Empirical Software Engineering (2025) 30:59
https://doi.org/10.1007/s10664-025-10615-3

Unveiling overlooked performance variance in serverless computing

Jinfeng Wen[1,2] · Zhenpeng Chen[3] · Federica Sarro[4] · Shangguang Wang[1,2]

Accepted: 10 January 2025

Abstract

Serverless computing is an emerging cloud computing paradigm for developing applications at the function level, known as *serverless functions*. Due to the highly dynamic execution environment, multiple identical runs of the same serverless function can yield different performance, specifically in terms of end-to-end response latency. However, surprisingly, our analysis of serverless computing-related papers published in top-tier conferences highlights that the research community lacks awareness of the performance variance problem, with only 38.38% of these papers employing multiple runs for quantifying it. To further investigate, we analyze the performance of 72 serverless functions collected from these papers. Our findings reveal that the performance of these serverless functions can differ by up to 338.76% (44.28% on average) across different runs. Moreover, 61.11% of these functions produce unreliable performance results, with a low number of repetitions commonly employed in the serverless computing literature. Our study highlights a lack of awareness in the serverless computing community regarding the well-known performance variance problem in software engineering. The empirical results illustrate the substantial magnitude of this variance, emphasizing that ignoring the variance can affect research reproducibility and result reliability.

Keywords Serverless computing · Performance variance · Empirical study · Cloud computing

Communicated by: Philipp Leitner

✉ Zhenpeng Chen
zhenpeng.chen@ntu.edu.sg

Jinfeng Wen
jinfeng.wen@bupt.edu.cn

Federica Sarro
f.sarro@ucl.ac.uk

Shangguang Wang
sgwang@bupt.edu.cn

[1] Beijing University of Posts and Telecommunications, Beijing, China

[2] Beiyou Shenzhen Institute, Beijing, China

[3] Nanyang Technological University, Singapore, Singapore

[4] University College London, London, United Kingdom

Published online: 21 January 2025

1 Introduction

Serverless computing is a rapidly growing cloud computing paradigm. It relieves software developers from the complexities and potential errors of cloud infrastructure management and has been widely adopted in various software applications, such as video processing (Fouladi et al. 2017), machine learning (Jiang et al. 2021), and big data analytics (Müller et al. 2020). Predictions indicate that by 2025, serverless computing will be adopted by around 50% of global enterprises (https://www.gartner.com/smarterwithgartner/the-cios-guide-to-serverless-computing). Its market size is also expected to skyrocket from $3 billion in 2017 to $22 billion by 2025 (https://www.researchandmarkets.com/reports/4828585/serverless-architecture-market-by-deployment).

By allowing software developers to focus on application logic, serverless computing empowers them to develop software applications as event-driven, stateless functions, commonly referred to as *serverless functions*. Leading cloud service providers have introduced dedicated serverless platforms to facilitate the execution of serverless functions, including AWS Lambda (https://docs.aws.amazon.com/lambda/latest/dg/welcome.html) and Google Cloud Functions (https://cloud.google.com/functions).

The growing presence of serverless computing in software domains has captured the attention of the Software Engineering (SE) field (Wen et al. 2023). The SE community has studied a broad range of topics about serverless computing, including its development characteristics (Eismann et al. 2021b), programming frameworks (Bermbach et al. 2022), application migration (Ristov et al. 2020), economic impact (Adzic and Chatley 2017), testing/debugging (Lenarduzzi and Panichella 2020), and performance optimization (Liu et al. 2023). Notably, performance has emerged as the most widely studied topic within the serverless computing literature (Wen et al. 2023). Ensuring accurate and reliable serverless computing performance is critical for research reproducibility and result reliability.

In the SE community, it is well recognized that multiple identical runs of the same application can exhibit varying performance (Pham et al. 2020; Qian et al. 2021; Weber et al. 2021; He et al. 2019a, 2021a). Serverless computing-based applications should not be an exception to the *performance variance* phenomenon. Multiple identical runs of the same serverless function can yield different performance, specifically in end-to-end response latency. Several factors can contribute to this performance variance: *1) Highly dynamic cloud underlying infrastructure.* Serverless platforms operate in a cloud environment susceptible to multi-tenancy and networking challenges (Papadopoulos et al. 2019; Mahgoub et al. 2022). Moreover, due to opaque instance scheduling and unpredictable invocations of serverless platforms (Patterson et al. 2022; Fuerst and Sharma 2022; Perron et al. 2020), the execution environment of serverless functions may be highly variable and dynamic. *2) High-density deployment of lightweight functions.* Serverless functions generally have small memory requirements and are hosted in small instances, leading to high-density deployments (Shahrad et al. 2020; Singhvi et al. 2021). This deployment environment increases the risk of disruptions (Zhao et al. 2021; Patterson et al. 2022).

To date, however, serverless computing research seems to have overlooked the well-known issue of performance variance. To address this gap, in this paper, we present an empirical study that sheds light on this oversight with solid scientific evidence. Additionally, we provide a comprehensive characterization of the magnitude of performance variance in serverless computing, emphasizing the significant consequences of disregarding this crucial aspect.

To this end, we collect and analyze 99 research papers related to serverless computing performance from 77 top-tier academic conferences. Our findings reveal that only 38.38%

of these research papers employ multiple runs to quantify the variance in serverless function performance. This shows that the current serverless computing literature lacks awareness for the well-known performance variance problems in SE.

Furthermore, we extract 72 serverless functions from these research papers and measure their end-to-end response latencies across multiple runs to assess the extent of performance variance. Our analysis demonstrates that serverless function performance can vary significantly, with a maximum variance of 338.76% (44.28% on average) observed among different runs.

Finally, we assess the reliability of serverless function performance obtained at commonly used numbers of repetitions within the serverless computing literature. Our investigation uncovers that under these conditions, 98.61% of the serverless functions require at least 50 executions to achieve reliability. This emphasizes the need for significantly more repetitions in measuring the serverless function performance, surpassing conventional practices in the existing literature.

Our findings offer practical implications for different stakeholders in the research community. **1) Researchers** should thoroughly assess serverless function performance variance, detailing their measurement approach, including repetitions and reasoning, to ensure research reproducibility and result reliability. Furthermore, a potential research space needs to be explored to develop novel performance testing techniques specialized for serverless computing, enhancing the accuracy and reliability of performance assessments. **2) Software developers** can customize strategies to mitigate the severe performance variance in serverless computing-based applications, ensuring consistent user experiences. **3) Cloud providers** offering serverless computing environments can provide consistent quality of service when delivering the services in a serverless computing manner to meet the demands of customers who seek stable performance.

In summary, this paper makes the following contributions:

- An empirical study highlighting the lack of awareness within the serverless computing community regarding the well-known performance variance problem in software engineering.
- A measurement study that illustrates the substantial magnitude of performance variance in serverless computing, underscoring the significant consequences of neglecting this critical aspect.
- A replication package (https://github.com/JinfengWenWork/ServerlessPerformanceVariance) including the paper's data and scripts, serving as a benchmark dataset for future research on performance in serverless computing.

2 Background

We start by introducing the background knowledge of serverless computing and performance of serverless functions.

2.1 Serverless Computing

Serverless computing is a popular cloud computing paradigm, which allows software developers to focus on their application logic without having to manage complex cloud underlying tasks. Function-as-a-Service (FaaS) is the most prominent implementation of serverless computing (Akkus et al. 2018; Taibi et al. 2020; Wen et al. 2021a; Copik et al. 2021; Jonas et al.

2019). Thus, in this study, we focus on FaaS. In the FaaS fashion, software developers implement applications as a series of event-driven and stateless dedicated functions, called *serverless functions*. They deploy and execute serverless functions on serverless platforms, such as AWS Lambda (https://docs.aws.amazon.com/lambda/latest/dg/welcome.html) and Google Cloud Functions (https://cloud.google.com/functions).

Figure 1 illustrates the process of developing, deploying, and executing serverless functions. ① First, software developers implement serverless functions using programming interfaces as event-driven and stateless functions (Wen et al. 2023). These functions are generally written in high-level programming languages, e.g., Python and JavaScript (https://www.serverless.com/blog/2018-serverless-community-survey-huge-growth-usage, https://www.datadoghq.com/state-of-serverless/, Eskandani and Salvaneschi (2021)). ② Meanwhile, software developers can define the rules to bind their serverless functions and the related events, such as HTTP requests and data changes of cloud storage. ③ Then, serverless functions and required dependent libraries are packaged together and deployed to serverless platforms. In this stage, software developers can specify the required configurations, including language runtime, memory size, and function timeout time (Wen et al. 2023). ④ When the serverless functions are triggered by pre-defined events, the serverless platforms can automatically launch or reuse the required function instances (e.g., VMs or containers) with restricted resources (e.g., CPU and memory) to serve these requests. This frees software developers from complex server management and makes their requests benefit from seamless resource elasticity. ⑤ After the requests are completed, software developers pay for the number of requests and the actually allocated or consumed resources (Lin and Khazaei 2020; Wen et al. 2023).

2.2 Performance of Serverless Functions

In this study, we focus on evaluating the performance of serverless functions by examining the end-to-end response latency (abbreviated as *e2eTime*). The end-to-end response latency is a widely adopted metric in the serverless computing literature (Yu et al. 2022; Scheuner and Leitner 2020; Lin and Khazaei 2020; Liu et al. 2023; Wen et al. 2021b). It is defined as the duration as when a request is sent from a client until its completion. This metric is considered the primary metric for assessing serverless function performance, and serverless computing research actively proposes optimization techniques targeting this metric (Wang et al. 2021; Mahgoub et al. 2022; Zhao et al. 2021). Additionally, practitioners commonly use end-to-end response latency as a reflection of user experience satisfaction (Eismann et al. 2021b; Wen et al. 2021b, a).

Generally, the response latency of a serverless function is divided into cold-start response latency and warm-start response latency. (1) When a serverless platform launches new func-

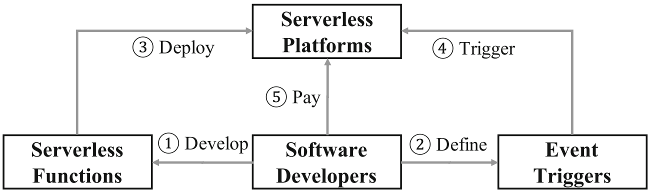

Fig. 1 The process of developing, deploying, and executing serverless functions by software developers

tion instances to handle requests, the serverless function encounters cold-start invocations. This kind of invocation involves the preparation process of function instances. Thus, this latency is referred to as the cold-start response latency. (2) On subsequent invocations of the same serverless function within a short keep-alive time, the serverless platform can reuse the previously launched function instances. This leads to warm-start invocations for the serverless function and produces warm-start response latency. In the absence of requests, the serverless platform automatically turns the launched function instances into an idle state and releases the corresponding resources. Generally, warm-start response latency tends to be smaller than the cold-start response latency for the same serverless function due to the absence of the preparation process of function instances during warm starts.

3 Methodology

To understand the variance of serverless function performance, we first identify relevant research papers and collect serverless functions from them. Based on them, we answer three research questions (RQs) related to the variance of serverless function performance. In Fig. 2, we show an overview of our methodology.

3.1 Research Questions

RQ1 (Current literature) *To what extent has the current literature addressed the variance of serverless function performance?* This RQ investigates whether and how researchers consider the performance variance of serverless functions when studying serverless computing, given that performance variance has been a well-known issue in SE.

RQ2 (Variance measurement) *How variable is the performance of serverless functions?* Despite the well-known performance variance issue, the magnitude of this variance in serverless computing remains unclear. If the variance is substantial, researchers and developers cannot disregard it during the development of serverless computing-based applications. This RQ aims to fill this knowledge gap and provide actionable insights for stakeholders.

RQ3 (Reliability and repetitions) *How reliable are the serverless function performance results obtained from different repetitions?* This RQ aims to explore the reliability of serverless function performance across multiple repetitions, highlighting the importance of repetition settings on the reliability of performance results. Additionally, it seeks to provide guidance on determining the appropriate number of repetitions needed to achieve more reliable performance measurements.

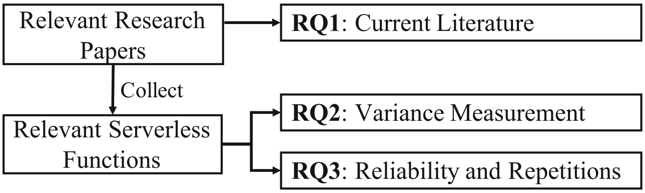
Relevant Research Papers
RQ1: Current Literature
Collect
Relevant Serverless Functions
RQ2: Variance Measurement
RQ3: Reliability and Repetitions

Fig. 2 An overview of our methodology

3.2 Collection of Relevant Papers

To address RQ1, we collect research papers related to serverless computing. Since studies on serverless computing have been published in academic venues across various research communities (Wen et al. 2023), we gather research papers from all 77 conferences listed in the Computer Science Rankings (CSRankings) (https://csrankings.org). CSRankings provides a curated list of top-tier conferences from various areas of computer science, including Software Engineering (e.g., ICSE, FSE, ASE, and ISSTA), Operating Systems (e.g., OSDI, SOSP, EuroSys, FAST, and ATC), and Computer Networks (e.g., SIGCOMM and NSDI). These conferences are recognized not only for their academic reputation but also for their influence in the respective research communities. We focus on papers published in these high-tier conferences because they reflect state-of-the-art research and provide a comprehensive view of the latest findings in serverless computing across diverse areas. We do not use search engines (e.g., Google Scholar) and citation-based thresholds to collect impactful papers related to serverless computing. This is because defining a consistent and objective citation threshold is challenging, as citation counts can be influenced by factors such as publication date, research topic popularity, and self-citation. Moreover, there is no universally accepted standard for what constitutes “impactful” based on citations. We do not evaluate the quality of papers based on their authors or assume that work published in other venues is of lower quality, as such judgments could introduce bias. Instead, we adopt a venue-based selection criterion, relying on CSRankings to identify high-quality, peer-reviewed research from top-tier conferences across various areas of computer science. This approach minimizes subjectivity and ensures consistency. However, we acknowledge that including papers from other venues (such as journals and other conferences) in our study would provide a more comprehensive view of the research landscape. In future work, we plan to expand our study to include more research papers to further enhance the scope of our research.

The process for collecting research papers is as follows. First, we collect all technical papers (excluding short papers, tutorials, posters, workshop papers, industry track papers, etc.) published in the 77 conferences between 2014 (the year serverless computing was popularized (Wen et al. 2021a, 2023; Jonas et al. 2019)) and the date we collect the papers (April 10, 2024). Then, the first two authors independently review the research papers to select those related to serverless computing. A paper is considered relevant if, upon manual inspection of its title, abstract, and full text, the authors determine that it discusses or addresses specific problems in the field of serverless computing. If the authors disagree on the relevance of a paper, an arbitrator, who has ten years of cloud computing experience, is involved in discussing to reach an agreement. To measure the inter-rater agreement level of the authors during the independent labeling, we use Cohen’s Kappa (κ) (Cohen 1960), which is the most widely-used agreement evaluation metric in SE (Uta et al. 2020; Wen et al. 2021a, 2023). The value of κ is 0.952, indicating an almost perfect agreement and a reliable labeling procedure (Landis and Koch 1977). Finally, we obtain 110 research papers related to serverless computing. The number of related papers per year is shown in Fig. 3. We observe an overall increasing trend, with the number of relevant papers rising from 1 in 2017 to 33 in 2023. This implies that serverless computing is gaining increasing attention from the research community, demonstrating the timeliness and urgency of our study. The number of serverless computing-related papers collected from various areas of computer science is illustrated in Fig. 4. The three leading areas are Operating Systems, Computer Architecture, and High-Performance Computing, which collectively account for 40, 23, and 15 papers, respectively. Furthermore, the first two authors jointly filter papers that do not report the performance results of serverless functions. As a result, 99 out of the 110 research papers are

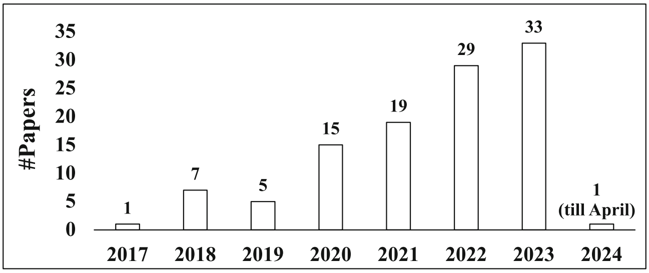

Fig. 3 Number of serverless computing-related papers in top-tier conferences per year

retained for our final performance analysis. This is consistent with findings reported by previous work (Wen et al. 2023; Li et al. 2023; Scheuner and Leitner 2020) that most serverless computing papers are related to serverless function performance. Specifically, the distribution is as follows: 5 papers in 2019 are reduced to 4 papers, 15 papers in 2020 are reduced to 14, 19 papers in 2021 are reduced to 15, 29 papers in 2022 are reduced to 26, and 33 papers in 2023 are reduced to 31. Details of the selected papers (including conference name, year, and paper title) are provided in our repository (https://github.com/JinfengWenWork/ServerlessPerformanceVariance).

3.3 Collection and Analysis of Serverless Functions

To answer RQ2 and RQ3, we extract serverless functions used in the 99 research papers. We follow previous work (Liu et al. 2023) to focus on serverless functions developed using the most widely adopted *serverless platforms* (i.e., AWS Lambda and Google Cloud Functions) and *programming languages* (i.e., Python and JavaScript). Moreover, we select serverless functions that are open-sourced and have the corresponding guidance documentation to assist with the execution. According to these criteria, the first two authors jointly extract and finally collect 72 serverless functions covering a wide range of tasks, such as Web request handling, video processing, scientific computing, machine learning, and natural language processing. Most of the collected serverless functions are included in commonly used benchmarks in serverless computing research (Yu et al. 2020; Kim and Lee 2019; Maissen et al. 2020) and industry (https://serverlessrepo.aws.amazon.com/applications), such as *Function-Bench* (Kim and Lee 2019), *ServerlessBench* (Yu et al. 2020), *AWS Sample* (https://github.

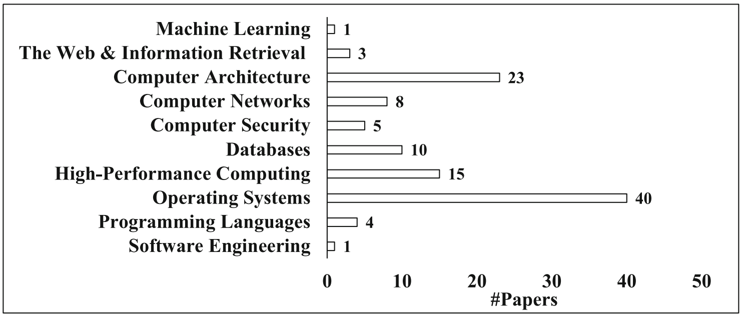

Fig. 4 Distribution of 110 serverless computing-related papers per computer science area

com/aws-samples), *SeBS* (Copik et al. 2021), and *FaaSDom* (Maissen et al. 2020). Among these serverless functions, 67 serverless functions are executed on AWS Lambda, while 5 serverless functions are executed on Google Cloud Functions. It can be because the advent of AWS Lambda was the main reason for the popularity of serverless computing (Wen et al. 2023; Jonas et al. 2019; Wen et al. 2021a). Moreover, 59 serverless functions are written in Python, while 13 serverless functions are written in JavaScript. Such a distribution can be attributed to the increasing number of tasks that use popular Python third-party libraries (Wen et al. 2023). We denote the 72 serverless functions as *Func1*, *Func2*, ..., and *Func72*. The number mapping information about serverless functions is provided in our repository (https://github.com/JinfengWenWork/ServerlessPerformanceVariance).

We then analyze the performance of the 72 serverless functions. We execute them with the input, configurations, and serverless platform used in their original papers. If no specific input is given in the original paper, we construct an input that matches the task execution according to the functionality of the serverless function. If no specific configurations are given, serverless functions are executed using the default configurations of serverless platforms, as developers typically rely on default configurations. At the time of our study, the default memory size of AWS Lambda is 128 MB (https://docs.aws.amazon.com/lambda/latest/operatorguide/computing-power.html), and its function timeout time is 3 seconds (https://docs.aws.amazon.com/lambda/latest/dg/configuration-timeout.html). For Google Cloud Functions, the default memory size and function timeout time are 256 MB (https://cloud.google.com/functions/docs/configuring/memory) and 60 seconds (https://cloud.google.com/functions/docs/configuring/timeout), respectively. We execute serverless functions to detect the sufficiency of their memory size or function timeout time. If the memory size or timeout is found to be insufficient during execution, the serverless platform reports errors and indicates the actual memory used or that the timeout is nearing its limit. In such cases, we continue to execute the functions by gradually increasing the memory size or function timeout until successful execution is achieved. To minimize the potential impact of the network variability on our measurements, we adopt the following strategies: (i) We use the same experimental machine in the same geographical area to send requests to the corresponding serverless platform and receive responses. (ii) We deploy our measured serverless functions to the same service region across different serverless platforms, such as “us-west-1” of AWS Lambda and “us-west1” of Google Cloud Functions.

To explore performance results across different times of the day and capture the comprehensive performance characteristics of serverless functions, we obtain response latencies of serverless functions for 50 trials. Meanwhile, to explore different performance types (especially “cold-start performance”) of serverless functions, we employ a different strategy of multiple interleaved trials, meaning that functions from different platforms and languages are executed in an interleaved manner, with each trial being separated by a fixed interval (e.g., half an hour). This strategy also mitigates the issue of functions being executed too closely in time and makes all performance results span at least one full day to capture variability across different times of the day. In each trial, we execute each tested serverless function twice using the same input and configuration to obtain the cold-start response latency and warm-start response latency, respectively. We set the fixed interval between each trial to half an hour, because we find that the resources used for execution can be released to ensure that the start of the next trial is still a cold start. This also reduces potential residual effects on subsequent trials. In addition, in each trial, the second execution of each function starts five seconds after the completion of the first execution, because we find that this duration can ensure the successful warm start of each serverless function. Finally, we obtain 72 × (50 + 50) = 7,200 data points (3,600 for cold start and 3,600 for warm start) about the end-to-end response latency of serverless functions. The execution results are produced between April 17, 2024 and May 10, 2024.

4 RQ1: Current Literature

RQ1 investigates whether and how the performance variance of serverless functions has been considered in the serverless computing literature, as this affects the reproducibility of conclusions drawn in research. To this end, we follow the standard criteria established by previous work (Uta et al. 2020) to analyze our selected 99 research papers related to serverless computing. Specifically, (Uta et al. 2020) defined standard criteria to determine if a study's conclusions are reproducible under performance variability, including checking whether:

- (1) Mean or median performance values are reported;
- (2) Confidence (e.g., confidence intervals) or variability (e.g., standard deviation, coefficient of variance, or percentiles) is reported;
- (3) The number of times an experiment was repeated is reported.

Such information can assist us in further understanding the reliability of the provided evidence and the study's conclusions. The first two authors separately read the full text of each paper to find out if it reports the aforementioned information. Moreover, if a paper reports the experiment repetition, i.e., *criterion (3)*, we also record the specific number of repetitions used in the paper. A paper may involve multiple pieces of information, which may result in the percentage of the total number of papers exceeding 100% when the information is counted. The inter-rater agreement values of labeling, measured by Cohen's Kappa (κ) (Cohen 1960), are 0.927, 0.914, and 0.893 for each type of information, respectively, indicating perfect agreement and a reliable labeling process (Landis and Koch 1977). During the labeling process, encountered conflicts are discussed to reach an agreement by the first two authors and the third arbitrator.

Results We observe that researchers lack awareness of the performance variance of serverless functions and often do not adequately describe related experimental settings, despite the fact that performance variance has become a well-known problem in SE. Specifically, Table 1 shows the results of our literature analysis. We observe that 62.63% (62 out of 99) of the papers report means or medians; only 37.37% (37 out of 99) report statistical variation data (e.g., percentiles and confidence intervals); and only 38.38% (38 out of 99) use multiple runs when conducting serverless computing-related experiments or performance analyses. In other words, over 60% of the research papers do not report how many times they repeated the experiments. This omission suggests that either no repetitions were performed or a specific number of repetitions was used but not disclosed. This lack of information affects the reproducibility of the results. We examine the details on reporting experiment repetitions in 99 final papers from various areas of computer science, shown in Table 2. The values are formatted as a/b, where a represents the number of papers that use experiment repeti-

Table 1 (RQ1) The details about experiment reporting for the studied 99 research papers

	Reporting Mean or Median	Reporting Statistical Variation Data	Reporting the Number of Repetitions
#Papers	62.63%	37.37%	38.38%

Table 2 (RQ1) Details about experiment repetitions reported in 99 final research papers from various areas of computer science, formatted as a/b

Machine Learning	The Web & Information Retrieval	Computer Architecture	Computer Networks	Computer Security
2/3	15/36	7/15	3/8	2/4
Databases	High-Performance Computing	Operating Systems	Programming Languages	
1/7	6/22	1/3	0/1	

a represents the number of papers that use experiment repetitions in each specific area and *b* is the total number of papers in this area

tions in each specific area and *b* is the total number of papers in this area. We observe that The Web&Information Retrieval has the largest representation with 36 papers, of which 15 papers report experiment repetitions. The areas of Computer Architecture (15 total papers) and High-Performance Computing (22 total papers) also show notable representation, with 7 and 6 papers, respectively, reporting experiment repetitions. Other areas, such as Machine Learning (2/3), Computer Networks (3/8), and Databases (1/7), have smaller numbers of papers reporting repetitions. Certain areas, e.g., Programming Languages, have minimal representation, with only 1 paper, which does not report repetitions (0/1). In summary, the reporting of experiment repetitions varies across the areas of computer science.

We further analyze the papers that report experiment repetitions. Figure 5 shows the number of repetitions used in these papers. Note that a paper can employ multiple numbers of repetitions, so the sum of the proportions in the figure may exceed 100%. There are 13 kinds of numbers of repetitions in the serverless computing literature. We find that 81.58% of these papers use no more than 50 repetitions. The top 3 frequently used repetitions are 10, 50, and 1,000, which account for 26.32%, 15.79%, and 13.16% of these papers, respectively.

Surprisingly, we find that none of these papers justify or explain the reason for the selection of the repetition number. In fact, the selection of this number is of high significance. If the number of repetitions is too low (e.g., 10 repetitions), the reliability of the obtained performance of serverless functions may not be guaranteed (as explained in Section 6), and it could lead to wrong or ambiguous conclusions. If the number of repetitions is too large

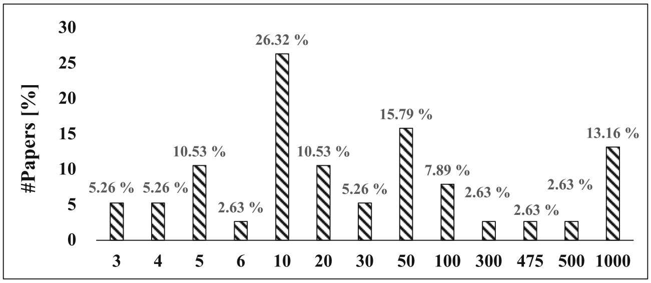

Fig. 5 (RQ1) Distribution of the number of repetitions used in the research papers that report experiment repetitions

(e.g., 1,000 repetitions), it may cause a huge runtime overhead, e.g., long experimentation time and high costs. In serverless computing, researchers need to pay for the number of requests and the actually allocated or consumed resources of serverless functions (Lin and Khazaei 2020; Wen et al. 2023). In this situation, a large repetition number can produce a huge cost. Thus, a too-small or too-large number of repetitions may be inappropriate in serverless computing-related experiments. A careful selection of the number of repetitions is needed in these experiments.

> **Finding 1**: The current serverless computing literature lacks awareness of the well-known performance variance problem in software engineering. Specifically, only 38.38% of the relevant papers use multiple runs to quantify the performance variance of serverless functions. 81.58% of the papers that report experiment repetitions use no more than 50 repetitions. Additionally, none of the papers provide any relevant description to justify the reason for the number of repetitions that they choose.

5 RQ2: Variance Measurement

RQ2 explores the magnitude of the performance variance of serverless functions. To this end, we analyze response latencies of 72 serverless functions over multiple runs from two aspects: *coefficient of variance (CV)* and *boxplot*, which are commonly used in performance variance analysis (Maricq et al. 2018; Guizzo et al. 2020; Zhao et al. 2021; Uta et al. 2020; Pham et al. 2020). Specifically, CV, known as the dispersion coefficient, is a statistical indicator that measures the degree of data variability. It represents the ratio of the standard deviation to the mean. Compared to standard deviation, CV provides a normalized, relative measure of dispersion and allows for comparison between data distributions with different units or means. A big value of CV means a large degree of performance variance. The boxplot can provide a visual representation of the dispersion degree for the performance results generated by each serverless function over multiple repetitions.

We use response latencies of each function over 50 repetitions to explore the magnitude of the variance. We chose 50 repetitions because this number is not less than the settings in over 80% of existing papers reporting repeated runs and is actually an upper value among them. We adopt this upper value to assume that most papers offer improved performance compared to the original setup. We investigate whether these assumed improved performance results suffer from a large magnitude of performance variance.

Results (1) *CV:* We analyze CV values calculated from response latencies generated by each serverless function under cold start and warm start. Figure 6 shows these results, where cold-start CV values are ordered in ascending order.

First, we observe that response latencies of serverless functions have different degrees of variance under cold and warm starts. Specifically, cold-start CV values range from 1.23% to 31.18%, while warm-start CV values range from 0.86% to 42.81%, differing by as much as 49 times and thus showing a wide range of variance. On average, the cold-start CV value is 6.07%, while the warm-start CV value is 8.39%. The median of cold-start CV values is 5.06%, while that of warm-start CV values is 6.47%. These results indicate that functions executed in cold and warm starts produce different degrees of variance in the end-to-end response latency.

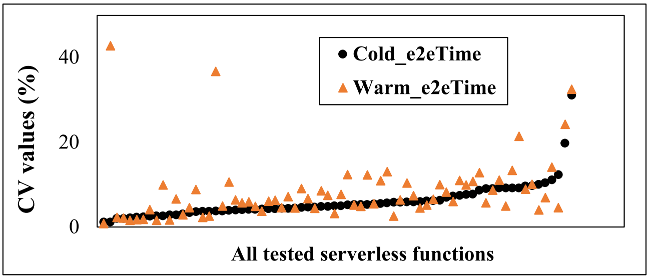

Fig. 6 (RQ2) CV values for cold-start response latency are ordered in ascending order. CV values for warm-start response latency are placed according to serverless functions that have been sorted by the cold-start CV values

However, providing only CV values is not a straightforward way to understand how the performance of serverless functions actually fluctuates. Thus, we give examples of the raw performance results of the serverless function. Figure 7 shows the performance data (50 data points) from cold starts for two serverless functions, $Func22$ and $Func2$. Multiple runs of the same function produced fluctuating response latencies, with no regular patterns of change. We also check the data from other serverless functions, but no consistent change patterns are found. The changes remain irregular, as seen in the data points in Fig. 7. The CV values of *Func22* and *Func2* are 2.04% and 5.88%, respectively. The difference between the maximum and minimum values of *Func22* achieves 6758.63 milliseconds (approximately 7 seconds), and that of *Func2* is 7244.32 milliseconds (also about 7 seconds). Generally, serverless functions execute short-lived and milliseconds-level tasks (Singhvi et al. 2021; Klimovic et al. 2018). Thus, the second-level difference size is severe for serverless functions. Moreover, from the right violin plot of Fig. 7, the distributions of data points of performance for different serverless functions are drastically different, being clustered in different positions. The data points of *Func22* are distributed near the middle of its violin plot, while those of *Func2* are distributed towards the bottom. Overall, we observe a significant variance in serverless function performance from two examples, where CV values are 2.04% and 5.88%, respectively. Therefore, we infer that other serverless functions with larger CV values than those shown in the examples will produce more significant fluctuations in performance.

We check the top 5 serverless functions with high CV values. For cold or warm starts, four serverless functions are executed on AWS Lambda and one on Google Cloud Functions.

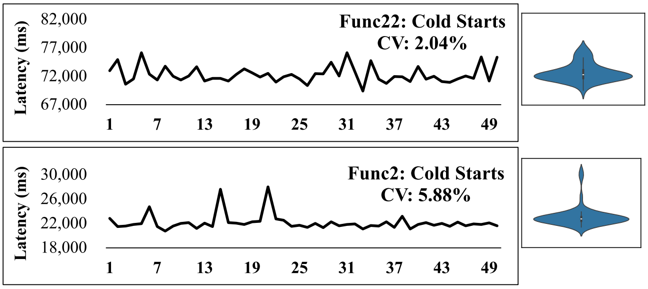

Fig. 7 (RQ2) Example of the generated serverless function performance. The left part shows 50 data points obtained in cold starts for *Func22* and *Func2*. The right part is the violin plot, whose thickness is proportional to the probability density of the data

This observation demonstrates that serverless functions with high CV values are not limited to a single serverless platform. The programming languages used in these functions include Python and JavaScript, which are not limited to a single programming language.

We further analyze the serverless functions with high CV values, e.g., *Func30* with warm-start CV value of 42.81%. The maximum and minimum values of response latencies of *Func30* differ by 338.76%, where the maximum value reaches 4.39 times its minimum value. The possible reason is that *Func30* needs to connect the cloud storage (e.g., AWS S3 https://aws.amazon.com/s3/) to transfer the file to the function instance through the network. However, the functionality of serverless functions that rely on platform library code to establish connections and transfer files with cloud services may lack robust design. Moreover, the network may be unstable. These likely result in unpredictable performance fluctuations. We check all serverless functions, and the corresponding maximum and minimum values of the response latency differ by a mean of 42.28% and a median of 29.67%.

To further investigate the impact of platforms and languages, we compare the latency differences between platforms and languages, as shown in Table 3. **Platform Comparison**: We compare the variability between Google Cloud Functions and AWS Lambda. For the mean, AWS Lambda shows a latency difference of 40.06%, whereas Google Cloud Functions shows 72.02%. In terms of median results, AWS Lambda has a latency difference of 29.24%, while Google Cloud Functions has 47.39%. This indicates that Google Cloud Functions experiences a more significant performance variability than AWS Lambda. **Language Comparison**: We compare Python and JavaScript. On average, Python-based functions exhibit a latency difference of 39.49%, while JavaScript functions show 54.92%. For the median, Python has a 29.24% difference, while JavaScript has 31.70%. This suggests that JavaScript has a higher performance variability than Python. **Platform and Language Interaction**: We explore the interaction between platforms and languages. Python-based serverless functions executed on AWS Lambda have a latency difference of 39.30%, whereas those on Google Cloud Functions show 44.85%. From the results, serverless functions written in JavaScript and executed on Google Cloud Functions exhibit the highest variability in both maximum and minimum latencies.

We further investigate how performance variance relates to the characteristics of the tasks by providing representative examples. We categorize some functions into three common types: CPU-memory-intensive tasks, I/O-intensive tasks, and network-intensive tasks. To explore this, we select three serverless functions from each category: for CPU-memory-intensive tasks, we examine *Func*5, *Func*6, and *Func*7; for I/O-intensive tasks, *Func*10, *Func*11, and *Func*33; and for network-intensive tasks, *Func*14, *Func*37, and *Func*38.

Table 3 (RQ2) A comparison of the performance variability across serverless platforms and languages by calculating the latency difference of the maximum and minimum values of the serverless function

	AWS Lambda	Google Cloud Functions	Python	JavaScript
Mean	40.06%	72.02%	39.49%	54.92%
Median	29.24%	47.39%	29.24%	31.70%
	AWS Lambda + Python	Google Cloud Functions + Python	AWS Lambda + JavaScript	Google Cloud Functions + JavaScript
Mean	39.30%	44.85%	44.36%	90.14%
Median	28.76%	47.39%	31.70%	46.20%

The latency differences between these task types are summarized in Table 4. Specifically, CPU-memory and network-intensive tasks show a latency difference of more than 30%, while I/O-intensive tasks exhibit a difference exceeding 20%. Similar trends are observed when considering median latency values. The results reveal different latency differences influenced by task characteristics, which serve as a preliminary exploration of the impact of task characteristics on performance variance.

Then, we observe that the variance of serverless function performance is more severe under warm starts than under cold starts. Specifically, the warm-start CV values for 65.28% (47/72) of the serverless functions are greater than those of the corresponding cold starts. We also calculate the number of the functions whose CV values are greater than 10%, which generally indicates a large degree of variance (Maricq et al. 2018; Zhao et al. 2021; Uta et al. 2020). In cold starts, there are 8.33% (6/72) functions, while warm starts have 29.17% (21/72). This implies that warm-start response latency has a more severe variance than cold-start response latency. One possible reason is that executing tasks in the reused function instances (i.e., warm starts) may be more susceptible to resource contention and underlying policies in serverless platforms. Another possible reason is that warm-start response latency tends to be smaller than cold-start response latency. The similar latency difference sizes may create the illusion of higher (lower) CV values when observed at smaller (larger) values.

(2) *Boxplot:* We use boxplots to visually show the distributions of response latencies of serverless functions. Before presenting the boxplots, we analyze the average latency of each serverless function. In cold starts, 36.11% (26/72) of functions have an average latency of under 2 seconds, 65.28% (47/72) have an average latency of under 5 seconds, and 83.33% (60/72) are under 10 seconds. In warm starts, 36.11% (26/72) have an average latency of under 1 second, 76.39% (47/72) are under 5 seconds, and 83.33% (60/72) are under 10 seconds. This highlights that most functions exhibit latencies in relatively small durations (e.g., milliseconds), in contrast to the minute- or hour-level latencies of traditional cloud applications. Since serverless functions have different levels of latency granularity, we apply the commonly used min-max normalization method (https://www.oreilly.com/library/view/hands-on-machine-learning/9781788393485/fd5b8a44-e9d3-4c19-bebb-c2fa5a5ebfee.xhtml) to normalize each set of response latencies generated by the serverless function to the 0 to 1 interval. Figures 8 and 9 respectively show the normalized boxplot about the response latency of 72 serverless functions under cold start and warm start. The bottom of the box represents the value at the 25*th* percentile, while its top is the 75*th* percentile. The line in the box is the median, i.e., 50*th* percentile. Dots outside of the whiskers are outliers. We observe that most response latencies (i.e., from the 25*th* percentile to the 75*th* percentile) of the serverless functions do not fall near the middle, i.e., 0.5 of the 0 to 1 interval. This occurs under both cold starts and warm starts. Moreover, the range sizes of most response latencies are inconsistent, indicating that there may not be a fixed distribution pattern for serverless function performance.

Table 4 (RQ2) A comparison of performance variance of serverless functions in the context of their exhibited characteristics by calculating the latency difference of the maximum and minimum values

	CPU-Memory Task	I/O Task	Network Task
Functions	Func5, Func6, Func7	Func10, Func11, Func33	Func14, Func37, Func38
Mean	38.94%	27.06%	34.45%
Median	34.68%	21.77%	36.60%

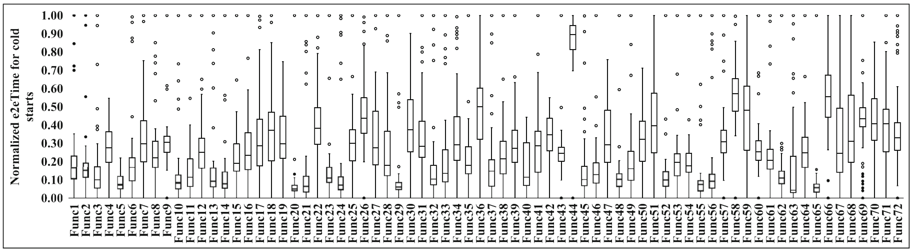

Fig. 8 (RQ2) The normalized boxplot for the cold-start response latencies of 72 serverless functions

To further determine if the data distribution of serverless function performance is concentrated or skewed, we check whether the median of response latencies falls within an error interval. This interval represents the 1% error above or below the middle value calculated from the maximum and minimum values. If the median is in this error interval, the data distribution for serverless function performance is determined to be concentrated, and otherwise, it is skewed. The results show that distributions of response latencies of more than 93% of the serverless functions are skewed both under cold starts and warm starts. We also try to determine the direction in which most response latencies of the serverless function are biased to the boxplot. We calculate the sum of distances of all response latencies of each serverless function from its maximum and minimum values, respectively. The results show that most response latencies of over 93% (cold starts: 69/72; warm starts: 67/72) of the serverless functions are skewed towards the corresponding minimum values, and another side of boxplots has a long tail. This can be observed visually in Figs. 8 and 9, where most boxplots are in the middle and lower part of the 0 to 1 interval.

Finding 2: The performance of serverless functions exhibits significant variance that cannot be ignored. Across multiple executions, the performance of serverless functions can vary by as much as 338.76%, with an average variance of 42.28%, indicating a large magnitude of the variance. Moreover, serverless function performance has different degrees of variance under cold and warm starts. The coefficient of variance values for serverless function performance under cold starts are from 1.23% to 31.18%, while those of warm starts are 0.86% to 42.81%. These values differ by as much as 49 times, showing a wide range of variance. Additionally, for 65.28% of the serverless functions, the response latency variance during warm starts is more pronounced than that observed during cold starts.

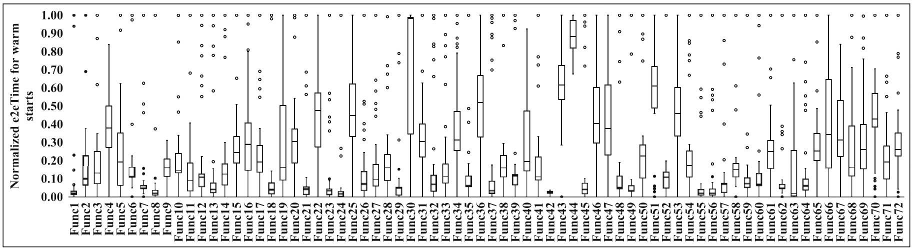

Fig. 9 (RQ2) The normalized boxplot for the warm-start response latencies of 72 serverless functions

6 RQ3: Reliability and Repetitions

In RQ3, we investigate the reliability of serverless function performance obtained at different repetitions. First, we use the response latencies of the serverless function at 50 repetitions as *standard result* of serverless function performance, since most of the collected papers that report repetitions use no more than 50 times, as summarized in Section 4. Then, we compare the serverless function performance obtained at **low repetitions** that are used in the collected papers and shown in Fig. 5, e.g., 3, 4, 5, 6, 10, and 20.

To facilitate performance analysis, we need to adopt appropriate statistical methods, either parametric or non-parametric, depending on the distribution of the performance results. For data that follows a normal distribution, parametric methods are suitable; for non-normal distributions, non-parametric methods are required. Thus, we first check whether serverless function performance follows a normal distribution. We adopt the Shapiro-Wilk test (Shapiro 1965) (abbreviated as W test), which is considered the most powerful normality test in most situations (https://pulmonarychronicles.com/index.php/pulmonarychronicles/article/view/805/1759). In the W test, its null hypothesis is that the performance results come from the population with the normal distribution. When running the W test, we can obtain a ρ value. At a ρ value greater than 0.05, we can accept the null hypothesis to indicate that serverless function performance follows a normal distribution; otherwise, we reject the null hypothesis and describe the serverless function performance as following a non-normal distribution. We apply normality checks to the response latencies of 50 runs of each serverless function. Figure 10 shows the CDF about all ρ values obtained from W tests. We observe that most ρ values are less than 0.05, i.e., rejecting the null hypothesis and presenting strong evidence for non-normality. Specifically, the response latencies of 88.89% (64/72) and 91.67% (66/72) of the functions follow a non-normal distribution in cold starts and warm starts, respectively. In this situation, non-parametric analysis methods, which do not assume normality, are more appropriate for the performance analysis of serverless functions, and they can also work for the normal distribution (Maricq et al. 2018). Thus, we use the most common metrics of interest in non-parametric analysis, e.g., median, tail percentile, and their confidence intervals (Maricq et al. 2018; Uta et al. 2020), to compare performance.

We calculate the median performance and tail performance (e.g., 90*th* percentile) of the serverless function at different low repetitions. Meanwhile, we calculate the corresponding 95% *confidence interval for the median* and *confidence interval for the 90th percentile* at 50 repetitions, as adopted by the previous work (Uta et al. 2020). The 95% *confidence interval for the median* or *confidence interval for the 90th percentile* represents the range in which

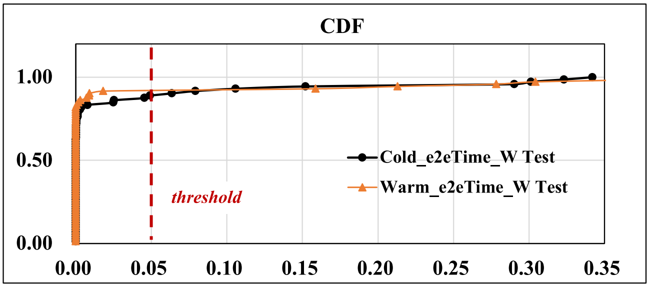

Fig. 10 (RQ3) The CDF of ρ values obtained from W tests (normality checks), which are applied to the response latencies of each serverless function

we could find the true median or 90*th* percentile with 95% probability if we could perform infinite repetitions. Thus, when a median or 90*th* percentile obtained at the low repetition lies outside the 95% *confidence interval for the median* or *confidence interval for the 90th percentile* obtained at the high repetition, it indicates that there is a 95% probability that this median or 90*th* percentile is inaccurate. The calculations of the *confidence interval for the median* and *confidence interval for the 90th percentile* can refer to the work (Le Boudec 2010; Maricq et al. 2018).

Result We observe that the serverless functions produce inaccurate median performance and tail performance under low repetitions. Specifically, Fig. 11 shows the percentage of the serverless functions in the case where the obtained median performance or 90*th* percentile performance is inaccurate under different low repetitions. When the number of repetitions is 3, for 56.94% (41/72) of the serverless functions in cold starts, the obtained median falls outside of the confidence intervals obtained at 50 repetitions, i.e., the obtained median performance is inaccurate, while warm starts have 59.72% (43/72) of the serverless functions. For the 90*th* percentile, 56.94% (41/72) of the serverless functions have inaccurate tail performance at 3 repetitions (warm start: 61.11% (44/72)). This indicates that experiments with low repetitions have a high risk of reporting unreliable performance results of serverless functions. We also observe that increasing the number of repetitions can make the percentage of inaccurate results decrease. However, at the 10-repetition most frequently used by the surveyed papers, 29.17% (21/72) of the serverless functions still show inaccurate median performance, and 40.28% (29/72) of serverless functions show inaccurate tail performance in cold starts. These results imply that serverless function performance is unreliable under low repetitions commonly used in most research papers, underscoring the significant consequences of neglecting the performance variance of serverless functions.

We try to find regularities in how to determine the appropriate repetitions for serverless functions to obtain reliable performance. We first observe the changes in top and bottom bounds for the confidence interval obtained at different repetitions. As an example, Fig. 12 shows the changes in the bounds of the *confidence intervals for the median*, with regard to *Func2*. We do not plot the *confidence intervals for the median* at smaller repetitions, e.g., 3, 4, 5, and 6 in Fig. 11, because these repetitions are also insufficient to calculate them (Landis and Koch 1977; Le Boudec 2010). Instead, we compare the number of repetitions from 10 to 50 in 5-step increments. From Fig. 12, confidence intervals gradually become tight as the number of repetitions increases. The line representing the difference between the top and bottom bounds also shows a gradual downward trend. However, when the number of

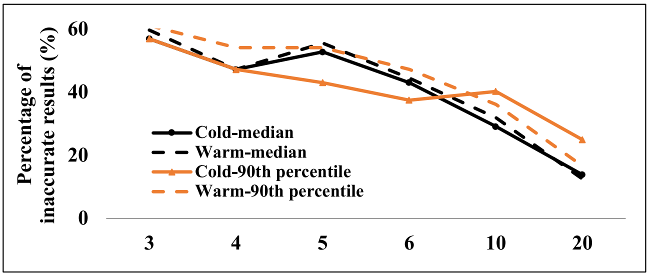

Fig. 11 (RQ3) The percentage of serverless functions, whose medians and 90*th* percentiles at low repetitions lie outside corresponding confidence intervals at 50 repetitions

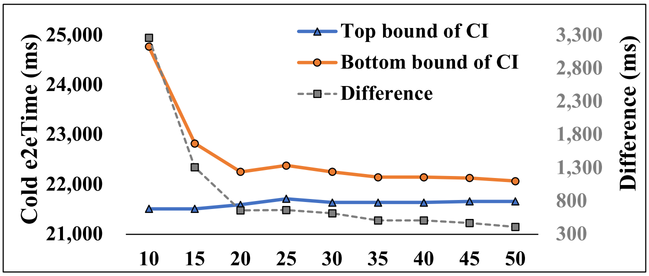

Fig. 12 (RQ3) Changes in the top and bottom bounds of the *confidence interval for the median* at different repetitions. An example is about *Func2*

repetitions is small, e.g., from 20 to 25 repetitions, the bounds of the confidence interval may have slight fluctuations. It is reasonable that the distribution with a small number of performance data may not be stable. Overall, performing more repetitions for functions may achieve a tight *confidence interval for the median*, which increases confidence in claiming the obtained results are close to the population distribution.

Leveraging the observation about tight confidence intervals for the median, we further explore how many repetitions may be required to achieve a sufficiently narrow confidence interval for serverless function performance. This interval is a desired interval representing satisfactory performance, where the empirical median differs from the observed true median by no more than the r error margin at a given confidence level, e.g., 95%. In other words, when the *confidence interval for the median* obtained from results at a certain repetition drops within the r error margin of the corresponding observed true median performance, the desired confidence interval is obtained to stop running the serverless function, and the current number of repetitions is regarded as the appropriate repetition for the serverless function. We show the results for $r = 0.5\%$ and 1% in Fig. 13. When $r = 0.5\%$, in cold starts, 98.61% (71/72) of the serverless functions require being executed at least 50 times to get the desired confidence interval, while warm starts have 81.94% (59/72) of the serverless functions. After we relax r to 1%, there are still 70.83% (51/72) serverless functions that need to be executed repeatedly at least 50 times in cold starts. There are 59.72% (43/72) serverless functions in warm starts. However, according to the aforementioned Fig. 5, only 28.95% of the research papers that report experiment repetitions execute serverless functions more than 50 times.

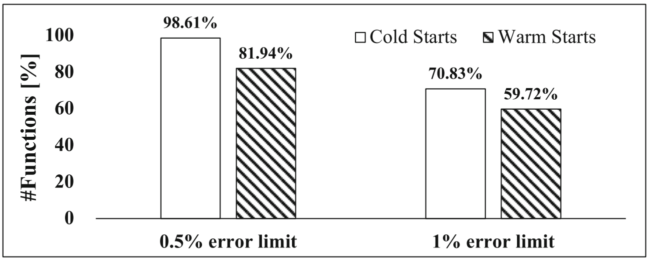

Fig. 13 (RQ3) The percentage of serverless functions that take *at least 50 times* to achieve a desired confidence interval about response latency, within 0.5% and 1% error demands

Overall, these results indicate that serverless functions require far more repetitions than the ones commonly used in most surveyed papers.

Finding 3: When a small number of repetitions, as commonly adopted in the collected papers, are executed, up to 61.11% of serverless functions exhibit unreliable performance results. This underscores the significant consequences of neglecting the performance variance of serverless functions. Additionally, up to 98.61% of serverless functions require a minimum of 50 executions to achieve the desired performance, but only 28.95% of the collected research papers, which report experiment repetitions, use repetition counts exceeding 50. These findings highlight the need for significantly more repetitions in serverless function experimentation than what is commonly employed in the existing literature.

7 Implications

Implications for researchers **1) Reproducibility.** The research community has increasingly emphasized the importance of reproducibility and replicability in studies (Anda et al. 2008; Cavezza et al. 2014; Demir et al. 2022; Rajiullah et al. 2019; Jueckstock et al. 2021; Amiri et al. 2022; Yu et al. 2023; Ardagna et al. 2018). Our study uncovers substantial performance variance in serverless functions, affecting research reproducibility and further deployment of cloud services. Insufficient documentation of performance measurement methods, including repetitions, hampers replication and result validation by other researchers. To enhance reproducibility in serverless computing, an open and explicit methodology is crucial. Researchers should clearly outline their performance measurement approach, including repetitions and rationale. **2) Reliability.** Our study highlights concerns about research reliability in serverless computing due to insufficient consideration of performance variance. Inadequate repetitions and justification can diminish result reliability, risking erroneous conclusions. This can lead to misguided decisions and resource wastage in academia and industry. To address this, researchers can provide well-justified repetitions and comprehensive results, e.g., median, coefficient of variance, percentiles, confidence intervals, and distribution. Serverless functions exhibit two types of starts: cold and warm starts, with more pronounced performance variance in warm starts. Researchers can tailor experimental designs, including repetitions, for each start type to capture specific characteristics effectively. **3) Tailored performance testing.** Our evaluation of RQ3 shows that while 50 repetitions are commonly used in the serverless computing community, they are insufficient for achieving the desired performance for 98.61% of serverless functions, due to significant performance variance, as shown in RQ2. In general, using more repetitions increases the accuracy and reliability of performance results by accounting for various potential impacts of the serverless platform. However, simply increasing repetitions for all serverless functions may not always be an effective solution because each function may require a different number of repetitions. This emphasizes the need for tailored performance testing in serverless computing, designed to ensure accurate and reliable performance measurement for serverless computing-based applications. Performance is a critical property of applications (Savasci et al. 2023; Zhang et al. 2021; Hounsel et al. 2020; Jayathilaka et al. 2017), and performance testing has been a standard procedure for acquiring reliable performance of these applications (He et al. 2021b, a). It involves iterative executions of the application-under-test with predefined inputs until a stopping criterion is met, thus mitigating the variance effect (He et al. 2019b, 2021b, a). Given the significant performance variance of serverless functions, there is research space to develop dedicated performance testing techniques and tailor the stopping criterion for this context. Furthermore, different from traditional cloud applications, serverless functions generally run for a short duration in milliseconds (Liu et al. 2023; Li et al. 2023). Thus, there is a need to use a finer performance accuracy analysis as the stopping criterion in serverless computing. Such approaches allow for efficient, targeted, and context-aware performance evaluation, regardless of when performance data for each function is captured or whether performance data experiences regular variances, thus avoiding the inefficiencies of simply increasing repetitions for all functions.

Implications for software developers **1) Mitigation strategies.** Given the substantial performance variance observed in serverless computing, it is essential for software developers to adopt strategies to mitigate this issue, especially when aiming for consistent user experiences. From our analysis of RQ2, the design of serverless function code may influence the magnitude of performance variance. Therefore, code optimization emerges as a potential mitigation strategy. This finding is consistent with various reports (https://appwrite.io/blog/post/serverless-functions-best-practices, https://www.serverless.com/guides/amazon-s3, https://www.serverlessguru.com/blog/aws-serverless-development-coding-best-practices) and exi sting studies (Liu et al. 2023; Taib et al. 2022), which highlight the influence of coding practices on serverless function performance. For instance, optimizing the usage of cloud storage services is crucial, as access patterns to cloud storage can affect performance (https://www.serverless.com/guides/amazon-s3). Furthermore, removing unused dependencies from serverless functions can reduce unnecessary overhead, improving performance consistency (https://www.serverless.com/guides/amazon-s3, Liu et al. (2023)). These examples underscore the potential of adopting efficient coding practices to mitigate performance variance in serverless functions. **2) Difference between start types.** In the development of serverless computing-based applications, software developers can carefully consider the distinct performance characteristics associated with both cold and warm starts. Our findings demonstrate that the response latency variance is more pronounced during warm starts than cold starts. To provide users with a seamless and consistent experience, software developers should ensure that these distinct characteristics do not impact the user experience under different start conditions.

Implications for cloud providers From our results, all the serverless platforms we study provide significant performance variance for serverless functions. Thus, cloud providers offering serverless environments should promise consistent quality of service when delivering the services by means of serverless computing. As highlighted in Finding 2, the performance of serverless functions can vary by as much as 338.76%, with an average variance of 42.28%. To ensure consistent quality of service, cloud providers could monitor performance changes of the serverless function as it is executed. If the magnitude of this change exceeds a specific threshold (e.g., average variance), cloud providers need to identify the function instance serving the current serverless function and conduct a detailed analysis of its resource usage to detect any abnormalities.

8 Threats to validity

Selection of relevant papers Our empirical study involves analyzing research papers related to serverless computing. Given the difficulty of collecting all such papers, we select a representative set for analysis. This selection process may pose a threat to the validity of our results. To mitigate this threat, we specifically gathered 110 research papers published in 77 top-tier conferences across different research communities. These top-tier conferences are known for including papers that have substantial impact and widespread recognition. The widespread recognition of these conferences underscores their status as trusted sources for analysis. This, in turn, enhances the representativeness of our data sources. However, we acknowledge that adding research papers from other journals or conferences would provide a more comprehensive analysis. In future work, we plan to expand more papers to further enhance the scope of our investigation.

Manual examination of papers In RQ1, we manually label the three types of reporting information of each collected paper. This may pose a potential threat to the validity of our summarized results. To minimize this threat, the first two authors independently read the full text of the papers to determine specific information. Then we calculate the inter-rater agreement during the labeling process, and the obtained agreement values indicate a perfect agreement level and reliable labeling procedures. Additionally, to resolve conflicts, an experienced arbitrator, who has ten years of cloud computing experience, is involved in discussing and reaching an agreement.

Root cause of performance variance In RQ2, we do not delve into the root cause of performance variance. The primary goal of our work is to raise awareness within the serverless computing community about the well-known performance variance problem in SE. Conducting a root cause analysis is beyond the scope of our study for several reasons. First, the serverless platforms commonly used by developers, such as AWS Lambda and Google Cloud Functions, are public and commercial services. While these platforms significantly reduce the management burden for developers, they also present challenges, as they operate as black boxes with opaque and uncontrollable policies. This lack of transparency makes it difficult to understand the underlying mechanisms that influence performance variance, complicating efforts to diagnose function performance issues. Second, the complexity of large-scale server management inherent to these serverless platforms further hinders the identification of specific root causes for performance variance. In future work, we plan to conduct a root cause analysis and connect the identified causes to the findings in literature, once serverless platforms provide more detailed information about their underlying runtime.

Source of spatial variability Our study investigates the magnitude of performance variance in serverless functions across different platforms. In the current experiment, we focused on maintaining consistency in the service regions of the platforms used to ensure comparison fairness among different platforms. We acknowledge that spatial variability could be an important source influencing performance. In future work, we plan to extend our study to explore the impact of executing serverless functions across different service regions on performance variance.

9 Related Work

Serverless computing The research community has exhibited an increasing interest in serverless computing, as evidenced by the growing body of literature in this field (Wen et al. 2023). A systematic review of serverless computing research (Wen et al. 2023) has been conducted, revealing a prominent focus on the performance of serverless computing in the existing literature.

Considerable research efforts have been dedicated to predicting and optimizing the performance of serverless computing. For performance prediction, previous studies (Eismann et al. 2021a; Lin and Khazaei 2020) have utilized historical data, e.g., memory size or resource consumption, to predict serverless function performance. For performance optimization, Liu et al. (2023) employed static program analysis techniques to optimize serverless function code and enhance its performance. Qi et al. (2022) presented a shared-memory framework and evaluated its performance improvement effectiveness on various serverless functions. For these studies, ignoring the performance variance problem can lead to inaccurate performance predictions and suboptimal optimization outcomes.

Additionally, there have been empirical studies that focus on characterizing the performance of serverless computing. For example, Wang et al. (2018) characterized serverless platform performance, examining scalability, cold start, etc. While they acknowledged the variability in latency during instance preparation of cold starts across multiple runs, they did not systematically investigate performance variance, as it was not the primary focus of their study. Wen et al. (2021b) performed a measurement study to evaluate the performance of commodity serverless platforms using different serverless functions. Similarly, Eismann et al. (2022) explored the stability of performance measurements on serverless platforms, specifically investigating various load or concurrency configurations. Different from these empirical studies, our objective is to examine the awareness of researchers about the performance variance problem and characterize the magnitude of this variance in serverless computing.

Performance variance Performance variance is a well-known problem in SE, attracting extensive research efforts for analysis. As cloud computing offers efficient resource management, it has gained widespread adoption. For instance, He et al. (2019a, 2021a) executed cloud applications hosted in a traditional cloud computing paradigm, Infrastructure-as-a-Service (Bhardwaj et al. 2010). They found that it is challenging to obtain accurate performance. Laaber et al. (2019) studied the impact of cloud environments on result variability. For other systems, Pham et al. (2020) quantified the variance of deep learning systems regarding models' accuracy, varying by up to 10.8%. Qian et al. (2021) explored the variance of fairness metrics in deep learning systems, identifying significant variance of up to 12.6%. Georges et al. (2007) delved into the performance variance of Java systems and highlighted the importance of statistically rigorous data analysis.

Although the significance of performance variance has been acknowledged, current serverless computing research still frequently overlooks this well-known problem. Moreover, the magnitude of performance variance in the new programming model - serverless functions, remains unclear. While it is true that many practitioners and researchers may have been apprehensive about the performance variance of serverless computing, there has not been solid scientific evidence of the magnitude of such an issue thus far. In this paper, we provide a comprehensive study to shed light on the oversight of performance variance in previous serverless computing research work and characterize the magnitude of this variance, emphasizing the significant consequences of disregarding this crucial aspect. Without this, all we are left with is a “belief” rather than quantitative scientific evidence.

10 Conclusion

We conducted an empirical study to highlight the lack of awareness in serverless computing research regarding the well-known performance variance problem in software engineering. To this end, we first collected and analyzed 99 research papers related to serverless computing performance, showing that attention to this performance variance is still in the infancy stage. Then, we conducted a measurement study to illustrate the substantial magnitude of performance variance in serverless computing. Specifically, we analyzed the end-to-end response latencies of 72 serverless functions collected from these papers obtained over multiple runs. We observed a significant performance variance, with a maximum variance of 338.76% (44.28% on average) among different runs. We further analyzed the reliability of serverless function performance obtained by executing a low number of repetitions, as commonly done in our collected papers. We found that 61.11% of the serverless functions have unreliable performance. This underscores the significant consequences of neglecting this critical aspect of performance variance. Moreover, 98.61% of the serverless functions required being executed at least 50 times to achieve reliable performance, but only 28.95% of the collected papers that report experiment repetitions execute serverless functions more than 50 times. This implies that serverless functions require far more repetitions than the ones commonly used in the literature.

Funding This work is supported by the National Natural Science Foundation of China under Grant No. 62032003.

Data Availability The detailed information for collected research papers and serverless functions are available in a public GitHub repository (https://github.com/JinfengWenWork/ServerlessPerformanceVariance). Moreover, we provide the deployment package and raw performance data of each serverless function, as well as code scripts used in our study.

Declarations

Conflicts of Interests/Competing Interests The authors declare that they have no conflict of interests and competing interests.

References

Adzic G, Chatley R (2017) Serverless computing: economic and architectural impact. In: Proceedings of the 11th Joint Meeting on Foundations of Software Engineering, pp 884–889

Akkus IE, Chen R, Rimac I, Stein M, Satzke K, Beck A, Aditya P, Hilt V (2018) Sand: Towards high-performance serverless computing. In: Proceedings of the 2018 USENIX Annual Technical Conference, pp 923–935

Amiri A, Zdun U, Van Hoorn A (2022) Modeling and empirical validation of reliability and performance trade-offs of dynamic routing in service-and cloud-based architectures. IEEE Trans Serv Comput 15(6):3372–3386

Anda BC, Sjøberg DI, Mockus A (2008) Variability and reproducibility in software engineering: A study of four companies that developed the same system. IEEE Trans Software Eng 35(3):407–429

Ardagna CA, Bellandi V, Bezzi M, Ceravolo P, Damiani E, Hebert C (2018) Model-based big data analytics-as-a-service: take big data to the next level. IEEE Trans Serv Comput 14(2):516–529

Bermbach D, Bader J, Hasenburg J, Pfandzelter T, Thamsen L (2022) Auctionwhisk: Using an auction-inspired approach for function placement in serverless fog platforms. Software: Pract Exp 52(5):1143–1169

Bhardwaj S, Jain L, Jain S (2010) Cloud computing: A study of infrastructure as a service (iaas). Int J Eng Inf Technol 2(1):60–63

Cavezza DG, Pietrantuono R, Alonso J, Russo S, Trivedi KS (2014) Reproducibility of environment-dependent software failures: An experience report. In: Proceedings of the 2014 IEEE 25th International Symposium on Software Reliability Engineering, pp 267–276. IEEE

Cohen J (1960) A coefficient of agreement for nominal scales. Educ Psych Measure 20(1):37–46

Copik M, Kwasniewski G, Besta M, Podstawski M, Hoefler T (2021) Sebs: A serverless benchmark suite for function-as-a-service computing. In: Proceedings of the 22nd International Middleware Conference, p 64–78

Demir N, Große-Kampmann M, Urban T, Wressnegger C, Holz T, Pohlmann N (2022) Reproducibility and replicability of web measurement studies. In: WWW ’22: The ACM Web Conference 2022, pp 533–544

Eismann S, Bui L., Grohmann J, Abad C, Herbst N, Kounev S (2021) Sizeless: Predicting the optimal size of serverless functions. In: Proceedings of the 22nd International Middleware Conference, pp 248–259

Eismann S, Costa DE, Liao L, Bezemer CP, Shang W, van Hoorn A, Kounev S (2022) A case study on the stability of performance tests for serverless applications. J Syst Software 189:111294

Eismann S, Scheuner J, Eyk EV, Schwinger M, Grohmann J, Herbst N, Abad C, Iosup A (2021) The state of serverless applications: collection, characterization, and community consensus. IEEE Trans Software Eng 48(10):4152–4166

Eskandani N, Salvaneschi G (2021) The wonderless dataset for serverless computing. In: Proceedings of the 2021 IEEE/ACM 18th International Conference on Mining Software Repositories, pp 565–569. IEEE

Fouladi S, Wahby RS, Shacklett B, Balasubramaniam KV, Zeng W, Bhalerao R, Sivaraman A, Porter G, Winstein K (2017) 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, pp 363–376

Fuerst A, Sharma P (2022) Locality-aware load-balancing for serverless clusters. In: Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing, pp 227–239

Georges A, Buytaert D, Eeckhout L (2007) Statistically rigorous java performance evaluation. ACM SIGPLAN Notice 42(10):57–76

Guizzo G, Sarro F, Harman M (2020) Cost measures matter for mutation testing study validity. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp 1127–1139

He S, Liu T, Lama P, Lee J, Kim IK, Wang W (2021) Performance testing for cloud computing with dependent data bootstrapping. In: Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering, pp 666–678

He S, Liu T, Lama P, Lee J, Kim IK, Wang W (2021) Performance testing for cloud computing with dependent data bootstrapping. In: Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021, pp 666–678

He S, Manns G, Saunders J, Wang W, Pollock L, Soffa ML (2019) A statistics-based performance testing methodology for cloud applications. In: Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp 188–199

He S, Manns G, Saunders J, Wang W, Pollock LL, Soffa ML (2019) A statistics-based performance testing methodology for cloud applications. In: Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, pp 188–199

Hounsel A, Borgolte K, Schmitt P, Holland J, Feamster N (2020) Comparing the effects of dns, dot, and doh on web performance. In: Proceedings of the Web Conference, pp 562–572

Jayathilaka H, Krintz C, Wolski R (2017) Performance monitoring and root cause analysis for cloud-hosted web applications. In: Proceedings of the International Conference on World Wide Web, pp 469–478

Jiang J, Gan S, Liu Y, Wang F, Alonso G, Klimovic A, Singla A, Wu W, Zhang C (2021) Towards demystifying serverless machine learning training. In: Proceedings of the 2021 International Conference on Management of Data, pp 857–871

Jonas E, Schleier-Smith J, Sreekanti V, Tsai CC, Khandelwal A, Pu Q, Shankar V, Carreira J, Krauth K, Yadwadkar N, Gonzalez JE, Popa RA, Stoica I, Patterson DA (2019) Cloud programming simplified: A berkeley view on serverless computing. arXiv:1902.03383

Jueckstock J, Sarker S, Snyder P, Beggs A, Papadopoulos P, Varvello M, Livshits B, Kapravelos A (2021) Towards realistic and reproducible web crawl measurements. In: Proceedings of the Web Conference, pp 80–91

Kim J, Lee K (2019) Functionbench: A suite of workloads for serverless cloud function service. In: Proceedings of the IEEE 12th International Conference on Cloud Computing, pp 502–504. IEEE

Klimovic A, Wang Y, Stuedi P, Trivedi A, Pfefferle J, Kozyrakis C (2018) Pocket: Elastic ephemeral storage for serverless analytics. In: Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation, pp 427–444

Laaber C, Scheuner J, Leitner P (2019) Software microbenchmarking in the cloud. how bad is it really? Empir Software Eng 24(4):2469–2508

Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. Biomet 33(1):159–174

Le Boudec JY (2010) Performance evaluation of computer and communication systems, vol 2. Epfl Press Lausanne

Lenarduzzi V, Panichella A (2020) Serverless testing: Tool vendors' and experts' points of view. IEEE Software 38(1):54–60

Li Y, Lin Y, Wang Y, Ye K, Xu CZ (2023) Serverless computing: state-of-the-art, challenges and opportunities. IEEE Trans Serv Comput 16(2):1522–1539

Lin C, Khazaei H (2020) Modeling and optimization of performance and cost of serverless applications. IEEE Trans Para Distrib Syst 32(3):615–632

Liu X, Wen J, Chen Z, Li D, Chen J, Liu Y, Wang H, Jin X (2023) Faaslight: general application-level cold-start latency optimization for function-as-a-service in serverless computing. ACM Trans Software Eng Method

Mahgoub A, Yi EB, Shankar K, Elnikety S, Chaterji S, Bagchi S (2022) Orion and the three rights: Sizing, bundling, and prewarming for serverless dags. In: Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation, pp 303–320

Maissen P, Felber P, Kropf P, Schiavoni V (2020) Faasdom: A benchmark suite for serverless computing. In: Proceedings of the 14th ACM International Conference on Distributed and Event-based Systems, pp 73–84

Maricq A, Duplyakin D, Jimenez I, Maltzahn C, Stutsman R, Ricci R (2018) Taming performance variability. In: Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation, pp 409–425

Müller I, Marroquín R, Alonso G (2020) Lambada: Interactive data analytics on cold data using serverless cloud infrastructure. In: Proceedings of the 2020 International Conference on Management of Data, pp 115–130

Papadopoulos AV, Versluis L, Bauer A, Herbst N, Von Kistowski J, Ali-Eldin A, Abad CL, Amaral JN, Tuma P, Iosup A (2019) Methodological principles for reproducible performance evaluation in cloud computing. IEEE Trans Software Eng 47(8):1528–1543

Patterson L, Pigorovsky D, Dempsey B, Lazarev N, Shah A, Steinhoff C, Bruno A, Hu J, Delimitrou C (2022) Hivemind: a hardware-software system stack for serverless edge swarms. In: Proceedings of the 49th Annual International Symposium on Computer Architecture, pp 800–816

Perron M, Castro Fernandez R, DeWitt D, Madden S (2020) Starling: A scalable query engine on cloud functions. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp 131–141

Pham HV, Qian S, Wang J, Lutellier T, Rosenthal J, Tan L, Yu Y, Nagappan N (2020) Problems and opportunities in training deep learning software systems: an analysis of variance. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, pp 771–783

Qi S, Monis L, Zeng Z, Wang Ic, Ramakrishnan K (2022) Spright: extracting the server from serverless computing! high-performance ebpf-based event-driven, shared-memory processing. In: Proceedings of the ACM SIGCOMM 2022 Conference, pp 780–794

Qian S, Pham VH, Lutellier T, Hu Z, Kim J, Tan L, Yu Y, Chen J, Shah S (2021) Are my deep learning systems fair? an empirical study of fixed-seed training. Adv Neural Inf Process Syst 34:30211–30227

Rajiullah M, Lutu A, Khatouni AS, Fida MR, Mellia M, Brunstrom A, Alay O, Alfredsson S, Mancuso V (2019) Web experience in mobile networks: Lessons from two million page visits. In: Proceedings of the World Wide Web Conference, pp 1532–1543

Ristov S, Pedratscher S, Wallnoefer J, Fahringer T (2020) Daf: Dependency-aware faasifier for node.js monolithic applications. IEEE Software 38(1):48–53

Savasci M, Ali-Eldin A, Eker J, Robertsson A, Shenoy P (2023) Ddpc: Automated data-driven power-performance controller design on-the-fly for latency-sensitive web services. In: Proceedings of the ACM Web Conference 2023, pp 3067–3076

Scheuner J, Leitner P (2020) Function-as-a-service performance evaluation: A multivocal literature review. J Syst Software 170:110708

Shahrad M, Fonseca R, Goiri Í, Chaudhry G, Batum P, Cooke J, Laureano E, Tresness C, Russinovich M, Bianchini R (2020) Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In: Proceedings of the 2020 USENIX Annual Technical Conference, pp 205–218

Shapiro SS, Wilk MB (1965) An analysis of variance test for normality (complete samples). Biomet 52(3/4):591–611

Singhvi A, Balasubramanian A, Houck K, Shaikh MD, Venkataraman S, Akella A (2021) Atoll: A scalable low-latency serverless platform. In: Proceedings of the ACM Symposium on Cloud Computing, pp 138–152

Taibi D, Kehoe B, Poccia D (2022) Serverless: from bad practices to good solutions. In: 2022 IEEE International Conference on Service-Oriented System Engineering, pp 85–92. IEEE

Taibi D, Spillner J, Wawruch K (2020) Serverless computing-where are we now, and where are we heading? IEEE Software 38(1):25–31

Uta A, Custura A, Duplyakin D, Jimenez I, Rellermeyer J, Maltzahn C, Ricci R, Iosup A (2020) Is big data performance reproducible in modern cloud networks? In: Proceedings of the 17th USENIX symposium on networked systems design and implementation, pp 513–527

Wang A, Chang S, Tian H, Wang H, Yang H, Li H, Du R, Cheng Y (2021) Faasnet: Scalable and fast provisioning of custom serverless container runtimes at alibaba cloud function compute. In: Proceedings of the 2021 USENIX Annual Technical Conference, pp 443–457

Wang L, Li M, Zhang Y, Ristenpart T, Swift M (2018) Peeking behind the curtains of serverless platforms. In: Proceedings of the 2018 USENIX Annual Technical Conference, pp 133–146

Weber M, Apel S, Siegmund N (2021) White-box performance-influence models: A profiling and learning approach. In: Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering, pp 1059–1071. IEEE

Wen J, Chen Z, Jin X, Liu X (2023) Rise of the planet of serverless computing: a systematic review. ACM Trans Software Eng Method 32(5):1–61

Wen J, Chen Z, Liu Y, Lou Y, Ma Y, Huang G, Jin X, Liu X (2021) An empirical study on challenges of application development in serverless computing. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp 416–428

Wen J, Liu Y, Chen Z, Chen J, Ma Y (2021) Characterizing commodity serverless computing platforms. J Software: Evol Process p e2394

Yu H, Wang H, Li J, Yuan X, Park SJ (2022) Accelerating serverless computing by harvesting idle resources. In: Proceedings of the ACM Web Conference, pp 1741–1751

Yu S, He P, Chen N, Wu Y (2023) Brain: Log parsing with bidirectional parallel tree. IEEE Trans Serv Comput 16(5):3224–3237

Yu T, Liu Q, Du D, Xia Y, Zang B, Lu Z, Yang P, Qin C, Chen H (2020) Characterizing serverless platforms with serverlessbench. In: Proceedings of the 2020 ACM Symposium on Cloud Computing, pp 30–44

Zhang J, Dong E, Meng Z, Yang Y, Xu M, Yang S, Zhang M, Yue Y (2021) Wisetrans: Adaptive transport protocol selection for mobile web service. In: Proceedings of the Web Conference, pp 284–294

Zhao L, Yang Y, Li Y, Zhou X, Li K (2021) Understanding, predicting and scheduling serverless workloads under partial interference. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp 1–15

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.