): Unclear relationship between micro- and application-level benchmarks Given the strong focus on micro-benchmarks in both FaaS and IaaS, it remains unclear how relevant these artificial benchmarks are to gaining insights into the performance of real-world applications. Despite extensive research of IaaS cloud infrastructures (see Section 1.2.5), existing work does not systematically combine different types of benchmarks using state-of-the-art execution methodology (see Section 1.2.6) and approaches for application performance prediction are limited in scope (see Section 1.2.7). Therefore, a systematic study of different benchmark types is needed to evaluate the usefulness of micro-benchmarks for application performance prediction.

Challenge 6 (C6): Unclear reliability of performance evaluation in the cloud Multi-tenant cloud infrastructures are known to cause unstable performance (see Section 1.2.8) and flawed measurement methodologies in cloud environments could lead to erroneous conclusions (see Section 1.2.5). However, it remains unclear to what extent different benchmarks are affected by performance variability, and how reliable software performance tests can be in unstable cloud environments.

1.4 Research Questions

To address the goal of this PhD thesis, I formulate the following high-level research questions (RQs) motivated by the six challenges raised in the previous section:

RQ1 *What is the current state of serverless applications and their performance?*

Serverless computing is a very active field of research but lacks a consolidated view on performance evaluation (C1) and application characteristics (C2). To address this gap, RQ1 aims to systematically map the landscape of existing work on serverless performance evaluation and identify common characteristics of serverless applications from diverse sources.

RQ2 *What are the performance challenges of serverless applications?*

Studies on serverless performance evaluation focus on artificial micro-benchmarks and realistic applications remain insufficiently studied (C3). To address this gap, RQ2 aims to propose a novel application benchmark constructed based on insights from RQ1 and subsequently conduct benchmarking experiments to identify performance challenges in realistic serverless applications through fine-grained trace analysis (C4).

RQ3 *How can limitations of benchmarking cloud infrastructure be addressed?*

The underlying cloud infrastructure of serverless platforms can affect the validity of performance measurements. Such limitations of cloud benchmarking can hamper the usefulness of benchmarks in predicting application performance and compromise the reliability in detecting performance regressions. Therefore, RQ3 targets IaaS clouds to clarify

Table 1.1: Mapping of research methodologies to research questions.

Research Methodology	Section	RQ
Literature Review	1.5.1	RQ1
Sample Study	1.5.2	RQ1
Engineering Research	1.5.3	RQ2+RQ3
Field Experiment	1.5.4	RQ2+RQ3

the relationship between micro- and application-benchmarks (C5) and quantify the performance variability and reliability in cloud benchmarking (C6).

1.5 Research Methodology

This section summarizes the research methodology used to answer the research questions of this thesis. The terminology is based on the framework "ABC of Software Engineering Research" [115] for knowledge-seeking primary studies and complemented with the "ACM SIGSOFT Empirical Standards" [116] and established research guidelines for solution-seeking [117] and secondary research studies [118, 119].

Table 1.1 summarizes the mapping of research methodologies (this section) to the research questions of this thesis (Section 1.4). For RQ1, a literature review and sample study were selected to address the lack of a consolidated view on FaaS performance evaluation and serverless application characteristics. The literature review on FaaS performance evaluation was suitable because many individual studies existed but a systematic topic mapping and synthesis of evidence were missing. The sample study on serverless application characteristics was suitable because no systematic collection of applications was available and the goal was to study the serverless applications (i.e., primary research) and not the contributions of existing studies (i.e., secondary research). The results of these knowledge-seeking research methodologies identify relevant gaps in the literature and practical problems to be addressed in subsequent solution-seeking research. Therefore, RQ2 and RQ3 adopt engineering research to propose novel approaches, tools, and algorithms (i.e., solution-seeking research) and use field experimentation to evaluate the proposed solutions.

1.5.1 Literature Review

A systematic literature review is a type of secondary research study that maps topics and synthesis evidence from original primary studies in a defined field of research. Figure 1.6 summarizes the taxonomy of systematic secondary studies by clarifying the types of analyses and sources under study. A multivocal literature review (MLR) [119] combines topic mapping and synthesis of evidence (i.e., aggregation of insights) for academic and grey literature. Non-peer-reviewed grey literature includes sources such as white papers, presentations, or blog posts. Including grey literature about FaaS performance was relevant given the strong industrial interest and the goal to identify potential mismatches

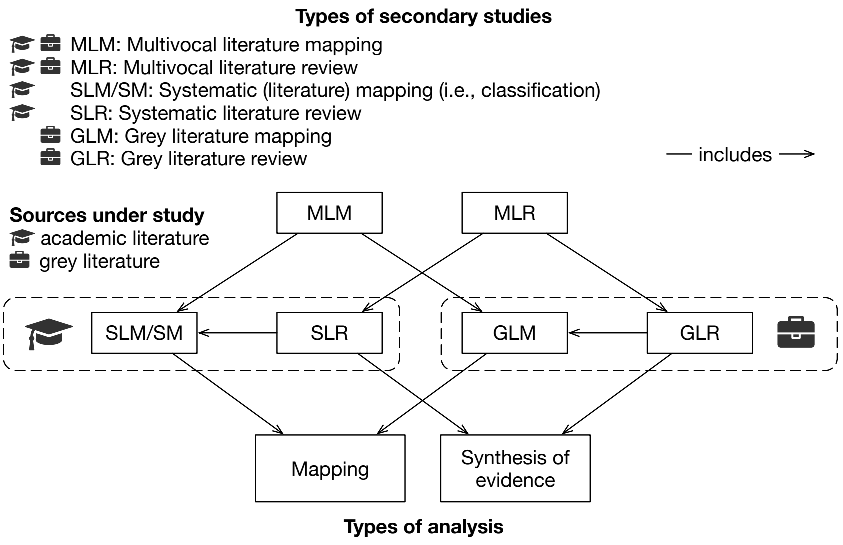
Types of secondary studies
MLM: Multivocal literature mapping
MLR: Multivocal literature review
SLM/SM: Systematic (literature) mapping (i.e., classification)
SLR: Systematic literature review
GLM: Grey literature mapping
GLR: Grey literature review
— includes ⟶
Sources under study
academic literature
grey literature
MLM
MLR
SLM/SM
SLR
GLM
GLR
Mapping
Synthesis of evidence
Types of analysis

Figure 1.6: Taxonomy of systematic secondary studies (adapted from [119]).

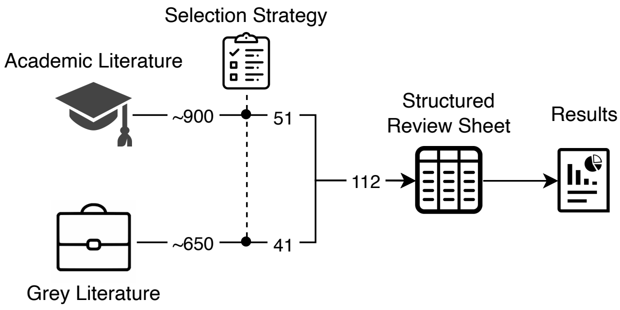

Figure 1.7: Multivocal literature review process summary.

between the academic and industrial perspectives. The mapping of FaaS experiment designs helps to identify research gaps and aggregated insights on reproducibility challenges can guide future studies.

Figure 1.7 summarizes the MLR process of Paper α, which identified 112 relevant primary studies from academic (51) and grey (41) literature. Peer-reviewed papers (e.g., papers published in journals, conferences, and workshops) are classified as academic literature (i.e., white literature) and other studies (e.g., preprints of unpublished papers, student theses, blog posts) as grey literature. The search process and source selection for academic literature follow a conventional systematic literature review (SLR) process [118]. It was guided through an initial seed of studies [120] discovered through manual search [121] and refined through complementary search strategies, such as alert-based search. The search and selection process for grey literature is based on guidelines for including grey literature [119].

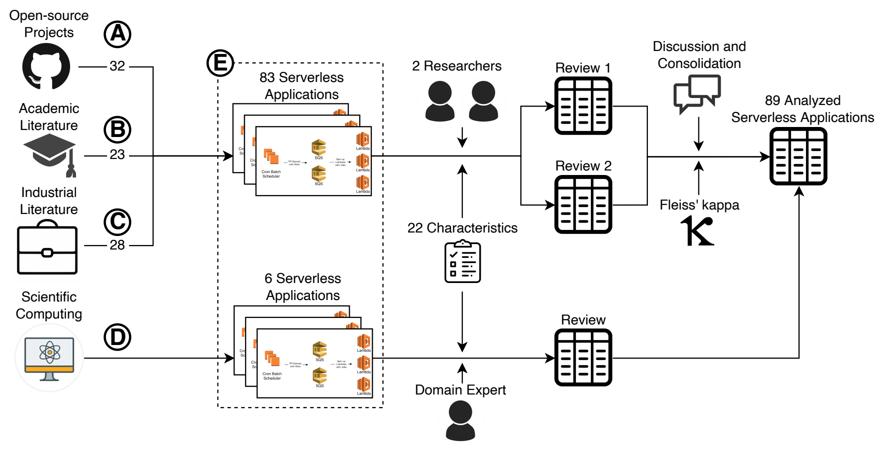

Figure 1.8: Sample study process.

1.5.2 Sample Study

A sample study is conducted in a neutral setting (i.e., desk research) and involves a purely observational analysis of artifacts such as documentation or source code [115]. This research strategy was suitable for characterizing serverless applications in Paper *β* to achieve high generalizability of findings by including applications from a broad range of different sources. A follow-up meta-analysis across related primary studies improves the generalizability of the findings even further. The direct analysis of documentation and source code related to subject applications qualifies as primary research. The inclusion of academic literature in the broad data collection might initially hint towards secondary research but the goal was to study serverless applications and not the contributions of primary studies. A sample study is inherently limited to the data available because data collection is not interactive (i.e., "data comes as is" [115]). Therefore, 6 characteristics were excluded due to insufficient information available.

Figure 1.8 summarizes the process of analyzing 89 serverless applications from four different sources. First, descriptions of 89 serverless applications Ⓔ were collected from open-source projects Ⓐ, academic literature Ⓑ, industrial literature Ⓒ, and a scientific computing organization Ⓓ. Second, two randomly assigned reviewers out of seven available reviewers characterized each application along 22 characteristics in a structured collaborative review sheet. The characteristics and potential values were defined a priori by the authors and iteratively refined, extended, and generalized during the review process. After an initial moderate inter-rater agreement [122], a discussion and consolidation phase resolved all differences between the two reviewers with consultation among all authors if necessary. The six scientific applications were not publicly available and therefore characterized by a single domain expert, who is either involved in the development of the applications or in direct contact with the development team.

The sampling strategy of serverless applications is important to achieve a varied sample from different sources, although the characterization is the

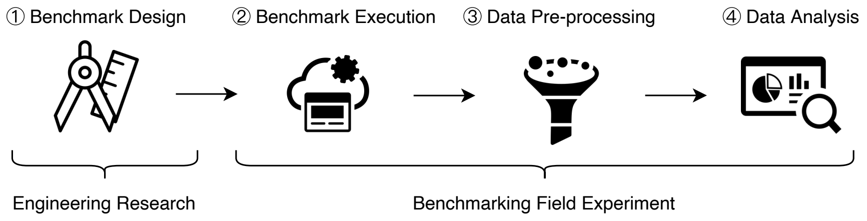

Figure 1.9: Research process of engineering research and field experiment.

primary goal of this study (i.e., following a positivist and reductionist philosophical stance [123]). Following the terminology and guidelines by Baltes and Ralph [124], this study applied different kinds of purposive sampling for the 83 publicly available serverless applications and convenience sampling for the six internal scientific computing applications analyzed by an author employed at the German Aerospace Center. Heterogeneity sampling motivated a roughly balanced selection of open source projects, academic literature (including scientific computing), and industrial literature. Search-based sampling was applied for open source projects through an initial keyword search of the offline GitHub mirror GHTorrent [125] and refined through filtering based on date range, repository activity, repository popularity, and manual selection following inclusion and exclusion criteria. Search-based sampling was applied for academic literature mainly based upon manual selection from the "Serverless Literature Dataset" [126]. Collaborative referral-chain sampling was the main source for grey literature seeded by case studies reported by cloud providers, an existing survey article [21], blog posts, discussion forums, and podcasts known to the authors.

1.5.3 Engineering Research

Engineering research is a type of solution-seeking research that invents and evaluates technical artifacts [115, 116] to solve a practical problem previously identified through knowledge-seeking research [117]. This thesis uses benchmarking studies (i.e., a specific type of field experiment) to evaluate the proposed solutions, which is a common combination according to the "ACM SIGSOFT Empirical Standards" [116, 127]. Figure 1.9 visualizes this common research process used in Papers γ to η. First ①, *benchmark design* involves developing workloads and measurement tools, typically in the form of a *kit-based* benchmark suite [128]. This process is guided by literature on benchmark construction [33, 127–129], cloud benchmarking [31, 81, 130], and existing benchmarks (see literature review in Paper α). It often includes a combination of configuring, porting, and implementing several performance benchmarks into a new benchmark suite. The artifact descriptions in Papers γ to η cover relevant aspects such design principles, architecture overview, measurement methodology, algorithms, applications, fairness design, and configurability. Implementations of key aspects are covered by unit and integration test suites. All artifacts are available as open source software and accompanied by extensive documentation.

A benchmarking field experiment uses the proposed solutions through engineering research (i.e., benchmark suite). Second ②, *benchmark execution* involves defining experiment plans, scheduling executions, and monitoring potentially multi-week experiments in public cloud environments. Benchmark execution generates large amounts of raw performance data (e.g., >70 GB in Paper γ). Third ③, *data pre-processing* prepares the raw data for analysis through filtering (e.g., skip irrelevant executions), validation (e.g., skip erroneous executions), re-shaping (e.g., transpose or rename), and refinement (e.g., convert units). Fourth ④, *data analysis* calculates summary statistics, applies statistical models, and visualizes performance distributions to answer specific research questions.

1.5.4 Field Experiment

According to the *ABC of Software Engineering* by Stol and Fitzgerald [115], a field experiment is a research strategy conducted in a natural setting (e.g., in a real public cloud environment) where the researcher manipulates some variables (e.g., instance type, benchmark configurations) to observe some effect (e.g., performance metrics). Public cloud environments are massive-scale multi-tenant systems and their emergent performance properties cannot be replicated in a fully contrived setting as a laboratory experiment. Therefore, only a natural setting can provide maximum realism for cloud benchmarking. Unfortunately, the limited control within real cloud environments impedes reproducible performance evaluation [46], which is an ongoing challenge in both IaaS [46] and in FaaS clouds (Paper α). Mitigating these reproducibility challenges is a cross-cutting concern throughout the field experiment studies in Papers γ to θ. In particular, these studies strive for full automation by leveraging infrastructure as code, configuration management, container technology, and programmable experiments. Each experiment provides a documented replication package including dataset, analysis scripts, and executable experiment plans. Finally, limited generalizability is an inherent limitation of this type of more intrusive research compared to less intrusive research (e.g., sample study described in Section 1.5.2).

The field experiments in this thesis are conducted as benchmarking research [127] to evaluate the performance characteristics of software systems in cloud environments. Benchmarking research in this thesis builds upon methodological guidance from the ACM SIGSOFT Empirical Standards [127], benchmark construction [128], requirements of a good benchmark [129], and generic approaches for IaaS cloud benchmarking [30, 31]. Benchmarking field experiments are often combined with engineering research as demonstrated in Figure 1.9.

Figure 1.10 visualizes the high-level architecture of a benchmarking field experiment used in Papers γ to ε. First ①, an application is deployed into a cloud provider using an automated deployment script. The application is instrumented with detailed trace points and forwards trace spans to a provider-specific tracing service. Second ②, a workload profile is applied to invoke the application. Third ③, the benchmark orchestrator retrieves raw traces from the tracing service. For traditional benchmarking of synchronous cloud applications in Papers ζ and η, distributed tracing is not necessary and performance is

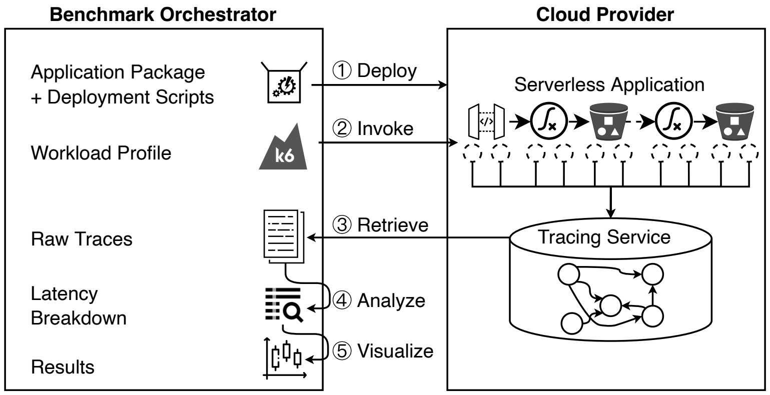

Figure 1.10: Field experiment process.

instead measured from the invoking client ②. For micro-benchmarks in Papers ζ to θ, no external invoker ② is needed because these benchmarks are directly invoked within a cloud VM deployed in a cloud provider. Fourth ④, raw traces need to be pre-processed (e.g., filtered, validated, refined) before they can be visualized ⑤ as results.

1.6 Contributions

This section summarizes the appended papers in Chapters α to θ, their main contributions this thesis is built on, and their relation to the challenges derived from the research questions of this thesis (see Section 1.4). The main results of this thesis are summarized in the next section (see Section 1.7).

Figure 1.11 visualizes the contributions of the appended papers in context of the research questions targeting serverless (RQ1 and RQ2) and cloud infrastructure (RQ3). To consolidate the current state in serverless computing (RQ1), Paper α contributes a literature review (Section 1.5.1) on FaaS performance evaluation and Paper β conducts a sample study (Section 1.5.2) on application characteristics. To address gaps in serverless application performance (RQ2), Paper γ proposes a realistic trace-based application benchmark, Paper δ discusses fair cross-provider application benchmarking, and Paper ε contributes a function trigger benchmark. To address the limitations of benchmarking cloud infrastructure, Paper ζ contributes an integrated benchmark suite, which is used in Paper η to estimate application performance through micro-benchmarks, and its execution methodology is applied in Paper θ to evaluate the reliability of software micro-benchmarking in cloud environments. For more details, Table 1.2 provides a per-paper summary including publication venue, main contribution, and a mapping to the challenges described in Section 1.3.

1.6.α Function-as-a-Service Performance Evaluation

Context Performance evaluation in FaaS environments (Section 1.2.1) is the most popular area of research in the field of FaaS computing [51] and previous research has indicated many performance-related challenges such as

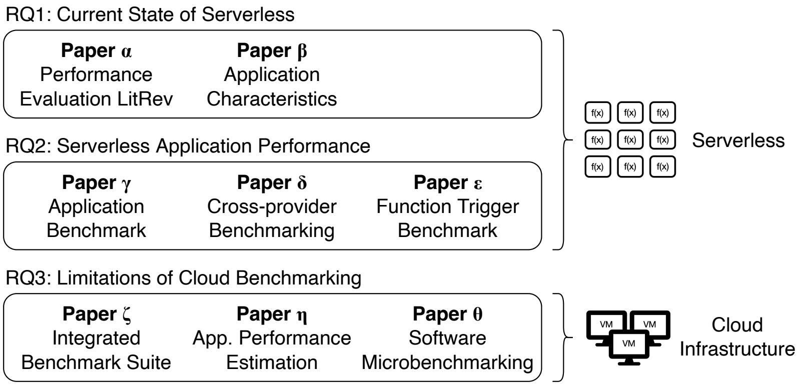
RQ1: Current State of Serverless
Paper α
Performance Evaluation LitRev
Paper β
Application Characteristics
RQ2: Serverless Application Performance
Paper γ
Application Benchmark
Paper δ
Cross-provider Benchmarking
Paper ε
Function Trigger Benchmark
RQ3: Limitations of Cloud Benchmarking
Paper ζ
Integrated Benchmark Suite
Paper η
App. Performance Estimation
Paper θ
Software Microbenchmarking
f(x)
Serverless
VM
Cloud Infrastructure

Figure 1.11: Overview of Contributions.

Table 1.2: Overview of papers with main contributions.

Paper	Venue	Main Contribution	Challenge
α	JSS'20	Review of 112 FaaS performance studies regarding performance characteristics, configurations, and reproducibility.	C1
β	TSE'21	Characterization of 89 serverless applications along 16 dimensions regarding motivation, context, and implementation. Meta-analysis across 10 studies.	C2
γ	Under submission	Realistic benchmark suite of 10 serverless applications. Large-scale performance experiment collecting over 7.5 million end-to-end traces. Novel approach for detailed latency breakdown analysis across asynchronous call boundaries.	C3 + C4
δ	Under submission	Tracing model for fair cross-provider benchmarking.	C4
ε	IC2E'22 (to appear)	Cross-provider benchmark for evaluating serverless function triggers.	C4
ζ	QUDOS'18	Automated benchmark suite that combines 23 micro- and 2 application-benchmarks.	C5
η	CLOUD'18	Cloud benchmarking methodology for application-benchmark estimation based on micro-benchmark profiling.	C5
θ	EMSE'19	Large-scale experiment collecting over 4.5 million software microbenchmark results.	C6

coldstarts [131], hardware heterogeneity [132], or function triggering delays [133]. So far, these reports are disparate and originate from different studies, executed with different setups, and different experimental assumptions. The FaaS communication is lacking a consolidated view on the state of research on FaaS performance.

Contribution Paper α fills this gap by conducting the first systematic and comprehensive literature review on FaaS performance evaluation studies from academic and grey literature. It maps the landscape of existing isolated FaaS performance studies, identifies gaps in current research, and systematically investigates their reproducibility based on principles for reproducible performance evaluation [46].

Method The literature review (Section 1.5.1) was designed based on guidelines for systematic literature reviews [118] and multivocal literature reviews [119]. A total of 112 studies were selected from academic (51) and grey (61) literature. The analysis visualizes, describes, and discusses results related to publication trends, benchmarked platforms, evaluated performance characteristics, used platform configurations, and reproducibility of experiments. The paper also highlights and discusses notable differences between academic and grey literature studies.

Relationship to Thesis The implications and gaps in the literature identified in this paper directly aim to guide future work on serverless performance evaluation. The lack of realistic application benchmarks motivated the empirical study in Paper β of real-world serverless applications. Based on these insights, Paper γ proposes a comprehensive application benchmark that addresses several gaps identified in this paper, including benchmark types, platform configurations, and reproducibility. Further research gaps in fair cross-provider comparison and function trigger types are targeted in Papers δ and ε.

1.6.β Serverless Application Characteristics

Context The emerging cloud computing paradigms Function-as-a-Service and serverless computing are increasingly adopted by the industry (shown by market analyses [134] and surveys [135]) and academics [26, 136–138]. Initial case studies from early adopters [139, 140] indicate significant cost reduction and time-to-market benefits for serverless applications compared to traditional cloud applications [141]. However, such existing reports are scattered and unstructured. The serverless community lacks an understanding of typical applications, which is crucial for designing relevant performance benchmarks.

Contribution Paper β characterizes 89 serverless applications along 16 dimensions regarding motivation, context, and implementation to answer questions such as: *Why do so many companies adopt serverless?*, *When are serverless applications well-suited?*, and *How are serverless applications currently implemented?* In addition to the 7 main findings of the sample study, a meta-analysis across 10 related studies identified 8 consensuses supported by evidence from multiple studies. This contribution extends our related IEEE Software article [l].

Method A sample study (Section 1.5.2) was used to collect and characterize existing serverless applications following a structured collaborative review process. Descriptions of serverless applications were collected from diverse sources including open source projects, academic literature, industrial literature, and a scientific computing organization. Each application was either reviewed by two researchers followed by a discussion and consolidation of all disagreements or by a single domain expert for the six scientific applications unavailable to the public. Finally, a meta-analysis compares the results of the sample study to 10 mostly industrial studies and datasets. This enables to validate results and identify points of agreement and disagreement towards building a community consensus.

Relationship to Thesis This paper shares several characteristics with the literature review on performance evaluation in Paper α, for example, cloud providers, programming languages/runtimes, external services, and trigger types. Such common characteristics help identify relevant research gaps by comparing to what extent performance studies evaluate characteristics common in real-world serverless applications (see Section 1.9.1). These research gaps motivated and guided further performance studies in Papers γ to ε. Most importantly, the empirical insights of this paper were essential for designing a realistic benchmark suite in Paper γ.

1.6.γ Serverless Application Benchmark

Context Most serverless performance studies focus on single-purpose micro-benchmarks that are not representative of real applications. Existing application benchmarks are insufficient because they are typically limited to single-function applications using at most one type of external service. Most importantly, no prior application benchmark includes asynchronously coordinated applications although serverless is inherently event-based and event-driven architectures are common in real-world applications.

Contribution Paper γ presents ServiTrace, a comprehensive benchmark suite of 10 heterogeneous applications with support for fine-grained tracing and invocation patterns derived from real-world invocation logs. It introduces a novel algorithm and heuristics for detailed latency breakdown analysis of asynchronously coordinated applications across a variety of external services. Using ServiTrace, a large-scale empirical performance study was conducted in the market-leading AWS environment, collecting over 7.5 million traces. The novel *latency breakdown analysis* enabled detailed insights into median latency, cold starts, and tail latency for different application types and invocation patterns. Finally, ServiTrace is released as a tested, extensible open-source tool under FAIR principles including software, data, results, and documentation.

Method ServiTrace was designed and implemented using engineering research (Section 1.5.3) based on insights from real-world application characteristics in Paper β and guided by goals from the literature review in Paper α. The field experiment (Section 1.5.4) follows guidelines on benchmarking [127] and reproducible cloud experimentation [46].

Relationship to Thesis Beyond essential methodological guidance of prior work from Papers α and β, the results of this paper motivate further research to extend ServiTrace and analyze important aspects for application performance in more detail. CrossFit in Paper δ extends ServiTrace with fair cross-provider comparison and disconnected trace correlation by refining a specific application scenario from this paper. TriggerBench in Paper ε specifically analyzes the latency of serverless function triggers for different external services because the results of this paper have shown that slow trigger latency can add substantial latency delay.

1.6.δ Cross-provider Application Benchmarking

Context Fair cross-provider comparison is challenging in serverless applications due to heterogeneous complex ecosystems. FaaS platforms are highly provider-specific and lack standardized interfaces such as VMs for IaaS. Furthermore, FaaS platforms are not intended as standalone systems but rather deeply integrated with other provider-specific external services through integrations with event sources. Another challenge is that existing comparisons provide no observability on why performance differs because they only compare overall response times.

Contribution Paper δ contributes CrossFit. It introduces a provider-independent tracing model for serverless applications, provides guidelines for fair cross-provider benchmarking, and demonstrates detailed drill-down analysis for an application in two leading cloud providers. The tracing model identifies matching trace points available in multiple providers. The fairness guidelines cover 12 important aspects related to architecting applications for fair performance comparison across cloud providers. The drill-down analysis identifies performance challenges for a realistic application under different cloud providers and workloads.

Method Engineering research (Section 1.5.3) was used to refine an application from ServiTrace in Paper γ and port it to another cloud provider inspired by prior work on serverless application migration [142]. A field experiment (Section 1.5.4) subsequently demonstrated the utility of the tracing model for cross-provider drill-down analysis using constant and bursty workloads.

Relationship to Thesis This paper addresses several research gaps identified in the literature review in Paper α such as cross-provider application benchmarking and insufficiently studied platform configurations. It also leverages ServiTrace from Paper γ to alleviate reproducibility challenges.

1.6.ε Serverless Function Trigger Benchmark

Context Function triggers are essential building blocks in serverless, as they initiate any function execution. However, Paper α shows that function triggering is insufficiently studied despite being a core building block of serverless applications in practice, as shown in Paper β. Additionally, function triggering is inherently hard to measure given the distributed, ephemeral, and asynchronous nature of event-based function coordination.

Contribution Paper ε introduces TriggerBench, a cross-provider benchmark for evaluating serverless function triggers based on distributed tracing. It describes a measurement methodology for synchronous and asynchronous function triggers and supports trace correlation of disconnected partial traces. TriggerBench implements three of the most popular trigger types [β] in AWS and Microsoft Azure [143], namely HTTP, storage, and queue triggers. The Azure implementation supports the following additional trigger types: database, event, stream, message, and timer.

Method TriggerBench was developed with engineering research (Section 1.5.3) building upon ServiTrace introduced in Paper γ. Subsequently, TriggerBench was used in a benchmarking field experiment (Section 1.5.4) to evaluate a total of 11 trigger types in two cloud providers.

Relationship to Thesis This paper addresses the research gap of insufficiently studied triggers raised by Paper α, especially across cloud providers. It is also motivated by the results on poor trigger performance in Papers γ and δ. It builds upon ServiTrace from Paper γ and generalizes trace correlation of disconnected traces first demonstrated in Paper δ.

1.6.ζ Integrated Cloud Benchmark Suite

Context In contrast to the more recent trend (starting 2015) of serverless performance evaluation (Section 1.2.1), the performance of IaaS clouds has been extensively studied for over a decade (starting 2008) using micro- and application-level benchmarks (Section 1.2.5). However, existing work largely focuses on evaluating performance benchmarks in isolation without systematically combining multiple types of performance benchmarks and often comes with several reproducibility challenges [46, 93].

Contribution Paper ζ addresses this gap by presenting an execution methodology that combines micro- and application-benchmarks into a new benchmark suite, integrating this suite into an automated cloud benchmarking framework, and implementing a repeatable execution methodology proposed in related work (Section 1.2.6). The execution methodology was instantiated in the AWS EC2 cloud and the paper presents selected results related to cost-performance efficiency, network bandwidth, and disk utilization.

Method Based on cloud benchmarking guidelines [31, 81, 82, 144], relevant benchmarks that cover different cloud resources and application domains were selected, designed, and integrated into the CWB [97] execution framework. The execution of these benchmarks was then automated following the RMIT execution methodology proposed by Abedi and Brecht [93].

Relationship to Thesis IaaS provides the underlying infrastructure of serverless computing platforms covered in Papers α to ε. Therefore, its performance characteristics are also relevant and similarly evaluated by micro- and application-benchmarks. Moreover, certain aspects are hard to measure within restricted serverless platforms (e.g., due to execution time limits and unavailable direct communication).

Paper ζ layed the methodological and technical foundations for the follow-up study in Paper η. The ability to systematically collect performance measurements for multiple benchmarks enables the investigation of benchmark correlations under different configurations to support cloud service selection.

1.6.η Cloud Application Performance Estimation

Context The continuing growth of the cloud computing market has led to an unprecedented diversity of cloud services. To support service selection, micro-benchmarks are commonly used to identify the best-performing cloud service. However, it remains unclear how relevant these synthetic micro-benchmarks are for gaining insights into the performance of real-world applications.

Contribution Paper η describes a cloud benchmarking methodology for estimating application performance based on micro-benchmark profiling, evaluates this methodology in an IaaS cloud provider, and releases a dataset for micro- and application-benchmarks of over 60 000 measurements from over 240 virtual machines across 11 distinct virtual machine types.

Method A field experiment in the AWS EC2 cloud environment quantified performance variability and evaluated the proposed methodology. A linear regression model was trained across 11 VM instance types using 38 metrics from 23 micro-benchmarks and evaluated in terms of relative error for two applications from different domains. To select the most relevant estimators, forward feature selection was used to identify the most useful micro-benchmarks and compare them against three common baselines.

Relationship to Thesis This paper uses the benchmark suite from Paper ζ to evaluate the idea of using synthetic resource-specific micro-benchmarks to estimate the performance of application-benchmarks inspired by real-world scenarios. It bridges the gap between ubiquitous micro-benchmarks (as shown by Paper α) and application benchmarks (as proposed in Paper γ and extended in Papers δ and ε). Although this papers targets IaaS, optimal VM instance type selection seems transferable to serverless function size selection as demonstrated in Sizeless [145] and SAAF [146] for FaaS and discussed in the threats to external validity (Section 1.8.5.3).

1.6.θ Software Microbenchmarking in the Cloud

Context The availability of seemingly infinite resources and on-demand elasticity makes public clouds attractive for software performance testing as an alternative to traditional controlled bare-metal environments. However, massive multi-tenant public cloud environments are susceptible to stochastic variation caused by noisy neighbors and potentially other opaque performance changes (e.g., hardware and software updates, network instabilities). Therefore, it remains unclear how suitable inherently unstable cloud environments are for software microbenchmarks and to what extent slowdowns can still be reliably detected.

Contribution Paper θ quantifies the effect of cloud environments on the variability of software performance tests and explores their reliability in detecting slowdowns. Performance variability is reported as the coefficient of variation for 19 performance tests in 9 cloud execution environments. Further, drill-down analysis reveals different sources of variability (i.e., benchmark vs. trial vs. total). For reliable slowdown detection, this paper compares two execution strategies and two statistical tests in terms of their false positive rate and minimal-detectable slowdown.

Method A large-scale field experiment (Section 1.5.4) collected over 4.5 million microbenchmark results across three cloud providers, three classes of instance types, two programming languages, 19 software microbenchmarks, and two execution strategies. Two standard statistical tests were used to detect software performance changes (i.e., A/B test) and investigate false positives (i.e., A/A test), namely Wilcoxon Rank-sum and overlapping confidence intervals. An experimental simulation explores minimal-detectable slowdowns through simulated performance regressions (i.e., slowdowns) without exceeding a 5 % false positive threshold during A/A testing.

Relationship to Thesis The surprisingly stable performance results from Paper η in comparison to prior work motivated this more in-depth study about software performance testing because predictable system performance is essential for efficient performance testing. Additionally, the toolkit for conducting cloud experiments in this study builds upon the Cloud WorkBench [97] extensions and the experiment scheduling methodology from Paper ζ. Finally, performance variability and reliability are important for mitigating reproducibility challenges, as discussed in Paper α for FaaS and in related work for IaaS [46].

1.7 Results

This section answers the research questions and summarizes solutions to the challenges raised in Section 1.4.

1.7.1 Current State of Serverless (RQ1)

RQ1 *What is the current state of serverless applications and their performance?*

Answer: Synthetic micro-benchmarks have been studied extensively but the serverless community lacks a detailed performance understanding of realistic applications that integrate with external services.

Landscape of serverless performance evaluation The review of 112 performance evaluation studies in Paper α found that AWS Lambda is the most evaluated FaaS platform (88 %), that micro-benchmarks are the most common type of benchmark (75 %), and that application-benchmarks are prevalently evaluated on a single platform. It also indicates a broad coverage of language runtimes but shows that other platform configurations focus on very few function triggers and external services.

Serverless application characteristics The analysis of 89 serverless applications in Paper β has shown that serverless is adopted to save costs for irregular or bursty workloads, avoid operational concerns, and for built-in scalability. Serverless applications are most commonly used for short-running tasks with low data volume and bursty workloads but are also frequently used for latency-critical, high-volume core functionality. Serverless applications are mostly implemented on AWS, in either Python or JavaScript, and make heavy use of external services for persistence and coordination functionality.

1.7.2 Serverless Application Performance (RQ2)

RQ2 *What are the performance challenges of serverless applications?*

Answer: External service calls and trigger-based function coordination are slow and suffer from long tail latency.

Serverless application benchmark Paper γ contributes a heterogeneous, representative, reproducible, and extensible application benchmark that implements end-to-end functionality and provides detailed insights through distributed tracing. The application benchmark is the most diverse to date by covering different external services, function triggers, programming languages, coordination architectures (synchronous and asynchronously), and application types. A novel algorithm and heuristics support tracing of asynchronous event-driven serverless architectures. The implementation is well-tested, has processed over 7.5 million traces, demonstrated its automated capabilities in long-running experiments over days and weeks, and has been used by several master thesis projects.

Fine-grained performance insights for serverless applications Performance experiments in Paper γ show that the median and 99th percentile of end-to-end application latency is often dominated by external service calls rather than computation. Some of the largest delays are caused by function triggers. Their performance differs substantially across cloud providers (Paper δ) and can add multi-second delays for asynchronously coordinated applications (Paper ε). Coldstart overhead is not limited to container initialization (i.e., the time to provision a new function instance) for serverless applications. In comparison, language runtime initialization adds more overhead and other factors such as one-off computation tasks can also contribute substantially.

1.7.3 Limitations of Cloud Benchmarking (RQ3)

RQ3 *How can limitations of benchmarking cloud infrastructure be addressed?*

Answer: Only selected micro-benchmarks are suitable for estimating application performance, performance variability depends on the resource type, and batch testing on the same instance with repetitions should be used for reliable performance testing.

Relationship between micro- and application-benchmarks The systematic combination of micro- and application-benchmarks in Papers ζ and η has shown that selected micro-benchmarks are better in estimating application performance than specification-based metrics. However, micro-benchmarks cannot necessarily be used interchangeably even if they seemingly test the same resource because benchmark parameters can have a profound impact.

Performance variability for different benchmark types Extensive performance experiments in Papers ζ to θ have shown that performance variability depends on the resource type and cloud provider but can also be caused by unstable benchmarks, which should be avoided for performance testing. For comparing alternative versions, Paper θ shows that batch testing (i.e., trial-based) significantly reduces false-positive rates and the number of repetitions required to reliability detect performance changes compared to version testing (i.e., instance-based).

1.8 Discussion

This section discusses the main findings and implications of this thesis for research and industry.

1.8.1 Serverless Observability

This thesis emphasizes the importance of detailed tracing and trace analysis for actionable insights into complex architectures for serverless applications. Without tracing, the client-side response time might cover the most latency-critical synchronous invocations but misses any asynchronous event-driven backend processing. Detailed tracing provides observability into the end-to-end lifecycle of a request and supports root cause analysis of performance challenges. Without detailed tracing, time-intensive experimentation of many configurations is required in an attempt to isolate a performance issue. Therefore, observability should provide application-level insights enriched with system-level information to comprehensively understand serverless performance. The need for some system-level performance information might sound counter-intuitive in the inherently opaque serverless paradigm. However, providers should offer APIs to expose certain performance-relevant information such as container and runtime initialization times to optimize coldstarts. An initial academic approach to serverless observability is limited to FaaS [147] but several companies aim to offer full end-to-end observability, including Lumigo[7], Epsagon[8], Dashbird[9], Thundra[10], and several others [148].

Tracing in serverless and cloud computing is complex and trace analysis raises "a new big data problem for software engineering" [65]. Current commercial solutions focus on collecting monitoring data but still face many challenges before advancing to generate better insights and optimization recommendations. An industrial interview study [65] revealed major difficulties in trace data quality and missing trace annotations. ServiTrace faced the same issues due to limitations in AWS X-Ray. Therefore, Paper γ proposes enhanced trace annotations to indicate the invocation type (i.e., synchronous or asynchronous). Such annotations enable more robust trace analysis than heuristics, which cannot detect asynchronous invocations that terminate before their invoker. The tracing community also discusses such annotations to enhance the specification of the OpenTelemetry [149] standard. To mitigate bad trace data quality, Paper γ validates each trace and reports missing or inconsistent fields (e.g., if the sum of a trace duration does not match the elapsed time between two timestamps). Disconnected traces are another issue in serverless due to unsupported trace token propagation for several external services. Paper ε demonstrates a solution to merge disconnected traces to enable end-to-end analysis. Finally, trace analysis only just started to develop and more robust and intelligent techniques are needed.

Serverless tracing comes with several limitations related to semantic challenges and tracing overheads. The event-driven serverless architecture causes semantic challenges related to passive tracing and batch execution. Asynchronous event-based triggers are often traced passively (or indirectly) in contrast to actively traced compute services such as FaaS. Tracing services such as X-Ray implement reparenting strategies to build a properly connected trace graph. However, such strategies can fail in rare cases and cause erroneous traces. Batch execution is a common feature for serverless queue or stream processing services. It introduces a one-to-many mapping, which violates the assumption that each span can only have one parent. Therefore, batch receiving remains an unsolved semantic challenge in the OpenTelemetry tracing community [149]. Tracing overhead has a limited impact on application performance but causes additional processing and storage demands. Tracing might add additional

[7]`https://lumigo.io/`
[8]`https://epsagon.com/`
[9]`https://dashbird.io/serverless-observability/`
[10]`https://thundra.io/`

coldstart overhead caused by tracing libraries and background initialization but it is often negligible during runtime when implemented asynchronously. The main overhead is caused by dedicated tracing services, which need to scale themselves for supporting high loads. To mitigate this overhead, sampling strategies (e.g., fixed rate or reservoir) can be used to limit ingestion rates and short retention periods (e.g., 30 days) to cap aggregated trace data.

1.8.2 Interactive Applications with Serverless

User-centric performance models describe the human perception of performance based on user experience research. According to long-standing research [150, Chapter 5], the guidelines related to human perception of performance remain the same since they were first formalized in 1968 [151]. These performance thresholds are determined by human perceptual abilities and interpreted for modern (web) applications by Nielsen [152, 153] and the RAIL model from Google [154]. Users perceive reactions below 0.1 s as immediate. Reactions below 1 s cause noticable delay but are not yet interrupting the flow of thought. Users lose attention for reactions beyond 10 s and seek alternative tasks.

This thesis shows that latency-sensitive applications are feasible with serverless but face many challenges, such as appropriate service selection. For example, interactive applications should adopt the synchronous HTTP trigger, choose a provider and language runtime with low coldstart overhead, and perform trace analysis to optimize slow application segments. Paper ε shows that HTTP triggers in multiple providers can achieve sub-0.1 s latency but Paper γ indicated that any external data service (e.g., Amazon S3) likely exceeds the limit for immediate response. Therefore, today's serverless offerings struggle with interactive applications unless data is served from low-latency caches or data services are exposed directly rather than hidden from the user behind a serverless function. Most user-facing applications cause noticeable delay but serverless enables building highly scalable applications within the 1 s limit. For example, BBC Online demonstrated that navigating to different pages in a personalized server-side rendered website is doable within 220 ms (90^{th} percentile) while running 100 million function invocations daily [155]. Nevertheless, the latency breakdown in Papers γ and δ indicates that end-to-end latency quickly adds up, and especially tail latency is challenging to control. Further, the choice of appropriate function triggers and external services is essential, not only for maximizing performance but also for building cost-effective applications under specific performance requirements. A related study demonstrated massive differences in their cost-performance comparison of alternative mechanisms for serverless function coordination [156]. Finally, Paper β indicates that other factors such as cost can be more important than performance for certain classes of applications.

1.8.3 Reproducibility Challenges in Cloud Performance

This section discusses challenges and mitigation strategies for reproducible performance evaluation in cloud environments based on the methodological principles proposed by Papadopoulos et al. [46] (summarized in Section α.5.5). This discussion enriches observed challenges from the literature review in

Paper α with challenges faced in engineering research and field experimentation in Papers γ to θ.

P1 *Repeated Experiments:* Ideally, every experiment configuration should be repeated many times until a statistically sound stopping criterion [157] is reached. Unfortunately, the nature of field experimentation in unstable cloud environments [87, 91–93] can make it hard to reliably detect performance changes within reasonable budget and time constraints, as shown for certain configurations in Paper θ. Further, repetitions can be implemented at different levels as exemplified in Papers ζ to θ and scheduling strategies profoundly impact the utility of different types of repetitions (e.g., trials, forks, iterations/executions) as shown in Paper θ. Robust statistical methods using hierarchical bootstrapping can be computationally intensive and are not (yet) widely known.
Beyond the statistical aspect, ensuring configuration equivalence across repetitions is hampered by incomplete experimental setup descriptions (P3) and lacking automation. The nature of field experimentation (Section 1.5.4) makes an exact replication of the measurements rarely possible but *technical reproducibility* is desired to be able to repeat the exact same measurement methodology [46]. Paper α shows that incomplete experimental setup descriptions make it hard to repeat any experiment. For example, the integration of existing applications into ServiTrace in Paper γ demonstrated that unpinned dependencies change or even break the experiment setup. Furthermore, reusable benchmarks should prevent naming conflicts due to global namespaces to support repeated experiments in different contexts (e.g., different tenants, data-center regions). Experiment designs with dynamic re-deployments require a high level of experiment automation and are essential for measuring coldstarts more reliably and efficiently rather than waiting for undocumented idle timeouts. Therefore, ServiTrace strives for full experiment automation by supporting executable experiment plans used in Papers γ to ε.

P2 *Workload and Configuration Coverage:* Papers α and β collect empirical evidence to motivate workloads and configurations for Papers γ to ε in terms of applications, cloud providers, programming languages, external services, trigger types, control flow (synchronous vs. asynchronous), and invocation patterns (e.g., bursty). Overall, Papers γ to ε focus on covering application aspects, Papers ζ and η cover a broad range of system-level resources through micro-benchmarks, and Paper θ covers software microbenchmarks from popular projects in multiple cloud providers.

P3 *Experimental Setup Description:* Only about half of the studies provide a sufficient experiment setup description, both in serverless performance evaluation as shown in Paper α and in cloud infrastructure performance evaluation [46]. To mitigate this issue, a replication package is available for every paper, and experiment plans are made executable. A replication package complements a paper with a detailed experiment description, instructions on how to replicate each experiment to obtain a new dataset with the same methodology as well as replication of the data analysis based on a documented dataset. Such a full experiment description

Table 1.3: Overview of open access artifacts.

Paper	**Code (Github)**	**Dataset**
α	joe4dev/faas-performance-mlr	[158]
β	ServerlessApplications/ReplicationPackage	[159]
γ	ServiTrace/ReplicationPackage	[160]
δ	serverless-crossfit/replication-package	[161]
ε	joe4dev/trigger-bench	[162]
$\zeta + \eta$	sealuzh/cwb-benchmarks	[163]
θ	sealuzh/cwb-benchmarks	[164]

provided in a replication package is often not possible nor desired within a paper due to space constraints and restricted presentation formats. This thesis strives for fully automated experiments to minimize manual steps, which are prone to human error and often incomplete. Nevertheless, some manual steps are often necessary for bootstrapping or security-sensitive tasks.

P4 *Open Access Artifact:* All papers in this thesis are complemented with technical artifacts including datasets (raw and processed) and software for experiment orchestration, benchmarks, and data analysis. Table 1.3 summarizes the open access artifacts produced in this thesis. Unit and integration tests are also provided for core functionality such as trace analysis or experiment orchestration.

P5 *Probabilistic Result Description:* This thesis favors plots that visualize the full empirical distribution such as violin plots and empirical cumulative distribution function (ECDF) plots. Otherwise, robust aggregations with respect to outliers are used by representing typical latency as median (p50) and tail latency as 99^{th} percentile (p99), with one exception using average aggregation in Paper η. Due to space constraints, additional plots and statistics are sometimes provided as part of a replication package. Papers η and θ specifically investigate performance variability by reporting the coefficient of variation and Paper θ performs A/A tests to evaluate false positive rates of software microbenchmarks in cloud environments.

P6 *Statistical Evaluation:* In the context of this thesis, statistical evaluation is most relevant for comparing alternative versions of software microbenchmarks in Paper θ. The evaluation of A/A tests with Wilcoxon rank-sum and overlapping confidence intervals using hierarchical bootstrapping [165, 166] has shown that Wilcoxon is more sensitive towards changes in the tested configurations. For other aspects in the thesis such as cross-provider comparisons in Paper δ, visualizing differences in the result distributions is often more insightful than reporting a binary outcome of a statistical test. Automation enables collecting large sample sizes, which might lead to statistically significant differences although the practical difference might be negligible and distribution characteristics such as tail latency are more relevant. Paper δ uses split violin plots to compare two distributions. Alternative options are the shift function [167], ratio function [168], or nonparametric Cohen's d-consistent effect size [169].

P7 *Measurement Units:* This thesis consistently reports measurement units and Paper α finds that this principle is generally followed in FaaS performance studies with only a few exceptions in grey literature figures.

P8 *Cost:* Reporting a conceptual cost model based on individual service pricing should be generally possible but is often incomplete because some cost factors are determined at runtime (e.g., based on memory consumption or execution time). Reporting actual costs is often difficult when running multiple services or using research credits.

1.8.4 Cross-Provider Portability

The portability of applications across providers remains a major challenge in serverless, which requires trade-off decisions as discussed in Paper δ. Unlike in IaaS where the standardized VM abstraction enables full code reuse across providers, serverless APIs are highly provider-specific, both for source code as well as for deployment options (e.g., memory size, shared storage layers, provisioned concurrency). Prior work confirms this issue in a migration study of multiple applications [142] and in a developer survey [56] where one-third of the respondents mentioned vendor lock-in as a significant challenge. Existing solutions are limited to specific domains such as data analytics through Lithops [170, 171] or simple single-function scenarios [172]. Vendor lock-in also remains one of the core obstacles for multi-cloud approaches [173]. Due to this lack of a common interface, it is not possible to implement a single provider-agnostic benchmark. Therefore, cross-provider support requires careful application migration [142] as discussed in Section δ.3.2 and demonstrated with the trigger types mapping of external services in Section ε.2.2.

1.8.5 Threats to Validity

This section discusses threats to the validity of the results of this thesis, limitations of the applied research methods, and a summary of mitigation strategies. It is structured based on the four common criteria for validity for empirical research [123, 174]: construct validity, internal validity, external validity, and reliability.

1.8.5.1 Construct Validity

Construct validity relates to *measuring the right thing*, i.e., the extent a study actually measures what it aims to measure according to the research questions.

For RQ1, construct validity mainly relates to inappropriate selection criteria and a lack of standard language and terminology. To mitigate these threats, the selection criteria were refined based on related work and documented insights from trial classifications. The lack of standard language is a major threat as there exist no established definitions of FaaS and serverless [19]. This threat was mitigated by clarifying selected definitions and providing illustrational examples where applicable.

For the field experiments in RQ2 and RQ3, construct validity in benchmarking is the threat to test or measure something different than intended. Performance benchmarking is "incredibly error prone" [130], especially in cloud environments. Therefore, this thesis performs active benchmarking [175] and focuses on reproducible experimentation. Active benchmarking uses observability tools to analyze performance while the benchmark is running to collect evidence that the benchmark tests what it intends to test. For example, Paper γ performs workload validation to compare the planned vs. sent vs. received invocation rates and uses detailed tracing to explain and validate end-to-end latency results. Another example includes resource monitoring in Papers ζ and η as demonstrated in Figure ζ.6 by utilization rate monitoring during I/O benchmarking. Reproducible experimentation (see Section 1.8.3) encourages a complete reporting of the experimental setup, which enables a thorough review of the experiment design.

1.8.5.2 Internal Validity

Internal validity relates to *measuring right*, i.e., the extent a study measures a causal relationship without interference from external factors.

For RQ1, the most common threats in literature reviews are bias in study selection, bias in data extraction, and inappropriate or incomplete database search terms. To mitigate selection bias, Paper α combines and refines [120] different established search strategies, and complements them with targeted strategies (e.g., alert-based search to discover recent studies). Search terms were iteratively refined and motivated in detail (see replication package [158]). Potential inaccuracies in data extraction were mitigated through traceability with over 700 additional comments and a well-defined MLR process based on established guidelines for SLR [118] and MLR [119] studies, methodologically related publications [176], and topically relevant publications [51, 53]. The main threat remains individual researcher bias as the majority of studies were reviewed or validated by a single researcher.

For RQ1, a sample study has inherent limitations in measurement precision due to its neutral setting and lack of interactivity (i.e., research must deal with discoverable data *as is*) [115]. To mitigate this threat, each serverless application in Paper β was reviewed by two researchers and after an initial moderate agreement, all differences were discussed and consolidated. The lack of interactive data collection could only be mitigated partially through explorative web search and backward snowballing for discovering new sources. Reviewers assigned the "Unknown" value to applications and characteristics where insufficient information was available. These "Unknowns" are excluded in the presented results (ranging from 0 % to 19 % with two outliers at 25 % and 30 %) and reported in the accompanying replication package [159].

For RQ2 and RQ3, cloud experimentation is inherently susceptible to confounding factors as a field experiment due to its natural setting [115]. Public clouds cannot be under full control of an experimenter but appropriate execution methodologies as used in RQ3 can mitigate this threat. Further mitigation includes careful experimental design based on cloud experimentation guidelines [30, 31, 46] and fully automated experiment execution [97]. For RQ2, the limited access to serverless infrastructure impedes detailed tracing and provider-internal tracing is sometimes impossible to validate independently. In some cases, inquiry with providers is essential to clarify potential inconsistencies. Another source of inconsistencies concerns clock synchronization common to distributed systems [177], both in terms of precision and accuracy. To mitigate this threat, trace analysis in Paper γ combines an error margin for timestamp comparison with logical trace validation of causal relationships. Finally, test suites of unit and integration tests are integrated into continuous integration pipelines to mitigate implementation errors.

1.8.5.3 External Validity

External validity relates to *generalizability*, i.e., the extent the results of a study can be transferred to other contexts.

For RQ1, the literature review (Section 1.5.1) was designed to systematically cover the field of FaaS performance benchmarking for peer-reviewed academic literature (i.e., white literature) and unpublished grey literature including preprints, theses, and articles on the internet. The inclusion of grey literature targets an industrial perspective but is limited to published and indexed content freely available and discoverable on the internet (e.g., excluding paywall articles or internal corporate feasibility studies). For RQ1, the sample study (Section 1.5.2) aims to collect a diverse collection of realistic serverless applications. Therefore its sampling strategy combines purposive sampling from different sources with snowballing. About half of the serverless applications are used in production and about half of them are open source, but only few of them are both used in production and open source. Generalizability of the results cannot be claimed to all serverless applications, in particular not for private serverless applications.

For RQ2 and RQ3, field experimentation inherently lacks statistical generalizability [115]. Thus, generalizability cannot be claimed beyond the specific study settings. RQ2 covers a wide variety of serverless applications and external services guided by the insights from RQ1 but does not include data-analytic applications [178–181] and serverless-optimized machine learning applications for training [182] and serving [183]. Nevertheless, the trace analysis proposed in Paper γ is generic for serverless and the trace analyzer is directly applicable to production applications instrumented with AWS X-Ray.

RQ3 highlighted differences in performance variability across three major cloud providers but the cross-VM performance estimation approach in Paper η was demonstrated for two geographically distinct data centers of a single cloud provider. Although related work also focuses almost exclusively on AWS as a single cloud provider, another study [102] indicated that a similar methodology can also work across multiple cloud providers. This is unsurprising given that most IaaS clouds build upon the same abstractions (i.e., virtualization technology) and individual benchmarks within the benchmark suite were previously used across four different cloud providers [91] with the same benchmark manager [97]. A related study published shortly after Paper η reports comparable results for scientific computing workflow applications [104].

Newer related studies also indicate that the results from IaaS are applicable to FaaS. For example, Sizeless [145] uses multi-target regression modeling to predict the execution time of a serverless function for all memory sizes.

BATCH [183] uses simple regression and proposes an analytical model to predict latency percentiles. COSE [184] uses Bayesian Optimization to find the optimal configuration. Further, Wang et al. [185] indicated that the underlying hardware infrastructure of AWS Lambda shares the same specifications as VM instance types evaluated for answering RQ3. When transferring the prediction approach to serverless, the configuration space becomes larger as envisioned for tailorable VM instance types because serverless functions of certain providers offer fine-grained memory configurations (e.g., up to 10 240 MB in 1 MB increments for AWS Lambda), which determine the CPU power. Conversely, function runtime prediction is less important in serverless because brute-force approaches such as AWS Lambda Power Tuning [186] are readily available and more viable with the fast elasticity of serverless.

1.8.5.4 Reliability

Reliability relates to *replicability by others*, i.e., the extent to which the results of a study can be replicated by other researchers.

For RQ1, structured review sheets with actionable guidance were used and published in online replication packages [158, 159]. The sample study alleviated subjective interpretation of the extracted data through multiple reviews from a total of seven reviewers. Bi-lateral and group discussions were an important part of the data consolidation process and captured through systematic spreadsheet commenting and meeting notes. The literature review mitigated this threat through detailed documentation and traceability annotations.

For RQ2 and RQ3, the field experiments strive for *technical reproducibility* [46] of the data collection and *replicability* [187] of the data analysis based on documented online replication packages. Technical reproducibility enables other researchers to conduct the same experiment and collect a new dataset representing the current state of performance because the exact reproduction of the measurement results is impossible in cloud experimentation due to limited control over the environment [46]. Such a new dataset will be subject to internal changes of the cloud provider, which continuously updates underlying software and hardware infrastructure. Therefore, it is essential to additionally provide the raw dataset and analysis scripts for independent inspection. All performance benchmarks are available as open source software together with extensive documentation, test suites, and scripts to automate their execution. The benchmark orchestration tools ServiTrace (Paper γ) and Cloud WorkBench (CWB) [97] are purposefully built for technically reproducible cloud performance evaluation and used in other studies beyond this thesis [97, 188, 189].

The data analysis process strives for replicability [187] based on documented online replication packages providing datasets and analysis code. The ability to re-run (R^1 as introduced by Benureau and Rougier [187]) the code is facilitated by automation and dependency management but could be further improved by adopting Docker containerization [190], similarly to ServiTrace for benchmark orchestration. Repeatability (R^2) requires repeated code executions to produce the same expected results [187] and was validated by managing interim data with version control. Reproducible (R^3) results require other researchers to be able to re-obtain the same result [187] and are fostered by publicly available data and code under version control but could also be improved by adopting Docker containerization [190]. Reusability (R^4) is addressed by documentation and testing by collaborators but hampered by using a commercial analysis tool in Paper η. Replicability (R^5) refers to the ability of independent investigators to obtain the same results without re-using the technical artifacts [190] and was partially addressed by re-implementing parts of the analysis in another tool for validation purposes in Paper η.

1.9 Future Work

This section discusses future work in serverless performance evaluation, trace analysis, and approaches towards automated performance optimization.

1.9.1 Relevant Gaps in Serverless Performance Evaluation

Combining the results of Papers α and β reveals similarities and differences between the *evaluated* characteristics by performance studies and the *actual* characteristics of serverless applications. Overall, cloud providers and programming languages are represented similarly in terms of relative frequency except for under-represented language runtimes in academic studies. The clearest differences occur between function triggers and external services. Most notably, performance studies seldomly use event-based triggers (<16 %) in applications although cloud events are common in serverless applications (41 %). Further, most external services are under-represented in performance studies and databases in particular as they are used in 10 % to 15 % of the academic and industrial studies although being used by 48 % of applications. The only external services well-covered by academic studies are the API gateway and cloud storage. More studies are needed to test common external services such as publish/subscribe, streaming, and queues.

The field of serverless performance evaluation remains very active since the literature review in Paper α and several of the mentioned research gaps received more attention. Within the two years since the literature review in Paper α, the number of potentially related studies more than doubled[11] and complementary literature reviews have been published afterwards [191, 192]. For example, Raza et al. [191] discuss FaaS measurement studies from a developer's perspective and explicitly categorize studies by causal relationships, i.e., the relationship between the controlled variable (configuration) and dependent variable (measured performance). There are promising signs that additional providers are covered, including hosted platforms such as Alibaba [193, 194], open-source platforms such as Knative, Kubeless, OpenFaaS [195, 196], and open-source platforms used in hosted platforms such as Firecracker [59, 197] and OpenWhisk [198, 199]. However, these studies still focus on micro-benchmarks without covering external services with a few exceptions for specific domains such as workflows [194] or parallel data processing with Lithops [170, 171]. Finally, another recent publication trend is the rising interest in performance evaluation for edge computing platforms.

[11] Based on the updated list of 139 new studies identified through alert-based complementary search as described in Section 1.5.1

1.9.2 Serverless Trace Analysis

Traditional distributed tracing focused on microservice architectures with synchronous remote procedure calls (RPCs) and serverless architectures raise novel challenges with asynchronous invocations. Despite its usefulness, tracing still suffers from many challenges related to collection, analysis, and visualization [35, 65, 66]. Serverless aggravates existing challenges and introduces novel conceptual challenges. According to a recent interview study [65], bad trace quality is a common issue and becomes even harder to manage in serverless due to the lack of control over parts of the serverless and tracing infrastructure, as experienced in Papers γ to ε. Currently, manual instrumentation and custom trace correlation are required to fix disconnected traces and sampling is necessary for high invocation rates to prevent missing trace segments, which cause incomplete traces that need to be ignored. Therefore, future research should explore more robust methods for handling bad quality traces during trace analysis. Tracing standards such as OpenTelemetry[12] deserve further attention and guidance, for example by suggesting useful trace annotations. In traditional synchronous invocation chains, every trace segment can have at most one casual parent relationship (i.e., invoked by). In serverless architectures, batch invocations violate this assumption and require novel tracing concepts (e.g., batch receiving [149]).

1.9.3 Automated Performance Optimizations

The research of this thesis focuses on evaluating (i.e., assessing) performance but future work could go a step further by leveraging the methodology and insights from this thesis to automate the exploration and exploitation of the configuration space towards self-optimizing applications. A key motivation for this research direction is to make performance insights more actionable in the context of application development through tighter integration of performance aspects into the development lifecycle. My vision paper [200] outlines a dynamic transpilation approach and FUSIONIZE [201] explores feedback-driven function fusion at runtime to optimize the latency of multi-function workflows. Recently, new optimization approaches started to emerge for optimal function sizing [145], application-aware data passing [202], workload-specific configuration tuning for video processing [203], and a combination of several workflow tuning strategies [204]. In addition to function fusion, WISEFUSE [204] also optimizes resource allocation and co-locates parallel function invocations through bundling. Existing approaches focus on serverless functions but trace-based optimization suggestions could include external services, for example by recommending suitable trigger types.

1.10 Conclusions

This PhD thesis consolidated (RQ1) and extended (RQ2) the body of research on reproducible performance evaluation for serverless applications and their underlying infrastructure (RQ3).

[12]`https://opentelemetry.io/`

RQ1 This thesis established a consolidated understanding of serverless applications and their performance through a sample study and literature review. The most comprehensive literature review on FaaS performance evaluation to date found that AWS Lambda is the most evaluated FaaS platform, that micro-benchmarks are the most common type of benchmark, and that application-benchmarks are prevalently evaluated on a single platform. It also indicated a broad coverage of language runtimes but showed that other platform configurations focus on very few function triggers and external services. Finally, the majority of studies did not follow principles on reproducible cloud experimentation from prior work [46]. The largest analysis of serverless applications to date identified common performance requirements and other characteristics related to adoption and implementation. In particular, serverless applications are most commonly used for short-running tasks with low data volume and bursty workloads but are also frequently used for latency-critical, high-volume core functionality.

RQ2 This understanding guided the construction of ServiTrace, a novel trace-based benchmark for serverless applications, which is used in field studies to identify performance challenges of serverless applications. ServiTrace contributes a novel algorithm and heuristics for detailed latency breakdown analysis of distributed serverless traces across asynchronous call boundaries and external services. Its comprehensive benchmark suite of 10 realistic open-source applications covers heterogeneous characteristics such as the form of coordination, programming language, size, and external service usage. Large-scale field experimentation in the market-leading AWS cloud environment has shown that external service calls often dominate the median end-to-end latency and cause excessive tail latency. Different forms of orchestration or trigger-based coordination caused substantial delay and were evaluated in further benchmarking experiments in addition to other aspects such as fair cross-provider benchmarking or different workload types.

RQ3 Targeting the underlying FaaS infrastructure in IaaS clouds, the utility of different benchmark types is evaluated in terms of insights for applications and reliability. Field experiments with system-level micro- and application-benchmarks and software microbenchmarks have shown that only selected micro-benchmarks are suitable for estimating application performance, performance variability depends on the resource type, and batch testing on the same instance with repetitions should be used for reliable performance testing. Benchmark-based metrics are better estimators for application performance of the tested applications than specification-based metrics (e.g., number of vCPUs, provider-defined unit for computational power), which are currently used as common baselines. However, the results also highlighted that presumably similar micro-benchmark estimators cannot necessarily be used interchangeably because benchmark parameters can have a profound impact on performance. The findings for software microbenchmarking indicate that state-of-the-art statistical tests (i.e., Wilcoxon rank-sum and overlapping bootstrapped confidence intervals of the mean) can reliably detect slowdowns in inherently unstable cloud environments but depending on the cloud provider and instance type, a substantial number of trials or instances is required. Further, batch testing might be required to detect small slowdowns reliably while avoiding false positives by co-locating the test and control group on the same instance.

Bibliography

[1] P. Mell and T. Grance, “The NIST definition of cloud computing,” National Institute of Standards and Technology (NIST), Tech. Rep., 2011. doi:10.6028/NIST.SP.800-145

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, and M. Zaharia, “Above the clouds: A berkeley view of cloud computing,” EECS Department, University of California, Berkeley, Tech. Rep., 2009. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[3] Gartner, “Gartner says worldwide iaas public cloud services market grew 40.7% in 2020,” 2021. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2021-06-28-gartner-says-worldwide-iaas-public-cloud-services-market-grew-40-7-percent-in-2020

[4] R. Bala, B. Gill, D. Smith, D. Wright, and K. Ji, “Magic quadrant for cloud infrastructure and platform services,” *Gartner*, 2021. [Online]. Available: https://aws.amazon.com/resources/analyst-reports/gartner-mq-cips-2021/

[5] Foundry, “Cloud computing study,” 2022. [Online]. Available: https://resources.foundryco.com/download/cloud-computing-executive-summary

[6] Markets and Markets, “Serverless architecture market,” 2020. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/serverless-architecture-market-64917099.html

[7] Allied Market Research, “Function-as-a-service market,” 2020. [Online]. Available: https://www.alliedmarketresearch.com/function-as-a-service-market-A06072

[8] Global Market Insights, “Serverless architecture market to exceed $30 bn by 2027,” 2021. [Online]. Available: https://www.globenewswire.com/news-release/2021/06/16/2247916/0/en/Serverless-Architecture-Market-to-exceed-30-Bn-by-2027-Global-Market-Insights-Inc.html

[9] Datadog, “The state of serverless,” 2021. [Online]. Available: https://www.datadoghq.com/state-of-serverless-2021/

[10] ——, “The state of serverless,” 2022. [Online]. Available: https://www.datadoghq.com/state-of-serverless/

[11] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds: Towards a cloud definition,” *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 50–55, 2008. doi:10.1145/1496091.1496100

[12] D. Hilley, “Cloud computing: A taxonomy of platform and infrastructure-level offerings,” Georgia Institute of Technology, Tech. Rep., 2009. [Online]. Available: http://www.cercs.gatech.edu/tech-reports/tr2009/git-cercs-09-13.pdf

[13] S. A. Ahson and M. Ilyas, *Cloud Computing and Software Services: Theory and Techniques.* CRC Press, Inc., 2010.

[14] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud computing: Principles and Paradigms.* John Wiley & Sons, 2011, vol. 87.

[15] S. Kächele, C. Spann, F. J. Hauck, and J. Domaschka, “Beyond IaaS and PaaS: An extended cloud taxonomy for computation, storage and networking,” in *Proceedings of the 6th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2013, pp. 75–2. doi:10.1109/UCC.2013.28

[16] J. Barr, “Amazon EC2 Beta,” Amazon Web Services, 2006. [Online]. Available: https://aws.amazon.com/blogs/aws/amazon_ec2_beta/

[17] R. S. Barga, “Serverless computing redefining the cloud,” Keynote of 1st International Workshop on Serverless Computing (WoSC), 2017. [Online]. Available: https://www.serverlesscomputing.org/wosc17/presentations/barga-keynote-serverless.pdf

[18] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, “Serverless programming (function as a service),” in *37th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2658–659. doi:10.1109/ICDCS.2017.305

[19] G. C. Fox, V. Ishakian, V. Muthusamy, and A. Slominski, “Status of serverless computing and function-as-a-service (FaaS) in industry and research,” *CoRR*, vol. abs/1708.08028, 2017. [Online]. Available: http://arxiv.org/abs/1708.08028

[20] S. Fink, “Serverless – where have we come? where are we going?” Keynote talk at the 3rd International Workshop on Serverless Computing (WoSC), 2018. [Online]. Available: https://www.serverlesscomputing.org/wosc3/presentations/keynote-Serverless0718.pdf

[21] P. C. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, “The rise of serverless computing,” *Communications of the ACM*, vol. 62, pp. 44–54, 2019. doi:10.1145/3368454

[22] M. Roberts and J. Chapin, *What is Serverless?* O’Reilly Media, Inc., 2017.

[23] M. Roberts, “Serverless architectures,” Martin Fowler, 2018. [Online]. Available: https://martinfowler.com/articles/serverless.html

[24] E. van Eyk, A. Iosup, S. Seif, and M. Thömmes, “The SPEC Cloud group’s research vision on FaaS and serverless architectures,” in *Proceedings of the 2nd International Workshop on Serverless Computing (WOSC)*, 2017, pp. 1–4. doi:10.1145/3154847.3154848

[25] S. Kounev, C. Abad, I. T. Foster, N. Herbst, A. Iosup, S. Al-Kiswany, A. A.-E. Hassan, B. Balis, A. Bauer, A. B. Bondi, K. Chard, R. L. Chard, R. Chatley, A. A. Chien, A. J. J. Davis, J. Donkervliet, S. Eismann, E. Elmroth, N. Ferrier, H.-A. Jacobsen, P. Jamshidi, G. Kousiouris, P. Leitner, P. G. Lopez, M. Maggio, M. Malawski, B. Metzler, V. Muthusamy, A. V. Papadopoulos, P. Patros, G. Pierre, O. F. Rana, R. P. Ricci, J. Scheuner, M. Sedaghat, M. Shahrad, P. Shenoy, J. Spillner, D. Taibi, D. Thain, A. Trivedi, A. Uta, V. van Beek, E. van Eyk, A. van Hoorn, S. Vasani, F. Wamser, G. Wirtz, and V. Yussupov, “Toward a definition for serverless computing,” in *Serverless Computing (Dagstuhl Seminar 21201)*, 2021, vol. 11, pp. 34–93. doi:10.4230/DagRep.11.4.34

[26] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, “Occupy the cloud: distributed computing for the 99%,” in *Proceedings of the 2017 Symposium on Cloud Computing*, 2017, pp. 445–451. doi:10.1145/3127479.3128601

[27] V. Persico, A. Montieri, and A. Pescapè, “On the network performance of amazon S3 cloud-storage service,” in *Proceedings of the 5th IEEE International Conference on Cloud Networking (Cloudnet)*, 2016, pp. 113–118. doi:10.1109/CloudNet.2016.16

[28] Z. Li, L. O’Brien, H. Zhang, and R. Cai, “On the conceptualization of performance evaluation of IaaS services,” *IEEE Transactions on Services Computing*, vol. 7, pp. 628–41, 2014. doi:10.1109/TSC.2013.39

[29] ——, “On a catalogue of metrics for evaluating commercial cloud services,” in *Proceedings of the 13th ACM/IEEE International Conference on Grid Computing (GRID)*, 2012, pp. 164–173. doi:10.1109/Grid.2012.15

[30] Z. Li, L. O’Brien, and H. Zhang, “CEEM: A practical methodology for cloud services evaluation,” in *Proceedings of the 9th IEEE World Congress on Services (SERVICES)*, 2013, pp. 44–51. doi:10.1109/SERVICES.2013.73

[31] A. Iosup, R. Prodan, and D. H. J. Epema, “IaaS cloud benchmarking: Approaches, challenges, and experience,” in *Cloud Computing for Data-Intensive Applications*, 2014, pp. 83–104. doi:10.1007/978-1-4939-1905-5_4

[32] D. J. Lilja, *Measuring computer performance: a practitioner’s guide*. Cambridge university press, 2005.

[33] S. Kounev, K. Lange, and J. von Kistowski, *Systems Benchmarking - For Scientists and Engineers.* Springer, 2020. doi:10.1007/978-3-030-41705-5

[34] N. Gibbs, “Microbenchmarks vs macrobenchmarks (i.e. what’s a microbenchmark?),” 2019. [Online]. Available: https://engineering.appfolio.com/appfolio-engineering/2019/1/7/microbenchmarks-vs-macrobenchmarks-ie-whats-a-microbenchmark

[35] R. R. Sambasivan, R. Fonseca, I. Shafer, and G. R. Ganger, “So, you want to trace your distributed system? key design insights from years of practical experience,” Parallel Data Laboratory, Carnegie Mellon University, Tech. Rep., 2014. [Online]. Available: https://www.pdl.cmu.edu/PDL-FTP/SelfStar/CMU-PDL-14-102.pdf

[36] R. Fonseca, G. Porter, R. H. Katz, S. Shenker, and I. Stoica, “X-trace: A pervasive network tracing framework,” in *4th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007. [Online]. Available: http://www.usenix.org/events/nsdi07/tech/fonseca.html

[37] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag, “Dapper, a large-scale distributed systems tracing infrastructure,” Google, Tech. Rep., 2010. [Online]. Available: https://research.google/pubs/pub36356/

[38] A. Parker, D. Spoonhower, J. Mace, B. Sigelman, and R. Isaacs, *Distributed tracing in practice: Instrumenting, analyzing, and debugging microservices.* O’Reilly Media, 2020.

[39] R. Schwarz and F. Mattern, “Detecting causal relationships in distributed computations: In search of the holy grail,” *Distributed Computation*, vol. 7, pp. 149–174, 1994. doi:10.1007/BF02277859

[40] P. Barham, R. Isaacs, R. Mortier, and D. Narayanan, “Magpie: Online modelling and performance-aware systems,” in *Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS)*, 2003, pp. 85–90. [Online]. Available: https://www.usenix.org/conference/hotos-ix/magpie-online-modelling-and-performance-aware-systems

[41] J. Zhou, “Distributed tracing, a survey of past and future,” Spectro Cloud, 2020. [Online]. Available: https://www.spectrocloud.com/blog/distributed-tracing-past-and-future/

[42] J. Mace, “End-to-end tracing: Adoption and use cases,” Brown University, Tech. Rep., 2017. [Online]. Available: https://cs.brown.edu/~jcmace/papers/mace2017survey.pdf

[43] C. S. Collberg and T. A. Proebsting, “Repeatability in computer systems research,” *Communications of the ACM*, vol. 59, pp. 62–69, 2016. doi:10.1145/2812803

[44] M. Baker, “Reproducibility crisis,” *Nature*, vol. 533, pp. 353–66, 2016. [Online]. Available: https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970

[45] B. N. Taylor and C. E. Kuyatt, “Guidelines for evaluating and expressing the uncertainty of NIST measurement results,” National Institute

of Standards and Technology, Tech. Rep., 1994. [Online]. Available: https://emtoolbox.nist.gov/Publications/NISTTechnicalNote1297s.pdf

[46] A. V. Papadopoulos, L. Versluis, A. Bauer, N. Herbst, J. von Kistowski, A. Ali-Eldin, C. L. Abad, J. N. Amaral, P. Tuma, and A. Iosup, "Methodological principles for reproducible performance evaluation in cloud computing," *IEEE Transactions on Software Engineering (TSE)*, vol. 47, pp. 93–94, 2019. doi:10.1109/TSE.2019.2927908

[47] D. G. Feitelson, "Experimental computer science: The need for a cultural change," The Hebrew University of Jerusalem, Tech. Rep., 2006.

[48] J. Lin and Q. Zhang, "Reproducibility is a process, not an achievement: The replicability of IR reproducibility experiments," in *Proceedings of the 42nd European Conference on Advances in Information Retrieval IR*, vol. 12036, 2020, pp. 43–49. doi:10.1007/978-3-030-45442-5_6

[49] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Serverless Computation with OpenLambda," in *Proceedings of the 8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, vol. 60, 2016. [Online]. Available: https://www.usenix.org/conference/hotcloud16/workshop-program/presentation/hendrickson

[50] G. McGrath, J. Short, S. Ennis, B. Judson, and P. Brenner, "Cloud event programming paradigms: Applications and analysis," in *Proceedings of the 9th IEEE International Conference on Cloud Computing (CLOUD)*, 2016, pp. 400–06. doi:10.1109/CLOUD.2016.0060

[51] V. Yussupov, U. Breitenbücher, F. Leymann, and M. Wurster, "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 (UCC)*, 2019, pp. 229–240. doi:10.1145/3344341.3368803

[52] J. Wen, Z. Chen, and X. Liu, "A literature review on serverless computing," *CoRR*, vol. abs/2206.12275, 2022. doi:10.48550/arXiv.2206.12275

[53] J. Kuhlenkamp and S. Werner, "Benchmarking FaaS platforms: Call for community participation," in *Companion of the 11th IEEE/ACM UCC: 4th International Workshop on Serverless Computing (WoSC)*, 2018, pp. 189–194. doi:10.1109/UCC-Companion.2018.00055

[54] N. Somu, N. Daw, U. Bellur, and P. Kulkarni, "PanOpticon: A comprehensive benchmarking tool for serverless applications," in *Proceedings of the International Conference on COMmunication Systems NETworkS (COMSNETS)*, 2020, pp. 144–51. doi:10.1109/COMSNETS48256.2020.9027346

[55] P. Swail, "Case studies of aws serverless apps in production," ServerlessFirst, 2018. [Online]. Available: https://serverlessfirst.com/real-world-serverless-case-studies/

[56] P. Leitner, E. Wittern, J. Spillner, and W. Hummer, "A mixed-method empirical study of function-as-a-service software development in industrial practice," *Journal of Systems and Software (JSS)*, vol. 149, pp. 340–359, 2019. doi:10.1016/j.jss.2018.12.013

[57] D. Taibi, N. El Ioini, C. Pahl, and J. R. S. Niederkofler, "Patterns for serverless functions (function-as-a-service): A multivocal literature review," *Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER)*, 2020. doi:10.5220/0009578501810192

[58] T. Yu, Q. Liu, D. Du, Y. Xia, B. Zang, Z. Lu, P. Yang, C. Qin, and H. Chen, "Characterizing serverless platforms with serverlessbench," in *Proceedings of the ACM Symposium on Cloud Computing (SoCC)*, 2020, pp. 30–44. doi:10.1145/3419111.3421280

[59] D. Ustiugov, P. Petrov, M. Kogias, E. Bugnion, and B. Grot, "Benchmarking, analysis, and optimization of serverless function snapshots," in *26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2021, pp. 559–572. doi:10.1145/3445814.3446714

[60] M. Shahrad, J. Balkind, and D. Wentzlaff, "Architectural implications of function-as-a-service computing," in *Proceedings of the 52nd IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2019, pp. 1063–1075. doi:10.1145/3352460.3358296

[61] J. Kim and K. Lee, "FunctionBench: A suite of workloads for serverless cloud function service," in *Proceedings of the 12th IEEE International Conference on Cloud Computing (CLOUD WIP)*, 2019, pp. 502–504. doi:10.1109/CLOUD.2019.00091

[62] M. Copik, G. Kwasniewski, M. Besta, M. Podstawski, and T. Hoefler, "SeBS: a serverless benchmark suite for function-as-a-service computing," in *Proceedings of the 22nd International Middleware Conference*, 2021, pp. 64–78. doi:10.1145/3464298.3476133

[63] M. Grambow, T. Pfandzelter, L. Burchard, C. Schubert, M. Zhao, and D. Bermbach, "Befaas: An application-centric benchmarking framework for faas platforms," in *IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 1–8. doi:10.1109/IC2E52221.2021.00014

[64] M. Waseem, P. Liang, M. Shahin, A. D. Salle, and G. Márquez, "Design, monitoring, and testing of microservices systems: The practitioners' perspective," *Journal of Systems and Software (JSS)*, vol. 182, 2021. doi:10.1016/j.jss.2021.111061

[65] B. Li, X. Peng, Q. Xiang, H. Wang, T. Xie, J. Sun, and X. Liu, "Enjoy your observability: an industrial survey of microservice tracing and analysis," *Empirical Software Engineering (EMSE)*, vol. 27, p. 25, 2021. doi:10.1007/s10664-021-10063-9

[66] A. Bento, J. Correia, R. Filipe, F. Araújo, and J. S. Cardoso, "Automated analysis of distributed tracing: Challenges and research directions," *J. Grid Comput.*, vol. 19, p. 9, 2021. doi:10.1007/s10723-021-09551-5

[67] J. Kaldor, J. Mace, M. Bejda, E. Gao, W. Kuropatwa, J. O'Neill, K. W. Ong, B. Schaller, P. Shan, B. Viscomi, V. Venkataraman, K. Veeraraghavan, and Y. J. Song, "Canopy: An end-to-end performance tracing and analysis system," in *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*, 2017, pp. 34–50. doi:10.1145/3132747.3132749

[68] J. Mace, R. Roelke, and R. Fonseca, "Pivot tracing: dynamic causal monitoring for distributed systems," *Communications of the ACM*, vol. 63, pp. 94–102, 2020. doi:10.1145/3378933

[69] M. Hendriks, J. Verriet, T. Basten, B. D. Theelen, M. Brassé, and L. J. Somers, "Analyzing execution traces: critical-path analysis and distance analysis," *Int. J. Softw. Tools Technol. Transf.*, vol. 19, pp. 487–510, 2017. doi:10.1007/s10009-016-0436-z

[70] M. Hendriks and F. W. Vaandrager, "Reconstructing critical paths from execution traces," in *15th IEEE International Conference on Computational Science and Engineering (CSE)*, 2012, pp. 524–531. doi:10.1109/ICCSE.2012.78

[71] H. Qiu, S. S. Banerjee, S. Jha, Z. T. Kalbarczyk, and R. K. Iyer, "FIRM: An intelligent fine-grained resource management framework for slo-oriented microservices," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2020, pp. 805–825. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/qiu

[72] S. Luo, H. Xu, C. Lu, K. Ye, G. Xu, L. Zhang, Y. Ding, J. He, and C. Xu, "Characterizing microservice dependency and performance: Alibaba trace analysis," in *ACM Symposium on Cloud Computing (SoCC)*, 2021, pp. 412–426. doi:10.1145/3472883.3487003

[73] W. Lin, C. Krintz, R. Wolski, M. Zhang, X. Cai, T. Li, and W. Xu, "Tracking causal order in AWS lambda applications," in *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*, 2018, pp. 50–60. doi:10.1109/IC2E.2018.00027

[74] W. Lin, C. Krintz, and R. Wolski, "Tracing function dependencies across clouds," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018, pp. 253–60. doi:10.1109/CLOUD.2018.00039

[75] M. C. Borges, S. Werner, and A. Kilic, "FaaSter troubleshooting - evaluating distributed tracing approaches for serverless applications," in *IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 83–90. doi:10.1109/IC2E52221.2021.00022

[76] J. Kuhlenkamp and M. Klems, "Costradamus: A cost-tracing system for cloud-based software services," in *Proceedings of the 15th International Conference on Service-Oriented Computing (ICSOC)*, vol. 10601, 2017, pp. 657–672. doi:10.1007/978-3-319-69035-3_48

[77] V. Lenarduzzi and A. Panichella, "Serverless testing: Tool vendors' and experts' points of view," *IEEE Software*, vol. 38, pp. 54–60, 2021. doi:10.1109/MS.2020.3030803

[78] S. L. Garfinkel, "An evaluation of Amazon's grid computing services: EC2, S3, and SQS," Harvard University Cambridge, Tech. Rep., 2007. [Online]. Available: https://dash.harvard.edu/handle/1/24829568

[79] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, A. Fox, and D. Patterson, "Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0," in *1st Workshop on Cloud Computing and Its Applications (CCA)*, 2008. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.163.7403&rep=rep1&type=pdf

[80] E. Walker, "Benchmarking amazon EC2 for high-performance scientific computing," *Usenix Login*, vol. 33, pp. 18–23, 2008. [Online]. Available: https://www.usenix.org/system/files/login/articles/277-walker.pdf

[81] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing, "How is the weather tomorrow?: Towards a benchmark for the cloud," in *Proceedings of the 2nd International Workshop on Testing Database Systems (DBTest)*, 2009, pp. 9:1–9:6. doi:10.1145/1594156.1594168

[82] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun, "Benchmarking in the cloud: What it should, can, and cannot be," in *Proceedings of the 4th TPC Technology Conference on Selected Topics in Performance Evaluation and Benchmarking (TPCTC)*, vol. 7755, 2012, pp. 173–188. doi:10.1007/978-3-642-36727-4_12

[83] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: A study of emerging scale-out workloads on modern hardware," *Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, vol. 40, pp. 37–48, 2012. doi:10.1145/2189750.2150982

[84] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. J. Epema, "A performance analysis of EC2 cloud computing services for scientific computing," in *Cloud Computing*, vol. 34, 2009, pp. 115–131. doi:10.1007/978-3-642-12636-9_9

[85] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 931–945, 2011. doi:10.1109/TPDS.2011.66

[86] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC)*, 2010, pp. 143–154. doi:10.1145/1807128.1807152

[87] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: Observing, analyzing, and reducing variance," *Proceedings of the VLDB Endowment*, vol. 3, pp. 460–471, 2010. doi:10.14778/1920841.1920902

[88] Z. Ou, H. Zhuang, J. K. Nurminen, A. Ylä-Jääski, and P. Hui, "Exploiting hardware heterogeneity within the same instance type of amazon EC2," in *Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Computing (HotCloud)*, 2012. [Online]. Available: https://www.usenix.org/conference/hotcloud12/workshop-program/presentation/ou

[89] Z. Ou, H. Zhuang, A. Lukyanenko, J. K. Nurminen, P. Hui, V. Mazalov, and A. Ylä-Jääski, "Is the same instance type created equal? exploiting heterogeneity of public clouds," *IEEE Transactions on Cloud Computing*, vol. 1, pp. 201–14, 2013. doi:10.1109/TCC.2013.12

[90] Z. Li, H. Zhang, L. O'Brien, R. Cai, and S. Flint, "On evaluating commercial cloud services: A systematic review," *Journal of Systems and Software*, vol. 86, pp. 2371–2393, 2013. doi:10.1016/j.jss.2013.04.021

[91] P. Leitner and J. Cito, "Patterns in the chaos – a study of performance variation and predictability in public IaaS clouds," *ACM Transactions on Internet Technology*, vol. 16, pp. 1–23, 2016. doi:10.1145/2885497

[92] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2011, pp. 104–13. doi:10.1109/CCGrid.2011.22

[93] A. Abedi and T. Brecht, "Conducting repeatable experiments in highly variable cloud computing environments," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering (ICPE)*, 2017, pp. 287–292. doi:10.1145/3030207.3030229

[94] M. Silva, M. Hines, D. Gallo, Q. Liu, K. D. Ryu, and D. Da Silva, "Cloudbench: Experiment automation for cloud environments," in *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*, 2013, pp. 302–11. doi:10.1109/IC2E.2013.33

[95] M. Cunha, N. das Chagas Mendonça, and A. Sampaio, "Cloud Crawler: a declarative performance evaluation environment for infrastructure-as-a-service clouds," *Concurrency and Computation: Practice and Experience*, vol. 29, 2017. doi:10.1002/cpe.3825

[96] D. Jayasinghe, G. Swint, S. Malkowski, J. Li, Q. Wang, J. Park, and C. Pu, "Expertus: A generator approach to automate performance testing in IaaS clouds," in *Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD)*, 2012, pp. 115–22. doi:10.1109/CLOUD.2012.98

[97] J. Scheuner, P. Leitner, J. Cito, and H. Gall, "Cloud WorkBench – infrastructure-as-code based cloud benchmarking," in *Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2014, pp. 246–253. doi:10.1109/CloudCom.2014.98

[98] A. Li, X. Zong, S. Kandula, X. Yang, and M. Zhang, "CloudProphet: Towards application performance prediction in cloud," in *Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM)*, 2011, pp. 426–427. doi:10.1145/2018436.2018502

[99] A. Evangelinou, M. Ciavotta, D. Ardagna, A. Kopaneli, G. Kousiouris, and T. Varvarigou, “Enterprise applications cloud rightsizing through a joint benchmarking and optimization approach,” *Future Generation Computer Systems*, pp. 102–114, 2016. doi:10.1016/j.future.2016.11.002

[100] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, “Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics,” in *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017, pp. 469–482. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/alipourfard

[101] B. Varghese, O. Akgun, I. Miguel, L. Thai, and A. Barker, “Cloud benchmarking for maximising performance of scientific applications,” *IEEE Transactions on Cloud Computing (TCC)*, vol. 7, pp. 170–182, 2019. doi:10.1109/TCC.2016.2603476

[102] N. J. Yadwadkar, B. Hariharan, J. E. Gonzalez, B. Smith, and R. H. Katz, “Selecting the best VM across multiple public clouds: a data-driven performance modeling approach,” in *Proceedings of the Symposium on Cloud Computing (SoCC)*, 2017, pp. 452–465. doi:10.1145/3127479.3131614

[103] W. Wang, N. Tian, S. Huang, S. He, A. Srivastava, M. L. Soffa, and L. Pollock, “Testing cloud applications under cloud-uncertainty performance effects,” in *11th IEEE Conference on Software Testing, Validation and Verification (ICST)*, 2018. doi:10.1109/ICST.2018.00018

[104] M. Baughman, R. Chard, L. T. Ward, J. Pitt, K. Chard, and I. T. Foster, “Profiling and predicting application performance on the cloud,” in *Proceedings of the 11th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2018, pp. 21–30. doi:10.1109/UCC.2018.00011

[105] Y. El-Khamra, H. Kim, S. Jha, and M. Parashar, “Exploring the performance fluctuations of HPC workloads on clouds,” in *Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2010, pp. 383–87. doi:10.1109/CloudCom.2010.84

[106] A. Uta, A. Custura, D. Duplyakin, I. Jimenez, J. S. Rellermeyer, C. Maltzahn, R. Ricci, and A. Iosup, “Is big data performance reproducible in modern cloud networks?” in *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020, pp. 513–527. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/uta

[107] J. Eickhoff, J. Donkervliet, and A. Iosup, “Meterstick: Benchmarking performance variability in cloud and self-hosted minecraft-like games extended technical report,” *CoRR*, vol. abs/2112.06963, 2021. [Online]. Available: https://arxiv.org/abs/2112.06963

[108] L. Kotthoff, “Reliability of computational experiments on virtualised hardware,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 26, pp. 33–9, 2014. doi:10.1080/0952813X.2013.784812

[109] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu, "Understanding performance interference of I/O workload in virtualized cloud environments," in *IEEE International Conference on Cloud Computing (CLOUD)*, 2010, pp. 51–58. doi:10.1109/CLOUD.2010.65

[110] V. Horký, P. Libic, L. Marek, A. Steinhauser, and P. Tuma, "Utilizing performance unit tests to increase performance awareness," in *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering (ICPE)*, 2015, pp. 289–300. doi:10.1145/2668930.2688051

[111] P. Stefan, V. Horký, L. Bulej, and P. Tuma, "Unit testing performance in java projects: Are we there yet?" in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering (ICPE)*, 2017, pp. 401–412. doi:10.1145/3030207.3030226

[112] P. Leitner and C. Bezemer, "An exploratory study of the state of practice of performance testing in java-based open source projects," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, ICPE 2017, L'Aquila, Italy, April 22-26, 2017*, 2017, pp. 373–384. doi:10.1145/3030207.3030213

[113] D. Costa, C. Bezemer, P. Leitner, and A. Andrzejak, "What's wrong with my benchmark results? studying bad practices in JMH benchmarks," *IEEE Trans. Software Eng.*, vol. 47, pp. 1452–1467, 2021. doi:10.1109/TSE.2019.2925345

[114] J. Chen and W. Shang, "An exploratory study of performance regression introducing code changes," in *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2017, pp. 341–352. doi:10.1109/ICSME.2017.13

[115] K.-J. Stol and B. Fitzgerald, "The ABC of software engineering research," *ACM Transactions on Software Engineering and Methodology*, vol. 27, pp. 1–51, 2018. doi:10.1145/3241743

[116] P. Ralph, S. Baltes, D. Bianculli, Y. Dittrich, M. Felderer, R. Feldt, A. Filieri, C. A. Furia, D. Graziotin, P. He, R. Hoda, N. Juristo, B. Kitchenham, R. Robbes, D. Mendez, J. Molleri, D. Spinellis, M. Staron, K. Stol, D. Tamburri, M. Torchiano, C. Treude, B. Turhan, and S. Vegas, "Acm sigsoft empirical standards," 2020. [Online]. Available: https://github.com/acmsigsoft/EmpiricalStandards

[117] R. J. Wieringa, "Design science as nested problem solving," in *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology (DESRIST)*, 2009. doi:10.1145/1555619.1555630

[118] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University, Tech. Rep., 2007. [Online]. Available: http://cdn.elsevier.com/promis_misc/525444systematicreviewsguide.pdf

[119] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Information and Software Technology*, vol. 106, pp. 101–121, 2019. doi:10.1016/j.infsof.2018.09.006

[120] H. Zhang, M. A. Babar, and P. Tell, "Identifying relevant studies in software engineering," *Information and Software Technology*, vol. 53, pp. 625–637, 2011. doi:10.1016/j.infsof.2010.12.010

[121] Y. Zacchia Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. a. Di Benedetto, "State of the art of cyber-physical systems security: An automatic control perspective," *Journal of Systems and Software (JSS)*, vol. 149, pp. 174–216, 2019. doi:10.1016/j.jss.2018.12.006

[122] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977. doi:10.2307/2529310

[123] S. Easterbrook, J. Singer, M. D. Storey, and D. E. Damian, "Selecting empirical methods for software engineering research," in *Guide to Advanced Empirical Software Engineering*, 2008, pp. 285–311. doi:10.1007/978-1-84800-044-5_11

[124] S. Baltes and P. Ralph, "Sampling in software engineering research: a critical review and guidelines," *Empir. Softw. Eng.*, vol. 27, p. 94, 2022. doi:10.1007/s10664-021-10072-8

[125] G. Gousios, "The GHTorent dataset and tool suite," in *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR)*, 2013, pp. 233–236. doi:10.1109/MSR.2013.6624034

[126] J. Spillner and M. Al-Ameen, "Serverless literature dataset," 2019. doi:10.5281/zenodo.2649001

[127] W. Hasselbring, "Benchmarking as empirical standard in software engineering research," in *Evaluation and Assessment in Software Engineering (EASE)*, 2021, pp. 365–372. doi:10.1145/3463274.3463361

[128] J. von Kistowski, J. A. Arnold, K. Huppler, K. Lange, J. L. Henning, and P. Cao, "How to build a benchmark," in *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering (ICPC)*, 2015, pp. 333–336. doi:10.1145/2668930.2688819

[129] K. Huppler, "The art of building a good benchmark," in *Performance Evaluation and Benchmarking, 1st TPC Technology Conference*, vol. 5895, 2009, pp. 18–30. doi:10.1007/978-3-642-10424-4_3

[130] B. Gregg, *Systems Performance: Enterprise and the Cloud.* Prentice Hall, 2013. [Online]. Available: http://books.google.ch/books?id=pTYkAQAAQBAJ

[131] J. Manner, M. Endreß, T. Heckel, and G. Wirtz, "Cold start influencing factors in function as a service," in *Companion of the 11th IEEE/ACM UCC: 4th International Workshop on Serverless Computing (WoSC)*, 2018, pp. 181–88. doi:10.1109/UCC-Companion.2018.00054

[132] K. Figiela, A. Gajek, A. Zima, B. Obrok, and M. Malawski, "Performance evaluation of heterogeneous cloud functions," *Concurrency and Computation: Practice and Experience*, vol. 30, 2018. doi:10.1002/cpe.4792

[133] I. Pelle, J. Czentye, J. Dóka, and B. Sonkoly, "Towards latency sensitive cloud native applications: A performance study on AWS," in *Proceedings of the 12th IEEE International Conference on Cloud Computing (CLOUD)*, 2019, pp. 272–280. doi:10.1109/CLOUD.2019.00054

[134] Markets and Markets, "Function-as-a-service market," 2017. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/function-as-a-service-market-127202409.html

[135] R. Magoulas and C. Guzikowski, "O'reilly serverless survey 2019: Concerns, what works, and what to expect," O'Reilly, 2019. [Online]. Available: https://www.oreilly.com/radar/oreilly-serverless-survey-2019-concerns-what-works-and-what-to-expect/

[136] S. Fouladi, R. S. Wahby, B. Shacklett, K. Balasubramaniam, W. Zeng, R. Bhalerao, A. Sivaraman, G. Porter, and K. Winstein, "Encoding, fast and slow: Low-latency video processing using thousands of tiny threads," in *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017, pp. 363–376. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/fouladi

[137] S. Fouladi, F. Romero, D. Iter, Q. Li, S. Chatterjee, C. Kozyrakis, M. Zaharia, and K. Winstein, "From laptop to lambda: Outsourcing everyday jobs to thousands of transient functional containers," in *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2019, pp. 475–488. [Online]. Available: https://www.usenix.org/conference/atc19/presentation/fouladi

[138] S. Bebortta, S. K. Das, M. Kandpal, R. K. Barik, and H. Dubey, "Geospatial serverless computing: Architectures, tools and future directions," *International Journal of Geo-Information (ISPRS)*, vol. 9, p. 311, 2020. doi:10.3390/ijgi9050311

[139] S. Brisals, "Accelerating with serverless!" LEGO at Medium, 2020. [Online]. Available: https://medium.com/lego-engineering/accelerating-with-serverless-625da076964b

[140] T. Rehemägi, "Companies using serverless in production," Dashbird, 2018. [Online]. Available: https://dashbird.io/blog/companies-using-serverless-in-production/

[141] G. Adzic and R. Chatley, "Serverless computing: Economic and architectural impact," in *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, 2017, pp. 884–889. doi:10.1145/3106237.3117767

[142] V. Yussupov, U. Breitenbücher, F. Leymann, and C. Müller, "Facing the unplanned migration of serverless applications: A study on portability problems, solutions, and dead ends," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2019, pp. 273–283. doi:10.1145/3344341.3368813

[143] M. Shahrad, R. Fonseca, I. Goiri, G. Chaudhry, P. Batum, J. Cooke, E. Laureano, C. Tresness, M. Russinovich, and R. Bianchini, "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider," in *USENIX Annual Technical Conference (ATC)*, 2020, pp. 205–218. [Online]. Available: https://www.usenix.org/conference/atc20/presentation/shahrad

[144] V. Vedam and J. Vemulapati, "Demystifying cloud benchmarking paradigm – an in depth view," in *Proceedings of the 36th IEEE Computer Software and Applications Conference (COMPSAC)*, 2012, pp. 416–21. doi:10.1109/COMPSAC.2012.61

[145] S. Eismann, L. Bui, J. Grohmann, C. Abad, N. Herbst, and S. Kounev, "Sizeless: Predicting the optimal size of serverless functions," in *Proceedings of the 22nd International Middleware Conference*, 2021, pp. 248–259. doi:10.1145/3464298.3493398

[146] R. Cordingly, W. Shu, and W. J. Lloyd, "Predicting performance and cost of serverless computing functions with SAAF," in *IEEE International Conference on Cloud and Big Data (CBDCom)*, 2020, pp. 640–649. doi:10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00111

[147] R. Cordingly, N. Heydari, H. Yu, V. Hoang, Z. Sadeghi, and W. Lloyd, "Enhancing observability of serverless computing with the serverless application analytics framework," in *Companion of the ACM/SPEC International Conference on Performance Engineering (ICPE Tutorial)*, 2021, pp. 161–164. doi:10.1145/3447545.3451173

[148] Anibal et al., "Awesome serverless: A curated list of awesome services, solutions and resources for serverless / nobackend applications." 2022. [Online]. Available: https://github.com/anaibol/awesome-serverless

[149] OpenTelemetry, "Opentelemetry specification (experimental): Messaging systems," 2022. [Online]. Available: https://github.com/open-telemetry/opentelemetry-specification/blob/70fecd2dcba505b3ac3a7cb1851f947047743d24/specification/trace/semantic_conventions/messaging.md

[150] J. Nielsen, *Usability engineering.* Morgan Kaufmann, 1994. [Online]. Available: https://www.nngroup.com/books/usability-engineering/

[151] R. B. Miller, "Response time in man-computer conversational transactions," in *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, 1968, pp. 267–277. doi:10.1145/1476589.1476628

[152] J. Nielsen, "Powers of 10: Time scales in user experience," Nielsen Norman Group, 2009. [Online]. Available: https://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/

[153] ——, “Response times: The 3 important limits,” Nielsen Norman Group, 1993. [Online]. Available: https://www.nngroup.com/articles/response-times-3-important-limits/

[154] Google, “Measure performance with the rail model,” 2015. [Online]. Available: https://www.smashingmagazine.com/2015/10/rail-user-centric-model-performance/

[155] J. Ishmael, “Optimising serverless for bbc online,” BBC at Medium, 2021. [Online]. Available: https://medium.com/bbc-design-engineering/optimising-serverless-for-bbc-online-118fe2c04beb

[156] S. Quinn, R. Cordingly, and W. Lloyd, “Implications of alternative serverless application control flow methods,” in *Proceedings of the Seventh International Workshop on Serverless Computing (WoSC)*, 2021, pp. 17–22. doi:10.1145/3493651.3493668

[157] C. Laaber, S. Würsten, H. C. Gall, and P. Leitner, “Dynamically reconfiguring software microbenchmarks: reducing execution time without sacrificing result quality,” in *28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2020, pp. 989–1001. doi:10.1145/3368089.3409683

[158] J. Scheuner and P. Leitner, “Replication package for "function-as-a-service performance evaluation: a multivocal literature review",” v1.0, 2020, dataset. doi:10.5281/zenodo.3906613

[159] S. Eismann, J. Scheuner, E. van Eyk, M. Schwinger, J. Grohmann, N. Herbst, and C. Abad, “The state of serverless applications: Collection, characterization, and community consensus – replication package,” 2021. doi:10.5281/zenodo.5185055

[160] Anonymous, “Let’s Trace It: Fine-Grained Serverless Benchmarking using Synchronous and Asynchronous Orchestrated Applications - Dataset,” Zenodo, 2022. doi:10.5281/zenodo.5879446

[161] J. Scheuner, R. Deng, J.-P. Steghöfer, and P. Leitner, “Serverless crossfit replication package,” Github, 2022, to be released on Zenodo. [Online]. Available: https://github.com/serverless-crossfit/replication-package

[162] J. Scheuner, M. Bertilsson, O. Grönqvist, H. Tao, H. Lagergren, J.-P. Steghöfer, and P. Leitner, “Replication package for "TriggerBench: A performance benchmark for serverless function triggers",” 2022. doi:10.5281/zenodo.6907484

[163] J. Scheuner and P. Leitner, “Replication package for "estimating cloud application performance based on micro-benchmark profiling",” Github, 2018. [Online]. Available: https://github.com/joe4dev/cwb-analysis

[164] C. Laaber, J. Scheuner, and P. Leitner, “Dataset, scripts, and online appendix "software microbenchmarking in the cloud. how bad is it really?",” 2019. doi:10.6084/m9.figshare.7546703.v1

[165] T. Kalibera and R. Jones, "Quantifying performance changes with effect size confidence intervals," University of Kent, Tech. Rep., 2012. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2012/3233

[166] ——, "Rigorous benchmarking in reasonable time," *SIGPLAN Not.*, vol. 48, pp. 63–74, 2013. doi:10.1145/2555670.2464160

[167] G. A. Rousselet, C. R. Pernet, and R. R. Wilcox, "Beyond differences in means: robust graphical methods to compare two groups in neuroscience," *European Journal of Neuroscience*, vol. 46, pp. 1738–1748, 2017. doi:https://doi.org/10.1111/ejn.13610

[168] J. Chen and Y. Liu, "Quantile and quantile-function estimations under density ratio model," *The Annals of Statistics*, vol. 41, pp. 1669 – 1692, 2013. doi:10.1214/13-AOS1129

[169] A. Akinshin, "Nonparametric cohen's d-consistent effect size," 2020. [Online]. Available: https://aakinshin.net/posts/nonparametric-effect-size/

[170] J. Sampé, P. G. López, M. S. Artigas, G. Vernik, P. Roca-Llaberia, and A. Arjona, "Toward multicloud access transparency in serverless computing," *IEEE Softw.*, vol. 38, pp. 68–74, 2021. doi:10.1109/MS.2020.3029994

[171] J. Sampe, M. Sanchez-Artigas, G. Vernik, I. Yehekzel, and P. Garcia-Lopez, "Outsourcing data processing jobs with lithops," *IEEE Transactions on Cloud Computing (TCC)*, pp. 1–1, 2021. doi:10.1109/TCC.2021.3129000

[172] M. Wurster, U. Breitenbücher, K. Képes, F. Leymann, , and V. Yussupov, "Modeling and automated deployment of serverless applications using TOSCA," in *Proceedings of the IEEE 11th International Conference on Service-Oriented Computing and Applications (SOCA)*, 2018, pp. 73–80. doi:10.1109/SOCA.2018.00017

[173] A. F. Baarzi, G. Kesidis, C. Joe-Wong, and M. Shahrad, "On merits and viability of multi-cloud serverless," in *ACM Symposium on Cloud Computing (SoCC)*, 2021, pp. 600–608. doi:10.1145/3472883.3487002

[174] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on Software Engineering (TSE)*, vol. 28, pp. 721–734, 2002.

[175] B. Gregg, "Benchmarking the cloud," 2014. [Online]. Available: https://www.brendangregg.com/blog/2014-01-10/benchmarking-the-cloud.html

[176] V. Garousi, M. Felderer, and T. Hacaloglu, "Software test maturity assessment and test process improvement: A multivocal literature review," *Information and Software Technology*, vol. 85, pp. 16–42, 2017. doi:10.1016/j.infsof.2017.01.001

[177] A. Najafi, A. Tai, and M. Wei, "Systems research is running out of time," in *Workshop on Hot Topics in Operating Systems (HotOS)*, 2021, pp. 65–71. doi:10.1145/3458336

[178] Q. Pu, S. Venkataraman, and I. Stoica, "Shuffling, fast and slow: Scalable analytics on serverless infrastructure," in *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2019, pp. 193–206. [Online]. Available: https://www.usenix.org/conference/nsdi19/presentation/pu

[179] I. Müller, R. Marroquín, and G. Alonso, "Lambada: Interactive data analytics on cold data using serverless cloud infrastructure," in *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2020, pp. 115–130. doi:10.1145/3318464.3389758

[180] A. Klimovic, Y. Wang, C. Kozyrakis, P. Stuedi, J. Pfefferle, and A. Trivedi, "Understanding ephemeral storage for serverless analytics," in *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2018, pp. 789–794. [Online]. Available: https://www.usenix.org/conference/atc18/presentation/klimovic-serverless

[181] A. Klimovic, Y. Wang, P. Stuedi, A. Trivedi, J. Pfefferle, and C. Kozyrakis, "Pocket: Elastic ephemeral storage for serverless analytics," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018, pp. 427–444. [Online]. Available: https://www.usenix.org/conference/osdi18/presentation/klimovic

[182] J. Jiang, S. Gan, Y. Liu, F. Wang, G. Alonso, A. Klimovic, A. Singla, W. Wu, and C. Zhang, "Towards demystifying serverless machine learning training," in *ACM International Conference on Management of Data (SIGMOD)*, 2021, pp. 857–871. doi:10.1145/3448016.3459240

[183] A. Ali, R. Pinciroli, F. Yan, and E. Smirni, "Batch: Machine learning inference serving on serverless platforms with adaptive batching," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2020, pp. 1–15. doi:10.1109/SC41405.2020.00073

[184] N. Akhtar, A. Raza, V. Ishakian, and I. Matta, "COSE: configuring serverless functions using statistical learning," in *39th IEEE Conference on Computer Communications (INFOCOM)*, 2020, pp. 129–138. doi:10.1109/INFOCOM41043.2020.9155363

[185] L. Wang, M. Li, Y. Zhang, T. Ristenpart, and M. Swift, "Peeking behind the curtains of serverless platforms," in *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2018, pp. 133–146. [Online]. Available: https://www.usenix.org/conference/atc18/presentation/wang-liang

[186] A. Casalboni, "AWS Lambda Power Tuning," 2019. [Online]. Available: https://github.com/alexcasalboni/aws-lambda-power-tuning

[187] F. C. Y. Benureau and N. P. Rougier, "Re-run, repeat, reproduce, reuse, replicate: Transforming code into scientific contributions," *Frontiers in Neuroinformatics*, vol. 11, p. 69, 2018. doi:10.3389/fninf.2017.00069

[188] P. Leitner and J. Scheuner, “Bursting with possibilities – an empirical study of credit-based bursting cloud instance types,” in *Proceedings of the 8th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2015, pp. 227–36. doi:10.1109/UCC.2015.39

[189] C. Davatz, C. Inzinger, J. Scheuner, and P. Leitner, “An approach and case study of cloud instance type selection for multi-tier web applications,” in *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2017, pp. 534–543. doi:10.1109/CCGRID.2017.12

[190] C. Boettiger, “An introduction to docker for reproducible research,” *Operating Systems Review*, vol. 49, pp. 71–79, 2015. doi:10.1145/2723872.2723882

[191] A. Raza, I. Matta, N. Akhtar, V. Kalavri, and V. Isahagian, “Sok: Function-as-a-service: From an application developer’s perspective,” *Journal of Systems Research (JSys)*, vol. 1, 2021. doi:10.5070/SR31154815

[192] H. B. Hassan, S. A. Barakat, and Q. I. Sarhan, “Survey on serverless computing,” *Journal of Cloud Computing*, vol. 10, p. 39, 2021. doi:10.1186/s13677-021-00253-7

[193] A. Wang, S. Chang, H. Tian, H. Wang, H. Yang, H. Li, R. Du, and Y. Cheng, “FaaSNet: Scalable and fast provisioning of custom serverless container runtimes at alibaba cloud function compute,” in *USENIX Annual Technical Conference (ATC)*, 2021, pp. 443–457. [Online]. Available: https://www.usenix.org/conference/atc21/presentation/wang-ao

[194] J. Wen and Y. Liu, “A measurement study on serverless workflow services,” in *IEEE International Conference on Web Services (ICWS)*, 2021, pp. 741–750. doi:10.1109/ICWS53863.2021.00102

[195] D. Xie, Y. Hu, and L. Qin, “An evaluation of serverless computing on X86 and ARM platforms: Performance and design implications,” in *IEEE 14th International Conference on Cloud Computing (CLOUD)*, 2021, pp. 313–321. doi:10.1109/CLOUD53861.2021.00045

[196] Y. Lin, K. Ye, Y. Li, P. Lin, Y. Tang, and C. Xu, “Bbserverless: A bursty traffic benchmark for serverless,” in *14th International Conference on Cloud Computing CLOUD 2021*, 2022, pp. 45–60. doi:10.1007/978-3-030-96326-2_4

[197] A. Agache, M. Brooker, A. Iordache, A. Liguori, R. Neugebauer, P. Piwonka, and D. Popa, “Firecracker: Lightweight virtualization for serverless applications,” in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020, pp. 419–434. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/agache

[198] K. Djemame, M. Parker, and D. Datsev, “Open-source serverless architectures: an evaluation of apache openwhisk,” in *13th IEEE/ACM*

International Conference on Utility and Cloud Computing (UCC), 2020, pp. 329–335. doi:10.1109/UCC48980.2020.00052

[199] A. Palade, A. Kazmi, and S. Clarke, "An evaluation of open source serverless computing frameworks support at the edge," in *IEEE World Congress on Services (SERVICES)*, vol. 2642-939X, 2019, pp. 206–11. doi:10.1109/SERVICES.2019.00057

[200] J. Scheuner and P. Leitner, "Transpiling applications into optimized serverless orchestrations," in *Proceedings of the 4th IEEE FAS*W: 2nd Workshop on Hot Topics in Cloud Computing Performance (HotCloudPerf) at ICAC/SASO*, 2019, pp. 72–73. doi:10.1109/FAS-W.2019.00031

[201] T. Schirmer, J. Scheuner, T. Pfandzelter, and D. Bermbach, "FUSIONIZE: Improving serverless application performance through feedback-driven function fusion," in *IEEE International Conference on Cloud Engineering (IC2E)*, 2022, to appear.

[202] A. Mahgoub, K. Shankar, S. Mitra, A. Klimovic, S. Chaterji, and S. Bagchi, "SONIC: Application-aware data passing for chained serverless applications," in *USENIX Annual Technical Conference (ATC)*, 2021, pp. 285–301. [Online]. Available: https://www.usenix.org/conference/atc21/presentation/mahgoub

[203] M. Zhang, Y. Zhu, J. Liu, F. Wang, and F. Wang, "Charm-Seeker: Automated pipeline configuration for serverless video processing," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2022. doi:10.1109/TNET.2022.3183231

[204] A. Mahgoub, E. B. Yi, K. Shankar, E. Minocha, S. Elnikety, S. Bagchi, and S. Chaterji, "WISEFUSE: workload characterization and DAG transformation for serverless workflows," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 6, pp. 26:1–26:28, 2022. doi:10.1145/3530892

[205] V. Ishakian, V. Muthusamy, and A. Slominski, "Serving deep learning models in a serverless platform," in *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*, 2018, pp. 257–62. doi:10.1109/IC2E.2018.00052

[206] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, and B. Regnell, *Experimentation in Software Engineering.* Springer, 2012. doi:10.1007/978-3-642-29044-2

[207] B. Farley, A. Juels, V. Varadarajan, T. Ristenpart, K. D. Bowers, and M. M. Swift, "More for your money: exploiting performance heterogeneity in public clouds," in *Proceedings of the 3rd ACM Symposium on Cloud Computing (SoCC)*, 2012. doi:10.1145/2391229.2391249

[208] J. Scheuner and P. Leitner, "Estimating cloud application performance based on micro-benchmark profiling," in *Proceedings of the 11th IEEE International Conference on Cloud Computing (CLOUD)*, 2018, pp. 90–97. doi:10.1109/CLOUD.2018.00019

[209] J. Kuhlenkamp, S. Werner, M. C. Borges, D. Ernst, and D. Wenzel, “Benchmarking elasticity of FaaS platforms as a foundation for objective-driven design of serverless applications,” in *Proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing (SAC)*, 2020, pp. 1576–1585. doi:10.1145/3341105.3373948

[210] P. J. Fleming and J. J. Wallace, “How not to lie with statistics: The correct way to summarize benchmark results,” *Communications of the ACM*, vol. 29, pp. 218–221, 1986. doi:10.1145/5666.5673

[211] E. van Eyk, J. Scheuner, S. Eismann, C. L. Abad, and A. Iosup, “Beyond microbenchmarks: The SPEC-RG vision for a comprehensive serverless benchmark,” in *Companion of the 11th ACM/SPEC ICPE: 3rd Workshop on Hot Topics in Cloud Computing Performance (HotCloudPerf)*, 2020, pp. 26–31. doi:10.1145/3375555.3384381

[212] C. Laaber, J. Scheuner, and P. Leitner, “Software microbenchmarking in the cloud. How bad is it really?” *Empirical Software Engineering (EMSE)*, vol. 24, pp. 2469–2508, 2019. doi:10.1007/s10664-019-09681-1

[213] M. Al-Ameen and J. Spillner, “Systematic and open exploration of FaaS and serverless computing research,” in *Proceedings of the European Symposium on Serverless Computing and Applications (ESSCA)*, vol. 2330, 2018, pp. 30–35. [Online]. Available: http://ceur-ws.org/Vol-2330/short2.pdf

[214] Z. Li, L. O’Brien, R. Cai, and H. Zhang, “Towards a taxonomy of performance evaluation of commercial cloud services,” in *Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD)*, 2012, pp. 344–51. doi:10.1109/CLOUD.2012.74

[215] N. Bjørndal, L. J. P. de Araújo, A. Bucchiarone, N. Dragoni, M. Mazzara, and S. Dustdar, “Benchmarks and performance metrics for assessing the migration to microservice-based architectures,” *J. Object Technol. (JOT)*, vol. 20, pp. 3:1–17, 2021. doi:10.5381/jot.2021.20.2.a3

[216] I. E. Akkus, R. Chen, I. Rimac, M. Stein, K. Satzke, A. Beck, P. Aditya, and V. Hilt, “SAND: towards high-performance serverless computing,” in *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2018, pp. 923–935. [Online]. Available: https://www.usenix.org/conference/atc18/presentation/akkus

[217] L. F. Albuquerque Jr, F. S. Ferraz, R. F. Oliveira, and S. M. Galdino, “Function-as-a-service x platform-as-a-service: Towards a comparative study on FaaS and PaaS,” in *12th International Conference on Software Engineering Advances (ICSEA)*, 2017, pp. 206–212. [Online]. Available: https://www.thinkmind.org/download.php?articleid=icsea_2017_9_30_10096

[218] T. Back and V. Andrikopoulos, “Using a microbenchmark to compare function as a service solutions,” in *Proceedings of the 7th European Service-Oriented and Cloud Computing (ESOCC)*, vol. 11116, 2018, pp. 146–160. doi:10.1007/978-3-319-99819-0_11

[219] D. Balla, M. Maliosz, C. Simon, and D. Gehberger, "Tuning runtimes in open source FaaS," in *Proceedings of the Internet of Vehicles. Technologies and Services Toward Smart Cities (IOV)*, vol. 11894, 2020, pp. 250–266. doi:10.1007/978-3-030-38651-1_21

[220] D. Bardsley, L. Ryan, and J. Howard, "Serverless performance and optimization strategies," in *Proceedings of the IEEE International Conference on Smart Cloud (SmartCloud)*, 2018, pp. 19–6. doi:10.1109/SmartCloud.2018.00012

[221] D. Bortolini and R. R. Obelheiro, "Investigating performance and cost in function-as-a-service platforms," in *Proceedings of the 14th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, vol. 96, 2019, pp. 174–185. doi:10.1007/978-3-030-33509-0_16

[222] J. Carreira, P. Fonseca, A. Tumanov, A. Zhang, and R. Katz, "A case for serverless machine learning," in *Workshop on Systems for ML and Open Source Software at NeurIPS*, 2018. [Online]. Available: http://learningsys.org/nips18/assets/papers/101CameraReadySubmissioncirrus_nips_final2.pdf

[223] Y. Gan, Y. Zhang, D. Cheng, A. Shetty, P. Rathi, N. Katarki, A. Bruno, J. Hu, B. Ritchken, B. Jackson, K. Hu, M. Pancholi, Y. He, B. Clancy, C. Colen, F. Wen, C. Leung, S. Wang, L. Zaruvinsky, M. Espinosa, R. Lin, Z. Liu, J. Padilla, and C. Delimitrou, "An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems," in *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019, pp. 3–18. doi:10.1145/3297858.3304013

[224] V. Giménez-Alventosa, G. Moltó, and M. Caballer, "A framework and a performance assessment for serverless mapreduce on AWS Lambda," *Future Generation Computer Systems*, vol. 97, pp. 259–274, 2019. doi:10.1016/j.future.2019.02.057

[225] V. Gupta, S. Wang, T. A. Courtade, and K. Ramchandran, "Oversketch: Approximate matrix multiplication for the cloud," in *Proceedings of the IEEE International Conference on Big Data (Big Data)*, 2018, pp. 298–304. doi:10.1109/BigData.2018.8622139

[226] A. Hall and U. Ramachandran, "An execution model for serverless functions at the edge," in *Proceedings of the International Conference on Internet of Things Design and Implementation (IoTDI)*, 2019, pp. 225–236. doi:10.1145/3302505.3310084

[227] C. Ivan, R. Vasile, and V. Dadarlat, "Serverless computing: An investigation of deployment environments for Web APIs," *Computers*, vol. 8, 2019. doi:10.3390/computers8020050

[228] D. Jackson and G. Clynch, "An investigation of the impact of language runtime on the performance and cost of serverless functions," in *Companion of the 11th IEEE/ACM UCC: 4th International Workshop*

on Serverless Computing (WoSC), 2018, pp. 154–60. doi:10.1109/UCC-Companion.2018.00050

[229] Q. Jiang, Y. C. Lee, and A. Y. Zomaya, "Serverless execution of scientific workflows," in *Proceedings of the 15th International Conference on Service-Oriented Computing (ICSOC)*, vol. 10601, 2017, pp. 706–721. doi:10.1007/978-3-319-69035-3_51

[230] J. Kim, J. Park, and K. Lee, "Network resource isolation in serverless cloud function service," in *Proceedings of the 7th International Workshop on Autonomic Management of High-Performance Grid and Cloud Computing (AMGCC) at ICAC/SASO*, 2019, pp. 182–187. doi:10.1109/FAS-W.2019.00051

[231] J. Kuhlenkamp, S. Werner, M. C. Borges, K. El Tal, and S. Tai, "An evaluation of FaaS platforms as a foundation for serverless big data processing," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2019, pp. 1–9. doi:10.1145/3344341.3368796

[232] H. Lee, K. Satyam, and G. C. Fox, "Evaluation of production serverless computing environments," in *Proceedings of the 11th IEEE CLOUD: 3rd International Workshop on Serverless Computing (WoSC)*, 2018, pp. 442–50. doi:10.1109/CLOUD.2018.00062

[233] J. Li, S. G. Kulkarni, K. K. Ramakrishnan, and D. Li, "Understanding open source serverless platforms: Design considerations and performance," in *Proceedings of the 5th International Workshop on Serverless Computing (WoSC) at Middleware*, 2019, pp. 37–42. doi:10.1145/3366623.3368139

[234] W. Lloyd, S. Ramesh, S. Chinthalapati, L. Ly, and S. Pallickara, "Serverless computing: An investigation of factors influencing microservice performance," in *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*, 2018, pp. 159–169. doi:10.1109/IC2E.2018.00039

[235] W. Lloyd, M. Vu, B. Zhang, O. David, and G. Leavesley, "Improving application migration to serverless computing platforms: Latency mitigation with keep-alive workloads," in *Companion of the 11th IEEE/ACM UCC: 4th International Workshop on Serverless Computing (WoSC)*, 2018, pp. 195–00. doi:10.1109/UCC-Companion.2018.00056

[236] P. G. López, M. Sánchez-Artigas, G. París, D. B. Pons, Á. R. Ollobarren, and D. A. Pinto, "Comparison of FaaS orchestration systems," in *Companion of the 11th IEEE/ACM UCC: 4th International Workshop on Serverless Computing (WoSC)*, 2018, pp. 148–53. doi:10.1109/UCC-Companion.2018.00049

[237] M. Malawski, A. Gajek, A. Zima, B. Balis, and K. Figiela, "Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions," *Future Generation Computer Systems*, 2017. doi:10.1016/j.future.2017.10.029

[238] S. Malla and K. Christensen, "HPC in the cloud: Performance comparison of function as a service (FaaS) vs infrastructure as a service (IaaS)," *Internet Technology Letters*, 2019. doi:10.1002/itl2.137

[239] G. McGrath and P. R. Brenner, “Serverless computing: Design, implementation, and performance,” in *Proceedings of the 37th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2017, pp. 405–10. doi:10.1109/ICDCSW.2017.36

[240] A. Mohan, H. Sane, K. Doshi, S. Edupuganti, N. Nayak, and V. Sukhomlinov, “Agile cold starts for scalable serverless,” in *Proceedings of the 11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2019. [Online]. Available: https://www.usenix.org/conference/hotcloud19/presentation/mohan

[241] S. K. Mohanty, G. Premsankar, and M. D. Francesco, “An evaluation of open source serverless computing frameworks,” in *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2018, pp. 115–120. doi:10.1109/CloudCom2018.2018.00033

[242] X. Niu, D. Kumanov, L. Hung, W. Lloyd, and K. Y. Yeung, “Leveraging serverless computing to improve performance for sequence comparison,” in *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB)*, 2019, pp. 683–687. doi:10.1145/3307339.3343465

[243] E. Oakes, L. Yang, D. Zhou, K. Houck, T. Harter, A. Arpaci-Dusseau, and R. Arpaci-Dusseau, “SOCK: Rapid task provisioning with serverless-optimized containers,” in *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2018, pp. 57–70. [Online]. Available: https://www.usenix.org/conference/atc18/presentation/oakes

[244] A. Pérez, G. Moltó, M. Caballer, and A. Calatrava, “A programming model and middleware for high throughput serverless computing applications,” in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC)*, 2019, pp. 106–113. doi:10.1145/3297280.3297292

[245] H. Puripunpinyo and M. H. Samadzadeh, “Effect of optimizing Java deployment artifacts on AWS Lambda,” in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017, pp. 438–43. doi:10.1109/INFCOMW.2017.8116416

[246] A. Saha and S. Jindal, “EMARS: efficient management and allocation of resources in serverless,” in *Proceedings of the 11th IEEE International Conference on Cloud Computing (CLOUD)*, 2018, pp. 827–830. doi:10.1109/CLOUD.2018.00113

[247] S. Shillaker, “A provider-friendly serverless framework for latency-critical applications,” in *12th Eurosys Doctoral Workshop*, 2018. [Online]. Available: http://conferences.inf.ed.ac.uk/EuroDW2018/papers/eurodw18-Shillaker.pdf

[248] A. Singhvi, S. Banerjee, Y. Harchol, A. Akella, M. Peek, and P. Rydin, “Granular computing and network intensive applications: Friends or foes?” in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, 2017, pp. 157–163. doi:10.1145/3152434.3152450

[249] J. Spillner, C. Mateos, and D. A. Monge, "Faaster, better, cheaper: The prospect of serverless scientific computing and HPC," in *Proceedings of the 4th Latin American Conference on High Performance Computing (CARLA)*, vol. 796, 2017, pp. 154–168. doi:10.1007/978-3-319-73353-1_11

[250] S. Werner, J. Kuhlenkamp, M. Klems, J. Müller, and S. Tai, "Serverless big data processing using matrix multiplication as example," in *Proceedings of the IEEE International Conference on Big Data (Big Data)*, 2018, pp. 358–65. doi:10.1109/BigData.2018.8622362

[251] M. Zhang, Y. Zhu, C. Zhang, and J. Liu, "Video processing with serverless computing: A measurement study," in *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2019, pp. 61–66. doi:10.1145/3304112.3325608

[252] IDC, "FutureScape: Worldwide it industry 2019 predictions," 2018. [Online]. Available: https://web.archive.org/web/20211029191036/https://www.idc.com/getdoc.jsp?containerId=US44403818

[253] Research and Markets, "$7.72 billion function-as-a-service market 2017," Businesswire, 2017. [Online]. Available: https://www.businesswire.com/news/home/20170227006262/en/7.72-Billion-Function-as-a-Service-Market-2017---Global

[254] E. van Eyk, L. Toader, S. Talluri, L. Versluis, A. Uta, and A. Iosup, "Serverless is more: From PaaS to present cloud computing," *IEEE Internet Computing*, vol. 22, pp. 8–17, 2018. doi:10.1109/MIC.2018.053681358

[255] E. Jonas, J. Schleier-Smith, V. Sreekanti, C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. J. Yadwadkar, J. E. Gonzalez, R. A. Popa, I. Stoica, and D. A. Patterson, "Cloud programming simplified: A berkeley view on serverless computing," *CoRR*, vol. abs/1902.03383, 2019. [Online]. Available: http://arxiv.org/abs/1902.03383

[256] E. V. Eyk, A. Iosup, C. L. Abad, J. Grohmann, and S. Eismann, "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 (ICPE)*, 2018, pp. 21–24. doi:10.1145/3185768.3186308

[257] J. M. Hellerstein, J. M. Faleiro, J. Gonzalez, J. Schleier-Smith, V. Sreekanti, A. Tumanov, and C. Wu, "Serverless computing: One step forward, two steps back," in *Proceedings of the 9th Conference on Innovative Data Systems Research (CIDR)*, 2019. [Online]. Available: http://cidrdb.org/cidr2019/papers/p119-hellerstein-cidr19.pdf

[258] E. Levinson, "Serverless community survey 2020," Nuweba, 2020. [Online]. Available: https://web.archive.org/web/20200627124546/https://nuweba.com/blog/serverless-community-survey-2020-results

[259] A. Eivy, "Be wary of the economics of "serverless" cloud computing," *IEEE Cloud Computing*, vol. 4, pp. 6–2, 2017. doi:10.1109/MCC.2017.32

[260] P. A. Witte, M. Louboutin, C. Jones, and F. J. Herrmann, "Serverless seismic imaging in the cloud," *CoRR*, vol. abs/1911.12447, 2019. [Online]. Available: http://arxiv.org/abs/1911.12447

[261] R. Crespo-Cepeda, G. Agapito, J. L. Vazquez-Poletti, and M. Cannataro, "Challenges and opportunities of amazon serverless lambda services in bioinformatics," in *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB)*, 2019, pp. 663–668. doi:10.1145/3307339.3343462

[262] M. Chan, "Containers vs. serverless: Which should you use, and when?" 2018. [Online]. Available: https://web.archive.org/web/20210116093357/https://www.thorntech.com/2018/08/containers-vs-serverless/

[263] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: A state-of-the-art review," *IEEE Transactions on Cloud Computing*, vol. 7, pp. 677–92, 2019. doi:10.1109/TCC.2017.2702586

[264] P. Maenhaut, B. Volckaert, V. Ongenae, and F. D. Turck, "Resource management in a containerized cloud: Status and challenges," *J. Netw. Syst. Manag.*, vol. 28, pp. 197–246, 2020. doi:10.1007/s10922-019-09504-0

[265] A. Orfin, "How droplr scales to millions with the serverless framework," Serverless.com, 2018. [Online]. Available: https://www.serverless.com/blog/how-droplr-scales-to-millions-serverless-framework

[266] E. van Eyk, A. Iosup, J. Grohmann, S. Eismann, A. Bauer, L. Versluis, L. Toader, N. Schmitt, N. Herbst, and C. L. Abad, "The SPEC-RG reference architecture for FaaS: From microservices and containers to serverless platforms," *IEEE Internet Computing*, vol. 23, pp. 7–18, 2019. doi:10.1109/MIC.2019.2952061

[267] I. Pavlov, S. Ali, and T. Mahmud, "Serverless development trends in open source: a mixed-research study," Bachelor's Thesis, 2019. [Online]. Available: https://gupea.ub.gu.se/handle/2077/62544

[268] S. Eismann, J. Scheuner, E. van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, "Serverless applications: Why, when, and how?" *IEEE Software*, vol. 38, pp. 32–39, 2021. doi:10.1109/MS.2020.3023302

[269] J. Walter, "Systematic data transformation to enable web coverage services (WCS) and ArcGIS image services within ESDIS cumulus cloud," 2019. [Online]. Available: https://earthdata.nasa.gov/esds/competitive-programs/access/arcgis-cloud

[270] J. Blomer, G. Ganis, S. Mosciatti, and R. Popescu, "Towards a serverless CernVM-FS," *EPJ Web Conf.*, vol. 214, p. 09007, 2019. doi:10.1051/epjconf/201921409007

[271] A. Bhattacharjee, A. D. Chhokra, Z. Kang, H. Sun, A. Gokhale, and G. Karsai, "BARISTA: efficient and scalable serverless serving system for deep learning prediction services," in *IEEE International Conference on Cloud Engineering (IC2E)*, 2019, pp. 23–33. doi:10.1109/IC2E.2019.00-10

[272] Z. Tu, M. Li, and J. Lin, "Pay-per-request deployment of neural network models using serverless architectures," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, 2018, pp. 6–10. doi:10.18653/v1/N18-5002

[273] A. Coffey and P. Atkinson, *Making sense of qualitative data: Complementary research strategies.* Sage Publications, Inc, 1996.

[274] G. Guest, K. M. MacQueen, and E. E. Namey, *Applied thematic analysis.* SAGE publications, 2011.

[275] K. L. Gwet, *Handbook of inter-rater reliability: The definitive guide to measuring the extent of agreement among raters.* Advanced Analytics, LLC, 2014.

[276] J. Scheuner and P. Leitner, "Function-as-a-service performance evaluation: A multivocal literature review," *Journal of Systems and Software (JSS)*, vol. 170, 2020. doi:10.1016/j.jss.2020.110708

[277] N. Malishev, "AWS Lambda cold start language comparisons, 2019 edition," LevelUp, 2019. [Online]. Available: https://levelup.gitconnected.com/aws-lambda-cold-start-language-comparisons-2019-edition-%EF%B8%8F-1946d32a0244

[278] S. Moellering and S. Grunwald, "Field notes: Optimize your Java application for AWS Lambda with Quarkus," 2020. [Online]. Available: https://aws.amazon.com/blogs/architecture/field-notes-optimize-your-java-application-for-aws-lambda-with-quarkus/

[279] I. Baldini, P. Cheng, S. J. Fink, N. Mitchell, V. Muthusamy, R. Rabbah, P. Suter, and O. Tardieu, "The serverless trilemma: Function composition for serverless computing," in *Proceedings of the ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!)*, 2017, pp. 89–103. doi:10.1145/3133850.3133855

[280] M. Laul, "Serverless case study - Netflix," dashbird, 2018. [Online]. Available: https://dashbird.io/blog/serverless-case-study-netflix/

[281] A. Williams, "Autodesk goes serverless in the AWS cloud, reduces account-creation time by 99%," Amazon Web Services, 2017. [Online]. Available: https://aws.amazon.com/solutions/case-studies/autodesk-serverless/

[282] S. E. Brockwell and I. R. Gordon, "A comparison of statistical methods for meta-analysis," *Statistics in medicine*, vol. 20, pp. 825–840, 2001.

[283] S. Makridakis, "Accuracy measures: theoretical and practical concerns," *International Journal of Forecasting*, vol. 9, pp. 527–529, 1993. doi:https://doi.org/10.1016/0169-2070(93)90079-3

[284] J. L. Myers, A. Well, and R. F. Lorch, *Research design and statistical analysis.* Routledge, 2010. doi:10.4324/9780203726631

[285] K.-I. Goh and A.-L. Barabási, “Burstiness and memory in complex systems,” *EPL (Europhysics Letters)*, vol. 81, p. 48002, 2008.

[286] A. Ali-Eldin, O. Seleznjev, S. S. Luna, J. Tordsson, and E. Elmroth, “Measuring cloud workload burstiness,” in *Proceedings of the 7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2014, pp. 566–572. doi:10.1109/UCC.2014.87

[287] J. Schleier-Smith, V. Sreekanti, A. Khandelwal, J. Carreira, N. J. Yadwadkar, R. A. Popa, J. E. Gonzalez, I. Stoica, and D. A. Patterson, “What serverless computing is and should become: The next phase of cloud computing,” *Communication of the ACM*, vol. 64, pp. 76–84, 2021. doi:10.1145/3406011

[288] D. Ustiugov, T. Amariucai, and B. Grot, “Analyzing tail latency in serverless clouds with stellar,” in *IEEE International Symposium on Workload Characterization (IISWC)*, 2021, pp. 51–62. [Online]. Available: http://www.iiswc.org/iiswc2021/index.html

[289] Wilkinson *et al.*, “The FAIR guiding principles for scientific data management and stewardship,” *Nature SciData*, vol. 3, 2016. doi:10.1038/sdata.2016.18

[290] A. Anwar, M. Mohamed, V. Tarasov, M. Littley, L. Rupprecht, Y. Cheng, N. Zhao, D. Skourtis, A. Warke, H. Ludwig, D. Hildebrand, and A. R. Butt, “Improving docker registry design based on production workload analysis,” in *16th USENIX Conference on File and Storage Technologies (FAST)*, 2018, pp. 265–278. [Online]. Available: https://www.usenix.org/conference/fast18/presentation/anwar

[291] S. Eismann, J. Scheuner, E. van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, “The state of serverless applications: Collection, characterization, and community consensus,” *IEEE Transactions on Software Engineering (TSE)*, 2021. doi:10.1109/TSE.2021.3113940

[292] Open Telemetry, “Open Telemetry: Sync and async children (follows_from),” Github. [Online]. Available: https://github.com/open-telemetry/opentelemetry-specification/issues/65

[293] A. Williams, “Guide to serverless technologies,” The New Stack, 2018. [Online]. Available: https://thenewstack.io/ebooks/serverless/guide-to-serverless-technologies/

[294] I. Cesar, “Amazon API Gateway to AWS Lambda,” Amazon, 2022. [Online]. Available: https://serverlessland.com/patterns/apigw-lambda-cdk

[295] Amazon, “Serverless image processing,” 2021. [Online]. Available: https://image-processing.serverlessworkshops.io

[296] A. Karandikar, “Realworld example app,” Github, 2022. [Online]. Available: https://github.com/anishkny/realworld-dynamodb-lambda

[297] J. McKim, "Serverless event sourcing at nordstrom," Nordstrom, 2017. [Online]. Available: https://web.archive.org/web/20210119044741/https://acloudguru.com/blog/engineering/serverless-event-sourcing-at-nordstrom-ea69bd8fb7cc

[298] M. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," *IEEE/ACM Trans. Netw.*, vol. 5, pp. 835–846, 1997. doi:10.1109/90.650143

[299] C. Avin, M. Ghobadi, C. Griner, and S. Schmid, "On the complexity of traffic traces and implications," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, pp. 20:1–20:29, 2020. doi:10.1145/3379486

[300] Amazon Web Services, "Using AWS Lambda with AWS X-Ray." [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/services-xray.html

[301] Microsoft Azure, "How to configure monitoring for Azure Functions," 2022. [Online]. Available: https://docs.microsoft.com/en-us/azure/azure-functions/configure-monitoring?tabs=v2#configure-scale-controller-logs

[302] R. Cordingly, H. Yu, V. Hoang, D. Perez, D. Foster, Z. Sadeghi, R. Hatchett, and W. J. Lloyd, "Implications of programming language selection for serverless data processing pipelines," in *IEEE International Conference on Cloud and Big Data (CBDCom)*, 2020, pp. 704–711. doi:10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00120

[303] D. Barcelona-Pons and P. García-López, "Benchmarking parallelism in FaaS platforms," *Future Generation Computer Systems (FGCS)*, vol. 124, pp. 268–284, 2021. doi:https://doi.org/10.1016/j.future.2021.06.005

[304] J. Wen, Y. Liu, Z. Chen, J. Chen, and Y. Ma, "Characterizing commodity serverless computing platforms," *Journal of Software: Evolution and Process*, 2021. doi:https://doi.org/10.1002/smr.2394

[305] Amazon Web Services, "Configuring function memory (console)," AWS Lambda documentation, 2022. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/configuration-function-common.html#configuration-memory-console

[306] H. Wang, D. Niu, and B. Li, "Distributed machine learning with a serverless architecture," in *IEEE Conference on Computer Communications (INFOCOM)*, 2019, pp. 1288–1296. doi:10.1109/INFOCOM.2019.8737391

[307] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, pp. 561–580, 2007. [Online]. Available: http://content.iospress.com/articles/intelligent-data-analysis/ida00303

[308] M. Brooker, A. C. Catangiu, M. Danilov, A. Graf, C. MacCárthaigh, and A. Sandu, "Restoring uniqueness in microvm snapshots," *CoRR*, vol. abs/2102.12892, 2021. [Online]. Available: https://arxiv.org/abs/2102.12892

[309] P. Maissen, P. Felber, P. G. Kropf, and V. Schiavoni, “Faasdom: a benchmark suite for serverless computing,” in *Proceedings of the 14th ACM International Conference on Distributed and Event-based Systems (DEBS)*, 2020, pp. 73–84. doi:10.1145/3401025.3401738

[310] B. Minic, “Improving cold start times of Java AWS Lambda functions using GraalVM and native images,” 2021. [Online]. Available: https://shinesolutions.com/2021/08/30/improving-cold-start-times-of-java-aws-lambda-functions-using-graalvm-and-native-images/

[311] J. Dean and L. A. Barroso, “The tail at scale,” *Communications of the ACM*, vol. 56, pp. 74–80, 2013. [Online]. Available: http://cacm.acm.org/magazines/2013/2/160173-the-tail-at-scale/fulltext

[312] T. Zhang, D. Xie, F. Li, and R. Stutsman, “Narrowing the gap between serverless and its state with storage functions,” in *Proceedings of the ACM Symposium on Cloud Computing (SoCC)*, 2019, pp. 1–12. doi:10.1145/3357223.3362723

[313] P. G. López, A. Arjona, J. Sampé, A. Slominski, and L. Villard, “Triggerflow: trigger-based orchestration of serverless workflows,” in *The 14th ACM International Conference on Distributed and Event-based Systems (DEBS)*, 2020, pp. 3–14. doi:10.1145/3401025.3401731

[314] C. Lin and H. Khazaei, “Modeling and optimization of performance and cost of serverless applications,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, pp. 615–632, 2020. doi:10.1109/TPDS.2020.3028841

[315] A. Filichkin, “GraalVM + AWS Lambda or solving Java cold start problem,” 2021. [Online]. Available: https://filia-aleks.medium.com/graalvm-aws-lambda-or-solving-java-cold-start-problem-2655eeee98c6

[316] B. Schaatsbergen, “Pre-jitting in AWS Lambda functions,” 2021. [Online]. Available: https://web.archive.org/web/20210807184839/https://www.bschaatsbergen.com/pre-jitting-in-lambda

[317] S. Kuenzer, V. Badoiu, H. Lefeuvre, S. Santhanam, A. Jung, G. Gain, C. Soldani, C. Lupu, S. Teodorescu, C. Raducanu, C. Banu, L. Mathy, R. Deaconescu, C. Raiciu, and F. Huici, “Unikraft: fast, specialized unikernels the easy way,” in *16th European Conference on Computer Systems (EuroSys)*, 2021, pp. 376–394. doi:10.1145/3447786.3456248

[318] C. Soto-Valero, T. Durieux, N. Harrand, and B. Baudry, “Coverage-based debloating for java bytecode,” *ACM Transactions on Software Engineering and Methodology*, 2022. doi:10.1145/3546948

[319] A. Tariq, A. Pahl, S. Nimmagadda, E. Rozner, and S. Lanka, “Sequoia: Enabling quality-of-service in serverless computing,” in *Proceedings of the ACM Symposium on Cloud Computing (SoCC)*, 2020, pp. 311–327. doi:10.1145/3419111.3421306

[320] D. C.-C. Walter Erquinigo and A. Tang, "Reverse debugging at scale," Meta, 2021. [Online]. Available: https://engineering.fb.com/2021/04/27/developer-tools/reverse-debugging/

[321] D. Balla, M. Maliosz, and C. Simon, "Performance evaluation of asynchronous FaaS," in *14th IEEE International Conference on Cloud Computing (CLOUD)*, 2021, pp. 147–156. doi:10.1109/CLOUD53861.2021.00028

[322] J. Czentye, I. Pelle, A. Kern, B. P. Gero, L. Toka, and B. Sonkoly, "Optimizing latency sensitive applications for Amazon's public cloud platform," in *IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–7. doi:10.1109/GLOBECOM38437.2019.9013988

[323] J. Brutlag, "Speed matters for google web search," Google, Tech. Rep., 2009. [Online]. Available: https://ai.googleblog.com/2009/06/speed-matters.html

[324] R. Kohavi and R. Longbotham, "Online experiments: Lessons learned," *Computer*, vol. 40, pp. 103–05, 2007. doi:10.1109/MC.2007.328

[325] J. Wen, Z. Chen, Y. Liu, Y. Lou, Y. Ma, G. Huang, X. Jin, and X. Liu, "An empirical study on challenges of application development in serverless computing," in *29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2021, pp. 416–428. doi:10.1145/3468264.3468558

[326] E. van Eyk, "Function composition in a serverless world," 2018. [Online]. Available: https://fission.io/blog/function-composition-in-a-serverless-world-video/

[327] L. Zhu, G. Giotis, V. Tountopoulos, and G. Casale, "RDOF: deployment optimization for function as a service," in *14th IEEE International Conference on Cloud Computing (CLOUD)*, 2021, pp. 508–514. doi:10.1109/CLOUD53861.2021.00066

[328] A. Mahéo, P. Sutra, and T. Tarrant, "The serverless shell," in *Proceedings of the 22nd International Middleware Conference: Industrial Track*, 2021, pp. 9–15. doi:10.1145/3491084.3491426

[329] D. A. Patterson, "Latency lags bandwith," *Communication of the ACM*, vol. 47, pp. 71–75, 2004. doi:10.1145/1022594.1022596

[330] T. Hoefler and R. Belli, "Scientific benchmarking of parallel computing systems: Twelve ways to tell the masses when reporting performance results," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2015. doi:10.1145/2807591.2807644

[331] J. Scheuner, S. Eismann, S. Talluri, E. V. Eyk, C. L. Abad, P. Leitner, and A. Iosup, "Let's trace it: Fine-grained serverless benchmarking using synchronous and asynchronous orchestrated applications," *CoRR*, vol. abs/2205.07696, 2022. doi:10.48550/arXiv.2205.07696

[332] T. Killalea, "A second conversation with werner vogels: The amazon cto sits with tom killalea to discuss designing for evolution at scale." *Queue*, vol. 18, pp. 67–92, 2020. doi:10.1145/3434571.3434573

[333] S. Sivasubramanian, "Amazon DynamoDB: a seamlessly scalable non-relational database service," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2012, pp. 729–730. doi:10.1145/2213836.2213945

[334] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, J. Haridas, C. Uddaraju, H. Khatri, A. Edwards, V. Bedekar, S. Mainali, R. Abbasi, A. Agarwal, M. F. ul Haq, M. I. ul Haq, D. Bhardwaj, S. Dayanand, A. Adusumilli, M. McNett, S. Sankaran, K. Manivannan, and L. Rigas, "Windows Azure Storage: a highly available cloud storage service with strong consistency," in *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP)*, 2011, pp. 143–157. doi:10.1145/2043556.2043571

[335] Y. Xu, Z. Musgrave, B. Noble, and M. Bailey, "Bobtail: Avoiding long tails in the cloud," in *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, vol. 13, 2013, pp. 329–341. [Online]. Available: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/xu_yunjing

[336] A. Chu, ".NET on Azure Functions roadmap," Microsoft, 2021. [Online]. Available: https://techcommunity.microsoft.com/t5/apps-on-azure-blog/net-on-azure-functions-roadmap/ba-p/2197916

[337] N. Daw, U. Bellur, and P. Kulkarni, "Xanadu: Mitigating cascading cold starts in serverless function chain deployments," in *Proceedings of the 21st International Middleware Conference*, 2020, pp. 356–370. doi:10.1145/3423211.3425690

[338] Microsoft Azure, "Choose between Azure messaging services – Event Grid, Event Hubs, and Service Bus," 2022. [Online]. Available: https://docs.microsoft.com/en-us/azure/event-grid/compare-messaging-services

[339] M. Hüttermann, *Infrastructure as Code.* Apress, 2012. doi:10.1007/978-1-4302-4570-4_9

[340] S. Bhatia and J. Ridoux, "Manage Amazon EC2 instance clock accuracy using Amazon Time Sync Service and Amazon CloudWatch," AWS, 2021. [Online]. Available: https://aws.amazon.com/blogs/mt/manage-amazon-ec2-instance-clock-accuracy-using-amazon-time-sync-service-and-amazon-cloudwatch-part-1/

[341] Microsoft Azure, "Time sync for Windows VMs in Azure," 2022. [Online]. Available: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/time-sync

[342] T. Palit, Y. Shen, and M. Ferdman, “Demystifying cloud benchmarking,” in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2016, pp. 122–132. doi:10.1109/ISPASS.2016.7482080

[343] J. Dejun, G. Pierre, and C.-H. Chi, “EC2 performance analysis for resource provisioning of service-oriented applications,” in *Proceedings of the 7th ICSOC / 2nd ServiceWave Workshops*, vol. 6275, 2009, pp. 197–207. doi:10.1007/978-3-642-16132-2_19

[344] A. H. Borhani, P. Leitner, B. Lee, X. Li, and T. Hung, “WPress: An application-driven performance benchmark for cloud-based virtual machines,” in *Proceedings of the 18th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*, 2014, pp. 101–109. doi:10.1109/EDOC.2014.23

[345] J. Scheuner, J. Cito, P. Leitner, and H. Gall, “Cloud WorkBench: Benchmarking IaaS providers based on infrastructure-as-code,” in *Companion of the 24th International Conference on World Wide Web (WWW Demo)*, 2015, pp. 239–242. doi:10.1145/2740908.2742833

[346] Cloud Spectator, “2017 top 10 european cloud providers,” 2017. [Online]. Available: https://cloudspectator.com/top-10-european-cloud-service-providers/

[347] SPEC, “SPEC Cloud™ IaaS 2016 Benchmark,” 2016. [Online]. Available: http://spec.org/cloud_iaas2016/

[348] M. Cunha, N. Mendonça, and A. Sampaio, “A declarative environment for automatic performance evaluation in IaaS clouds,” in *Proceedings of the 6th IEEE International Conference on Cloud Computing (CLOUD)*, 2013, pp. 285–92. doi:10.1109/CLOUD.2013.12

[349] J. Scheuner, “Cloud benchmarking – estimating cloud application performance based on micro benchmark profiling,” Master Thesis, University of Zurich, 2017. [Online]. Available: https://www.merlin.uzh.ch/publication/show/15364

[350] J. Scheuner and P. Leitner, “A cloud benchmark suite combining micro and applications benchmarks,” in *Companion of the 9th ACM/SPEC ICPE: 4th Workshop on Quality-Aware DevOps (QUDOS)*, 2018, pp. 161–166. doi:10.1145/3185768.3186286

[351] J. O’Loughlin and L. Gillam, “Towards performance prediction for public infrastructure clouds: An EC2 case study,” in *Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 1, 2013, pp. 475–80. doi:10.1109/CloudCom.2013.69

[352] A. Li, X. Yang, S. Kandula, and M. Zhang, “CloudCmp: Comparing public cloud providers,” in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2010, pp. 1–14. doi:10.1145/1879141.1879143

[353] D. Cerotti, M. Gribaudo, P. Piazzolla, and G. Serazzi, "Flexible CPU provisioning in clouds: A new source of performance unpredictability," in *Proceedings of the 9th International Conference on Quantitative Evaluation of Systems (QEST)*, 2012, pp. 230–37. doi:10.1109/QEST.2012.23

[354] S. K. Barker and P. Shenoy, "Empirical evaluation of latency-sensitive application performance in the cloud," in *Proceedings of the 1st ACM SIGMM Conference on Multimedia Systems (MMSys)*, 2010, pp. 35–46. doi:10.1145/1730836.1730842

[355] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of amazon EC2 data center," in *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM)*, 2010, pp. 1163–1171. doi:10.1109/INFCOM.2010.5461931

[356] M. Canuto, R. Bosch, M. Macias, and J. Guitart, "A methodology for full-system power modeling in heterogeneous data centers," in *Proceedings of the 9th International Conference on Utility and Cloud Computing (UCC)*, 2016, pp. 20–29. doi:10.1145/2996890.2996899

[357] K. Hoste, A. Phansalkar, L. Eeckhout, A. Georges, L. K. John, and K. De Bosschere, "Performance prediction based on inherent program similarity," in *Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2006, pp. 114–122. doi:10.1145/1152154.1152174

[358] C. Stewart and K. Shen, "Performance modeling and system management for multi-component online services," in *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation (NSDI)*, 2005, pp. 71–84. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251203.1251209

[359] B. G. Tabachnick, L. S. Fidell, and J. B. Ullman, *Using Multivariate Statistics.* Pearson, 2012.

[360] C. Lowery, R. Bala, L. Leong, and D. Smith, "Magic quadrant for cloud infrastructure as a service, worldwide," *Gartner*, 2017. [Online]. Available: https://www.gartner.com/en/documents/3738058/magic-quadrant-for-cloud-infrastructure-as-a-service-wor

[361] J. Cito, P. Leitner, T. Fritz, and H. C. Gall, "The making of cloud applications: An empirical study on software development for the cloud," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, 2015, pp. 393–403. doi:10.1145/2786805.2786826

[362] C. Laaber and P. Leitner, "An evaluation of open-source software microbenchmark suites for continuous performance assessment," in *Proceedings of the 15th International Conference on Mining Software Repositories (MSR)*, 2018, pp. 119–130. doi:10.1145/3196398.3196407

[363] L. Bulej, V. Horký, and P. Tuma, "Do we teach useful statistics for performance evaluation?" in *Companion Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering (ICPE)*, 2017, pp. 185–189. doi:10.1145/3053600.3053638

[364] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney, "Producing wrong data without doing anything obviously wrong!" *SIGPLAN Not.*, vol. 44, pp. 265–276, 2009. doi:10.1145/1508284.1508275

[365] A. Georges, D. Buytaert, and L. Eeckhout, "Statistically rigorous java performance evaluation," in *Proceedings of the 22Nd Annual ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications (OOPSLA)*, 2007, pp. 57–76. doi:10.1145/1297027.1297033

[366] N. Cliff, *Ordinal Methods for Behavioral Data Analysis.* Psychology Press, 1996. doi:10.4324/9781315806730

[367] J. Romano, J. Kromrey, J. Coraggio, and J. Skowronek, "Appropriate statistics for ordinal level data: Should we really be using t-test and cohen'sd for evaluating group differences on the nsse and other surveys?" in *Annual Meeting of the Florida Association of Institutional Research*, 2006, pp. 1–3.

[368] R. Jain, *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* Wiley, 1991.

[369] L. K. John and L. Eeckhout, *Performance Evaluation and Benchmarking.* CRC Press, 2005.

[370] A. C. Davison and D. Hinkley, *Bootstrap Methods and Their Application.* Cambridge University Press, 1997, vol. 94. doi:10.1017/CBO9780511802843

[371] S. Ren, H. Lai, W. Tong, M. Aminzadeh, X. Hou, and S. Lai, "Nonparametric bootstrapping for hierarchical data," *Journal of Applied Statistics*, vol. 37, pp. 1487–1498, 2010. doi:10.1080/02664760903046102

[372] T. C. Hesterberg, "What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum," *The American Statistician*, vol. 69, pp. 371–86, 2015. doi:10.1080/00031305.2015.1089789

[373] C. M. Woodside, G. Franks, and D. C. Petriu, "The future of software performance engineering," in *Workshop on the Future of Software Engineering (FOSE) at International Conference on Software Engineering (ISCE)*, 2007, pp. 171–187. doi:10.1109/FOSE.2007.32

[374] M. M. Arif, W. Shang, and E. Shihab, "Empirical study on the discrepancy between performance testing results from virtual and physical environments," *Empirical Software Engineering*, vol. 23, pp. 1490–1518, 2018. doi:10.1007/s10664-017-9553-x

[375] D. A. Menascé, "Load testing of web sites," *IEEE Internet Comput.*, vol. 6, pp. 70–74, 2002. doi:10.1109/MIC.2002.1020328

[376] Z. M. Jiang and A. E. Hassan, "A survey on load testing of large-scale software systems," *IEEE Trans. Software Eng.*, vol. 41, pp. 1091–1118, 2015. doi:10.1109/TSE.2015.2445340

[377] E. J. Weyuker and F. I. Vokolos, "Experience with performance testing of software systems: Issues, an approach, and case study," *IEEE Trans. Software Eng.*, vol. 26, pp. 1147–1156, 2000. doi:10.1109/32.888628

[378] C. Barna, M. Litoiu, and H. Ghanbari, "Autonomic load-testing framework," in *Proceedings of the 8th International Conference on Autonomic Computing (ICAC)*, 2011, pp. 91–100. doi:10.1145/1998582.1998598

[379] T. H. D. Nguyen, M. Nagappan, A. E. Hassan, M. N. Nasser, and P. Flora, "An industrial case study of automatically identifying performance regression-causes," in *11th Working Conference on Mining Software Repositories (MSR)*, 2014, pp. 232–241. doi:10.1145/2597073.2597092

[380] K. C. Foo, Z. M. Jiang, B. Adams, A. E. Hassan, Y. Zou, and P. Flora, "An industrial case study on the automated detection of performance regressions in heterogeneous environments," in *37th IEEE/ACM International Conference on Software Engineering (ICSE)*, 2015, pp. 159–168. doi:10.1109/ICSE.2015.144

[381] M. Grechanik, C. Fu, and Q. Xie, "Automatically finding performance problems with feedback-directed learning software testing," in *34th International Conference on Software Engineering (ICSE)*, 2012, pp. 156–166. doi:10.1109/ICSE.2012.6227197

[382] L. Gillam, B. Li, J. O'Loughlin, and A. Tomar, "Fair benchmarking for cloud computing systems," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, 2013. doi:10.1186/2192-113X-2-6