An Empirical Study on Challenges of Application Development in Serverless Computing

Jinfeng Wen*
Key Lab of High-Confidence Software Technology, MoE (Peking University)
Beijing, China
jinfeng.wen@stu.pku.edu.cn

Zhenpeng Chen*
Key Lab of High-Confidence Software Technology, MoE (Peking University)
Beijing, China
czp@pku.edu.cn

Yi Liu
Key Lab of High-Confidence Software Technology, MoE (Peking University)
Beijing, China
liuyi14@pku.edu.cn

Yiling Lou
Key Lab of High-Confidence Software Technology, MoE (Peking University)
Beijing, China
yiling.lou@pku.edu.cn

Yun Ma†
Institute for Artificial Intelligence, Peking University
Beijing, China
mayun@pku.edu.cn

Gang Huang
Key Lab of High-Confidence Software Technology, MoE (Peking University)
Beijing, China
hg@pku.edu.cn

Xin Jin
Key Lab of High-Confidence Software Technology, MoE (Peking University)
Beijing, China
xinjinpku@pku.edu.cn

Xuanzhe Liu†
Key Lab of High-Confidence Software Technology, MoE (Peking University)
Beijing, China
xzl@pku.edu.cn

ABSTRACT

Serverless computing is an emerging paradigm for cloud computing, gaining traction in a wide range of applications such as video processing and machine learning. This new paradigm allows developers to focus on the development of the logic of serverless computing based applications (abbreviated as *serverless-based applications*) in the granularity of function, thereby freeing developers from tedious and error-prone infrastructure management. Meanwhile, it also introduces new challenges on the design, implementation, and deployment of serverless-based applications, and current serverless computing platforms are far away from satisfactory. However, to the best of our knowledge, these challenges have not been well studied. To fill this knowledge gap, this paper presents the first comprehensive study on understanding the challenges in developing serverless-based applications from the developers' perspective. We mine and analyze 22,731 relevant questions from Stack Overflow (a popular Q&A website for developers), and show the increasing popularity trend and the high difficulty level of serverless computing for developers. Through manual inspection of 619 sampled questions, we construct a taxonomy of challenges that developers encounter, and report a series of findings and actionable implications. Stakeholders including application developers, researchers, and cloud providers can leverage these findings and implications to better understand and further explore the serverless computing paradigm.

CCS CONCEPTS

• **Software and its engineering** → **Software creation and management**; • **Computer systems organization** → **Cloud computing**; • **General and reference** → **Empirical studies**.

KEYWORDS

Serverless Computing, Application Development, Empirical Study, Stack Overflow

ACM Reference Format:
Jinfeng Wen, Zhenpeng Chen, Yi Liu, Yiling Lou, Yun Ma, Gang Huang, Xin Jin, and Xuanzhe Liu. 2021. An Empirical Study on Challenges of Application Development in Serverless Computing. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3468264.3468558

1 INTRODUCTION

Serverless computing is an emerging paradigm for cloud computing, gaining traction in a wide range of domains including video processing [79, 90], scientific computing [91, 104], machine learning [83, 89], big data processing [108], Internet of things [97], etc. It has been estimated that the market of serverless computing is expected to grow from USD 1.88 billion in 2016 to USD 7.72 billion by 2021 [64]. Moreover, it is predicted that 50% of global enterprises will employ serverless computing by 2025 [63].

The increasing popularity of serverless computing can be attributed to its benign characteristics. Specifically, it allows developers to focus on the application logic, where functions are composed

*Jinfeng Wen and Zhenpeng Chen made equal contributions to this work.
†Corresponding authors

ESEC/FSE '21, August 23–28, 2021, Athens, Greece

ACM ISBN 978-1-4503-8562-6/21/08…$15.00
https://doi.org/10.1145/3468264.3468558

and packaged as the format of services (i.e., Function-as-a-Service (*FaaS*) [77, 92, 96, 105, 106]), thereby freeing developers from tedious and error-prone infrastructure management like load-balancing, auto-scaling, fault tolerance, operational monitoring, etc. Moreover, it reduces the cost as developers pay for only the actual function executions. What is more, serverless computing also brings great benefits for cloud providers as it allows them to better utilize resources [92]. Therefore, major cloud providers have rolled out their serverless platforms, such as AWS Lambda [61], Microsoft Azure Functions [62], and Google Cloud Functions [65]. There are also increasing open-source implementations available for serverless computing like OpenWhisk [70] and OpenFaaS [69].

Given the popularity of serverless computing, there is a lot of room that can be explored by researchers and developers. Researchers have conducted a series of studies [96, 103, 107, 110] to compare the features, architectures, and performance properties, among different serverless platforms, in order to understand their differences and derive guidelines for choosing the most suitable platform for a given application scenario. Furthermore, since current serverless platforms have several shortcomings such as high startup time and limited communication bandwidth, recent work [77, 90, 99] has proposed new architectures and solutions to alleviate or eliminate them. Developers have also paid attention to the development of serverless computing based applications (in short of *serverless-based applications*). The specific programming challenges that developers encounter when developing serverless-based applications, e.g., the design, implementation, and deployment of serverless-based applications, configurations of related resources and environments, networking, etc., are frequently asked on developers' Q&A forums [1–7]. However, to the best of our knowledge, these challenges have not been well studied.

To bridge the knowledge gap, we perform the first comprehensive study to identify challenges in developing serverless-based applications from the developers' perspectives. Such a study is timely and valuable since it can aid developers in avoiding common pitfalls and make researchers and cloud providers better positioned to help developers develop serverless-based applications in a more targeted way. To this end, we mine and analyze the relevant questions from a variety of developers on Stack Overflow (SO), which is a popular Q&A forum for developers to seek advice from peers when they have programming issues [87]. Specifically, we collect 22,731 SO questions related to serverless computing to answer the following three research questions.

RQ1 (Popularity trend): *What is the popularity trend of serverless computing among developers?* Via quantitative analysis, we find that serverless computing is gaining increasing attention on Stack Overflow, demonstrating its rising popularity and the timeliness and the urgency of our study.

RQ2 (Difficulty level): *How difficult is serverless computing for developers?* To answer this question, we explore the difficulty of developers answering questions related to serverless computing. Results show that these questions are more difficult to answer than those related to other challenging topics in software engineering, which motivates us to further identify the specific challenges behind serverless computing.

RQ3 (Taxonomy of challenges): *What specific challenges do developers encounter when developing serverless-based applications?* To identify the challenges, we randomly sample 619 relevant SO questions for manual examination. For each question, we qualitatively extract the challenges behind it. Finally, we construct a taxonomy consisting of 36 categories, linked to challenges in developing serverless-based applications. It indicates that developers face a wide spectrum of challenges in serverless computing, covering conceptual questions, version control, programming language support, database connection, resource configuration, security, etc. Based on the taxonomy, we summarize a series of findings and actionable implications for developers, researchers, and cloud providers.

In addition, we offer the scripts and the dataset used in this study[1] as an additional contribution to the research community for other researchers to replicate and build upon.

2 BACKGROUND

Cloud computing has become a widely adopted paradigm for the delivery of computing services via the Internet. It has various forms, including Infrastructure as a Service (*IaaS*), Platform as a Service (*PaaS*), Software as a Service (*SaaS*), etc. They are built on top of each other and form a "cloud computing stack", on which each layer gradually relieves the tedious and error-prone infrastructure management, e.g., load balancing and auto-scaling. Serverless computing currently is the ultimate paradigm of cloud computing. In this paradigm, it introduces opportunities for software development. Specifically, developers can focus on their own application logic and save cost without the need for cloud computing expertise. The function becomes the basic unit in serverless computing. Developers write only the function with a specific event trigger (e.g., adding a file into the cloud storage), package and upload their code, and specify the required environment (e.g., language runtime, memory). Serverless platforms provide sandboxing environments (virtual machines or containers) to host functions, and automatically deal with scalability, fault tolerance, and other runtime issues. Different from traditional cloud computing, where developers often need to continuously manage and run servers and pay for them, such a programming paradigm abstracts away most operational tasks and simplifies the application development on the cloud. Meanwhile, its storage and computation scale separately, and are provisioned and priced independently.

However, this new programming paradigm changes the way of traditional software development and may introduce new challenges for developers. First, inherent limits of serverless computing prevent the migration from serverful applications [83, 88, 92]. For example, serverless computing lacks an efficient communication channel among functions, and has restricted resources (e.g., short execution time, confined memory size). Second, existing serverless platforms lack a unified programming framework and rich support tools, making it more difficult to develop serverless-based applications [92, 94]. For example, due to the lack of support tools, debugging and testing serverless-based applications are more challenging than serverful applications for developers. Last but not least, developers have to follow the programming model of serverless computing, with special attention to the design, implementation and deployment of serverless-based applications, configurations of resources and environments, processing of events, etc. [88, 94].

[1] https://github.com/WenJinfeng/FSE21-Dataset_Script

These challenges may affect the developers' practice, thereby reducing their enthusiasm for application development on the cloud. Thus, it is essential to understand the challenges in developing serverless-based applications, which can improve the development efficiency of developers, give actionable implications to researchers, and boost the usefulness of serverless platforms.

We briefly introduce the development process of serverless-based applications. Generally, to help developers develop better, serverless platforms provide local development tools, e.g., Command-Line Interfaces (CLIs), Software Development Kits (SDKs), etc. Moreover, some third-party tools like *Serverless Framework* [72], *Local Stack* [67] also emerge to allow developers to build and deploy their applications. Leveraging these development tools, the development process can be divided into five stages as shown in Figure 1.

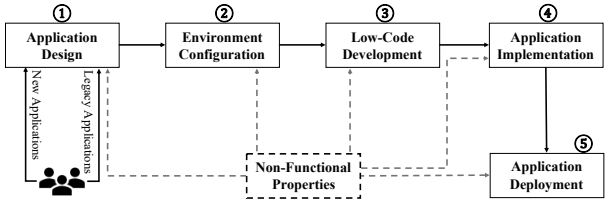

Figure 1: The development process of serverless-based applications.

Developers can have two cases for application development, i.e., (i) they develop new serverless-based applications from scratch, and (ii) they migrate legacy applications to serverless platforms to simplify the development or improve performance. For the first case, developers care about how to design the application functionality. For the second case, developers mainly take into account the migration design issues of existing legacy applications. They are called **Application Design**. After application design, developers configure the environment related to serverless computing to develop functions (called **Environment Configuration**). What is more, if functions require additional resources like compute and storage, developers also need to configure these resources and the corresponding permissions for functions leveraging the cloud console or a specific template. This configuration way can be viewed as **Low-Code Development** [68], which is performed through graphical user interfaces or configuration operation instead of traditional hand-coded programming. After accomplishing the required configurations, developers can focus on the code implementation of applications (called **Application Implementation**). Developers confirm the correctness of applications, then deploy applications to run on serverless platforms through the packaging way (called **Application Deployment**). Except for the above five stages, developers also need to consider **Non-Functional Properties**, e.g., performance, security and version control. These properties can be involved in one or more of the development stages. For instance, performance or security is considered in the low-code development and application implementation. Thus, non-functional properties are vital for the development of serverless-based applications.

3 METHODOLOGY

To understand the challenges that developers encounter when developing serverless-based applications, we follow previous studies [76, 78, 80, 95, 100, 109, 111] to collect and analyze the relevant questions posted on SO, where developers often seek technical assistance on unsolvable issues. In Figure 2, we show an overview of the methodology of our study.

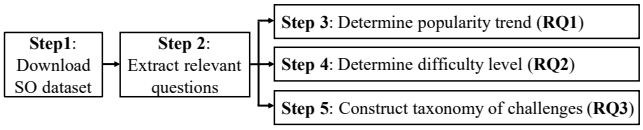

Figure 2: An overview of the methodology.

Step 1: Download SO dataset. To collect questions related to serverless computing, we first download the entire SO dataset S_{all} from the official Stack Exchange Data Dump [73] on December 9, 2020. The SO dataset includes 20,511,138 SO questions from July 31, 2008, to December 8, 2020. For each question, the metadata includes its identifier, creation date, body, title, answers, one to five tags that represent its topics, etc. In addition, the developer who posts a question can mark an answer as an accepted answer to indicate that it works for the question.

Step 2: Extract relevant questions. To collect serverless computing related questions from S_{all}, we follow previous work [80, 84, 95] to first construct a set of tags related to serverless computing and then extract questions tagged with at least one of these tags. The detailed procedures are as follows.

First, we use T_{ini} = {"serverless", "faas"} as our initial tag set, since this study focuses on serverless computing (also known as "*FaaS*" [77, 92, 96, 105, 106]).

Second, we extend the initial tag set by extracting all the tags of questions whose tags match at least one tag in T_{ini} (denote these questions as S_{cand}), and refine the extended tag set (denoted as T_{cand}) by keeping tags that are significantly relevant to serverless computing and excluding others. Specifically, we follow previous work [80, 109] to use two heuristics μ and υ to measure the significance and relevance of a tag t in T_{cand}.

$$(significance)\ \mu = \frac{\#\ of\ questions\ with\ tag\ t\ in\ S_{cand}}{\#\ of\ questions\ in\ S_{cand}} \quad (1)$$

$$(relevance)\ \upsilon = \frac{\#\ of\ questions\ with\ tag\ t\ in\ S_{cand}}{\#\ of\ questions\ with\ tag\ t\ in\ S_{all}} \quad (2)$$

A tag t is significantly relevant to serverless computing if its μ and υ are higher than the specific thresholds. To avoid omitting relevant tags, we employ the lowest thresholds used by previous studies [80, 95], and only the tags whose μ is higher than 0.005 and whose υ is higher than 0.05 are kept in T_{cand}.

Finally, the first two authors jointly examine each candidate tag in T_{cand} to filter out tags that are not related to serverless computing. As a result, the final tag set T_{final} consists of 13 tags, including "serverless", "faas", "serverless-framework", "aws-serverless",

"openwhisk", "aws-lambda", "aws-sam", "aws-sam-cli", "serverless-architecture", "serverless-offline", "vercel", "serverless-plugins" and "localstack". We extract 22,731 questions tagged with at least one of the 13 tags as the relevant questions (denoted as S_{rel}).

Step 3: Determine popularity trend (RQ1). To analyze the popularity trend of the topic of serverless computing, we follow previous work [76, 78, 80, 81, 84, 100, 109, 111] to calculate the number of questions and users related to serverless computing per year. Since the concept of serverless computing has attracted widespread attention through AWS Lambda from 2015 [92], the metrics are calculated based on S_{rel} for each of the past six years, i.e., from 2015 to 2020. Section 4 answers RQ1.

Step 4: Determine difficulty level (RQ2). To measure the difficulty level of answering questions related to serverless computing, we follow previous work [75, 76, 78, 80, 84, 100, 109, 111] to adopt two metrics, including the percentage of questions with no accepted answer ("*%no acc.*") and the median response time needed to receive an accepted answer ("*resp time*"). We calculate the two metrics based on S_{rel}. In addition, we calculate the average difficulty level of SO questions (i.e., *%no acc.* and *resp time* for S_{all}) as the baseline. For the first metric, we use the proportion test [98] to examine the statistical significance of comparison. Proportion test is used for testing the null hypothesis that the proportions in several groups are the same [71], and thus appropriate for the comparison in *%no acc.* [84]. For the second metric, we calculate the median value of response time to receive an accepted answer for both serverless-related questions (i.e., S_{rel}) and SO questions (i.e., S_{all}). Specifically, for each question, we extract the creation timestamps of this question and the corresponding accepted answer to calculate the time span between them as the response time. Section 5 answers RQ2.

Step 5: Construct taxonomy of challenges (RQ3). To analyze specific challenges that developers encounter and construct the taxonomy of challenges, we follow previous studies [85, 95] to use questions that have an accepted answer, ensuring that we consider only questions with a confirmed solution. As a result, 8,751 questions in S_{rel} are kept. Due to the large sample size, manually labeling all questions is time-consuming and infeasible. Following previous work [74], we randomly select a statistically significant sample ensuring a 99% confidence level ± 5% from 8,751 questions. As a result, 619 questions constitute the taxonomy dataset of our study, and this dataset size is larger than the one in existing work [82, 85, 112] that needs also manual analysis. Next, we illustrate the process of constructing the taxonomy of challenges.

First, we randomly sample 70% of 619 questions to construct the initial taxonomy of challenges. We adopt an open coding procedure [102] to analyze the sampled questions, in order to inductively create categories and subcategories of our taxonomy in a bottom-up way. The first two authors, both of whom have two years of cloud/serverless computing experience, jointly participate in the taxonomy construction. They read the sampled questions over and over again, in order to be acquainted with them. In this process, all the elements of each question, including the title, body, code snippets, comments, and even URLs contained in questioners and answerers, are taken into account for careful inspection.

The detailed procedure of open coding is as follows. Questions not related to serverless computing are labeled as *False positives*, and thus not included in our taxonomy. Regarding the remaining questions, the authors give short phrases to represent the challenges that developers encounter. Specifically, some questions are raised without any attempts, and they are mainly in the form of "how", e.g., "*How to use request module in node.js lambda*" [8]. For such questions, the authors often can clearly understand the challenges from the question description. Additionally, some questions describe the faults or unexpected results that developers are trying development practices. For such questions, the authors identify their causes as the challenges. For instance, when the developer posts an error message about the implementation of asynchronous functions [5], the authors find that the error is caused due to mixing asynchronous statements through checking the question descriptions, comments, and answers. Thus, the authors consider assigning asynchronous processing to be the challenge behind this question.

Second, the authors continue to group similar short phrases into categories and establish a hierarchical taxonomy of challenges. Regarding the grouping process, the authors iterate repeatedly between categories and questions. If questions are related to multiple categories, they are assigned to all related categories. In addition, if the authors have conflicts in labeling questions, a third arbitrator is introduced to discuss and resolve these conflicts. Particularly, the third arbitrator has seven years of cloud computing experiences. Through such a rigorous procedure, all questions come to an agreement and the final label results (i.e., the initial taxonomy) are confirmed by all the participants.

Finally, we perform reliability analysis and extended construction of the taxonomy of challenges. Based on the initial taxonomy, the remaining 30% questions are independently labeled by the first two authors to conduct reliability analysis. Each question is marked with *False positives* or the leaf categories of our taxonomy. Questions that cannot be classified into the current taxonomy are placed to a new category named *Pending*. We use Cohen's Kappa (κ) [86] to calculate the inter-rater agreement during the independent labeling. The value of the inter-rater agreement is 0.836, which indicates an almost perfect agreement [93] and reliable labeling procedure. Existing conflicts are then discussed and resolved by the first two authors and the third arbitrator. In addition, questions in *Pending* category are also identified with the help of the third arbitrator, and some new categories are added to our taxonomy. As a result, five new leaf categories are added and all questions in *Pending* can be assigned into our taxonomy.

To sum up, for the 619 sampled questions, 4 questions are marked as *False positives* and 17 questions are assigned into two categories. As a result, 632 samples are contained in the final taxonomy. Section 6 answers RQ3.

4 RQ1: POPULARITY TREND

Figure 3 represents the popularity trend of serverless computing in terms of the number of questions and users on SO. We find that this topic has been gaining increasing attention since 2015, demonstrating the timeliness and urgency of this study.

For the topic of serverless computing, questions and users increase in a steady trend as shown in Figure 3. Since Amazon first

rolled out AWS Lambda in November 2014[2], the concept of serverless computing has drawn widespread attention. We observe that both the number of questions and users related to serverless computing in 2016 increased by more than 340% compared to those of 2015. In addition, we calculate the annual growth rate of the SO dataset as the baseline. We find that the growth rate of SO questions and users from 2015 to 2020 ranges from 6% to 22%, while the growth rate of questions and users related to serverless computing ranges from 47% to 380%.

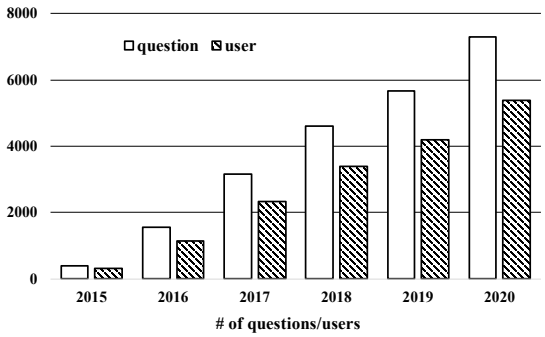

Figure 3: The popularity trend of serverless computing.

Finding 1: The topic of serverless computing is gaining increasing attention from developers, demonstrating the timeliness and urgency of this study.

5 RQ2: DIFFICULTY LEVEL

Table 1 shows metrics *%no acc.* and *resp time* of serverless-related questions and all SO questions. The values of *%no acc.* of serverless-related questions and all SO questions are 61.5% and 48.3%, respectively. The significance of this difference is ensured by the result of the proportion test ($\chi^2 = 1,577$, df = 1, p-value < 2.2e-16), indicating that questions related to serverless computing have a larger percentage of questions with no accepted answer. In addition, the values of *resp time* of serverless-related questions and all SO questions are 190 and 35 minutes, respectively. It implies that questions related to serverless computing need longer time to receive an accepted answer.

Based on the table, we can find that it is more challenging to answer questions related to serverless computing than other SE topics. In particular, 61.5% of questions related to serverless computing have no accepted answers, which is remarkably more frequent than other SE topics, such as Web development (i.e., 48.0% [78]), concurrency (i.e., 43.8% [76]), and mobile (i.e., 55.0% [100]). In addition, it often takes non-trival time to respond questions related to serverless computing (i.e., 190 min), which is substantially more time-consuming than other SE topics, such as Web development (i.e., 19 min [78]), concurrency(i.e., 42 min [76]), mobile (i.e., 55 min [100]). Since answers are often edited on SO, to make our results more reliable, we also calculate the response time as the time span between the creation time of a question and the last editing time of its accepted answer. The obtained result is 242 min, which is longer than our previous result (190 min). In summary, answering questions related to serverless computing needs longer time. Long answering time could sometimes be caused by lacking experts [78]. However, the response time to receive an accepted answer is still a well-adopted metric for difficulty, since lacking experts indicates that a large proportion of developers have not yet mastered serverless-related development skills, making these questions difficult to answer. Overall, our results demonstrate the necessity of identifying the challenges behind questions related to serverless computing.

[2]https://docs.aws.amazon.com/lambda/latest/dg/lambda-releases.html

Table 1: Difficulty level Comparison.

	SO	Web	concurrency	mobile	serverless
%no acc.	48.3%	48.0%	43.8%	55.0%	**61.5%**
resp time	35 min	19 min	42 min	55 min	**190 min**

Finding 2: Questions related to serverless computing are more challenging to answer than the average level of all SO questions and other topics related to software development tasks, e.g., Web development, concurrency, and mobile.

6 RQ3: TAXONOMY OF CHALLENGES

Figure 4 illustrates the hierarchical taxonomy of challenges in developing serverless-based applications. Nodes are in descending grey-level along with their depth in the hierarchy (e.g., leaf nodes are in white). Each leaf node represents a leaf category and its non-white parent node that consists of multiple categories is an *inner category*. For example, *Performance (G.6)* is an inner category that can be further divided into four leaf categories: *Execution Latency (G.6.1)*, *Resource Utilization (G.6.2)*, *Cold Startup (G.6.3)*, and *Throughput (G.6.4)*. The percentage for samples related to each category is in the parentheses. In total, our taxonomy consists of 11 inner categories and 36 leaf categories. We observe that when developing serverless-based applications, developers encounter problems in a broad spectrum of aspects, which indicates the diversity of challenges related to serverless computing. We next describe and exemplify each category by groups, and report our findings and implications for developers, researchers, and cloud providers.

Finding 3: The challenges that developers encounter have a broad spectrum of 36 leaf categories ranging from *Conceptual Questions* to *Version Control*, *Programming Language Support* to *Database Connection* and *Resource Configuration* to *Security*.

6.1 General Questions (A)

General questions represent the challenges involving no specific implementation details, which are often proposed by developers when they are looking for primary knowledge related to serverless computing. Our results show that 7.9% of challenges belong to general questions. We then discuss several categories in detail as follows.

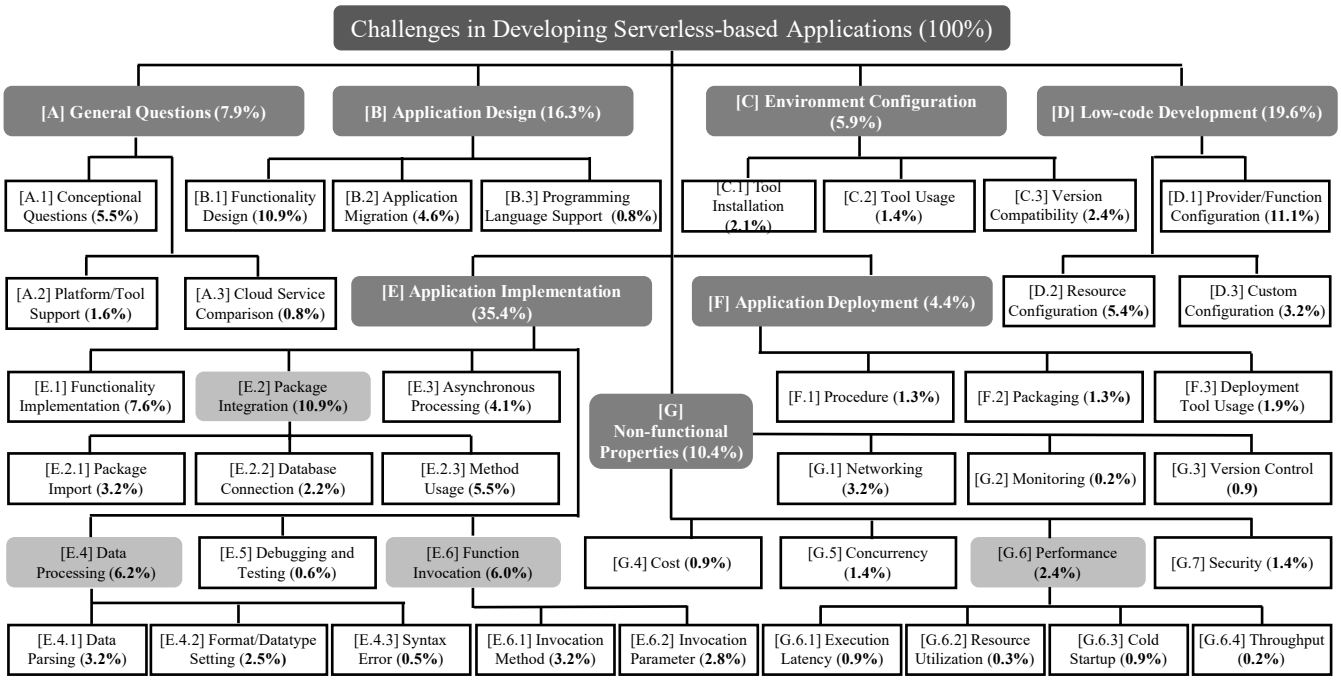

Figure 4: Taxonomy of challenges in developing serverless-based applications.

Conceptional Questions (A.1) This category represents questions about basic concepts or background knowledge of serverless computing, such as how AWS Lambda works [1] and the difference between functions in idle state and not [9]. Answerers mainly handle these questions by providing documentation-like information or translate the jargon-heavy documentation into case-specific guidance phrased in a developer-friendly way. In our taxonomy, 5.5% of the total challenges appear in conceptional questions. It illustrates that developers are eager to gain a comprehensive understanding of concepts related to serverless computing.

Platform/Tool Support (A.2) Developers wonder whether a certain serverless platform or development tool can satisfy their specific requirements. Specifically, for serverless platforms, developers may have concerns about hard restrictions (e.g., memory [10]) or task types (e.g., machine learning [11]). For development tools like *Serverless Framework* [72], developers often ask whether certain features, e.g., user authentication [12], are integrated with them.

Cloud Service Comparison (A.3) Developers may have challenges in understanding and choosing cloud services with similar functionality. For example, it is often confusing for developers to choose an appropriate database service in their functions, since there are often various database services, such as Amazon DynamoDB [58] and RDS [59].

Finding 4: 7.9% of the total challenges occur before actual practices and are classified in *General Questions*. Among the three categories, *Conceptual Questions* is the largest one, accounting for 70.0% of challenges in *General Questions*.

Discussion and implication: Our results show that developers encounter frequent challenges related to the basic concepts before actual practices. We further analyze the conceptual questions and find that there are mainly three aspects of understanding, i.e., underlying working mechanism, proprietary terms, and platform-specific restrictions. Developers need to spend non-negligible time on learning the required knowledge, especially proprietary terms and platform-specific restrictions, and think more about the working mechanism of serverless platforms. Questions of the underlying working mechanism are asked frequently in the way of *how it works*. Thus, cloud providers can supplement more organized documents to help developers search them and alleviate concerns.

With the popularity of machine learning and deep learning, more and more applications need to deal with data-intensive and computation-intensive tasks. However, due to inherent restrictions of serverless platforms (e.g., memory, storage, package size, and execution time), both data-processing applications with a large database and machine-learning applications with large libraries like TensorFlow cannot be handled normally. Thus, developers have to indirectly implement such serverless-based applications with other strategies that may increase the development complexity. Specifically, to avoid transmitting a large volume of data through the network, developers need to use other storage services to save query results [10]. They also need to pay attention to keep an efficient database connection to avoid exceeding the execution time limit. In addition, machine-learning applications may run for a long time with high memory demands, whose cost will skyrocket. The answerer suggests "*you will be better off with something like EC2 and ECS*" [11]. Thus, such computation-intensive applications may be more suitable to use the "traditional" *IaaS* paradigms such as virtual

machines or containers compared to serverless computing. Moreover, existing serverless platforms do not support GPU well, which can prevent fast-growing deep-learning tasks. If developers hope to enjoy advantages in simplifying the application development, cloud providers need to lift some restrictions of serverless platforms to support various workloads. Currently, serverless-based applications are most commonly used for running short-lived, transient, and even emergent tasks with low data volume and hardware capacity.

6.2 Application Design (B)

The challenges of application design account for 16.3% of the total challenges. On the one hand, developers need to develop a serverless-based application from scratch, so they ask questions about specific functionality design in a serverless way. On the other hand, developers may have legacy applications, and want to migrate them to serverless platforms, in order to simplify the development, improve performance, and save cost. Thus, migration-related questions are issued. In addition, it is inevitable to ask questions about programming language support in this stage. Detailed leaf categories are illustrated as follows.

Functionality Design (B.1) This category covers design-related challenges for implementing specific functionalities during the development of a new serverless-based application. They are often described with the key words "best approach", such as "*What is the best approach for generating thumbnails [...]?*" [2]. Answerers handle these questions by providing specific design steps or alternative options.

Application Migration (B.2) This category describes challenges in migrating from legacy applications to serverless-based applications. Developers wonder whether or how legacy applications can be migrated and run on serverless platforms, e.g., "*Is it possible to run Beautiful Soup [...] on AWS Lambda?*" [13] and "*How can run Rails web app in AWS Lambda server*" [14]. In addition, developers may need to migrate existing serverless-based applications due to the platform update or development tools update, so they may ask how to migrate integrated resources and services in *Serverless Framework* from version 0.5 to 1.0 in a safe manner [15].

Programming Language Support (B.3) This category covers challenges about the programming language support. They are often considered in the situation of constructing functions with different programming languages, e.g., "*AWS lambda with both python and java language support*" [16]. Developers also prefer to develop functions with the daily used language, e.g., creating *Openwhisk* actions with *Swift* [17].

> **Finding 5**: The third-largest category in proportion (i.e., 16.3%) covers the challenges appearing in the *Application Design* stage. The majority (67.0%) of these challenges are thrown with *Functionality Design*.

Discussion and implication: In the application design stage, developers frequently ask *Functionality Design*-related questions. It shows that developers look forward to obtaining the guidance of best practices or examples for their requirements. Cloud providers should provide more practical function templates and detailed instructions to help developers construct their functions more easily and quickly. We further observe that the most questions (about 90%) discuss the functionality design integrating other services that are provided by the same cloud providers. This is a common pattern for developing serverless-based applications, since integrating such services can simplify the development of serverless-based applications and improve their performance. However, it can be challenging for developers to design functions with such a large number of cloud services. Thus, both researchers and cloud providers can summarize a standard design specification to help developers better construct their functions with cloud services. For example, if cloud providers can maintain a function store, it may be convenient for developers to reuse existing functions to construct their serverless-based applications. In addition, it can further make the serverless-based application development more compositional in nature.

With the popularity of serverless computing, the trend of migrating legacy applications to serverless platforms may also increase. Thus, the design issues arising from application migration need to be solved urgently. Both researchers and cloud providers can pay more attention to the migration approach of legacy applications to guide developers in practice. For example, researchers can identify and refactor specific APIs with dynamic and static analysis. Cloud providers can further support more mostly-used frameworks (e.g., Web frameworks like *Spring*) so that developers directly run their legacy applications on serverless platforms.

With serverless computing, it is easier for developers to develop their applications with different languages (as known as hybrid programming). In a collaborative team, different developers can construct their functions with their own familiar languages, and different functions will be hosted in a dedicated instance and communicate with each other. However, not all languages are well supported. Cloud providers can further support more languages and optimize their run-time performance. Every coin has two sides. It is notable that hybrid programming can also increase the complexity of the serverless-based application development, especially for testing and debugging, which should be addressed by researchers with more advanced code analysis techniques.

6.3 Environment Configuration (C)

Developers install and configure local environments for further development. 5.9% of the total challenges are related to the environment configuration, and detailed leaf categories are explained as follows.

Tool Installation (C.1) This category represents challenges in installing development tools (e.g., AWS CLIs, Azure CLIs, and *Serverless Framework*) for serverless-based applications. When developers fail to install *Serverless Framework*, they may ask the question like "*Unable to install serverless framework on macosx*" [18]. Based on our observations from all answers and comments, we find that most installation problems are caused by incorrectly configuring system variables and missing specific dependencies.

Tool Usage (C.2) This category represents challenges in using related development tools. They often happen when the command parameters are used incorrectly. For example, developers fail to update the value of *EventSourceArn* of a Lambda function with an update command, which is caused by the incorrectly configured parameter "*EventSourceArn*" in the update command [19].

Version Compatibility (C.3) This category refers to challenges in dealing with certain versions of tools. Specifically, we find that commands may not run correctly due to the inherent bugs in certain versions [20] or the incorrect usage of package versions [3]. For example, developers ask that the command "project" is not found for *Serverless Framework* [3]. After checking answers, the solution is to use the 0.5.x version of *Serverless Framework* instead of the 1.x version, because the 1.x version implements the "services" command to replace the "project" command. Therefore, ensuring version compatibility is a prerequisite that cannot be ignored in the environment configuration.

> **Finding 6**: The challenges of development environment configuration (5.9%) include tool installation and usage, and version compatibility. *Version Compatibility* is dominating, accounting for 40.5% of the challenges in *Environment Configuration*.

Discussion and implication: In the environment configuration stage, how to install the appropriate version is the main concern for developers. Based on our observations, developers may use incorrect versions of tools, and endure inherent bugs of certain versions. The rapid evolution of development tools is making it challenging for developers to install the appropriate version, not only in the serverless-based application development, but also in many other software tasks, e.g., deep learning deployment [84] and software build [95]. Therefore, it is necessary for researchers to deliver more consistency detection techniques for version compatibility checking.

6.4 Low-code Development (D)

The significant feature of serverless computing is to represent infrastructure as code [66] that simplifies the management (i.e., creation and modification) of resources. Developers use the cloud console or a definition file written in JSON or YAML format to integrate with needed resources, which is known as *low-code development* [68]. This category covers challenges about low-code development and accounts for 19.6% of the total challenges. Detailed leaf categories are illustrated as follows.

Provider/Function Configuration (D.1) This category refers to challenges about configurations of cloud providers or functions. They often occur during the event and permission configurations. For the event configuration, the event source can be Amazon API Gateway [57], with which functions can be invoked once receiving an HTTPS request. In our study, we find that developers encounter request-related problems like "*defining the request mapping template in serverless.yml* " [21]. Additionally, the event source may come from other services (e.g., storage service like Amazon S3 [60]), which will be triggered when an object is added to a bucket on S3 [4]. Compared to developing a monolithic application, developers often need to specify the permissions of a function, such as the permission of accessing the network, and permission of accessing other services. If not correctly configured, the function will fail to access the target service due to access denied [22]. In addition, there are also some configuration challenges, such as choosing the appropriate region of cloud providers [23], specifying the function timeout [24], etc. Interestingly, we also find that even a small mistake in the definition file, e.g., wrong indentation [25] and typos of specific terms [26], can result in catastrophic consequences. However, there are few effective tools to deal with such urgent problems.

Resource Configuration (D.2) This category represents challenges about the configuration of integrated resources in serverless-based applications. Developers configure specifications (e.g., database name, table information, etc.) when integrating other cloud services (e.g., Amazon DynamoDB) [27]. Other functions can also be referenced as resources to compose a complicated serverless-based application [28].

Custom Configuration (D.3) In addition to the above configurations, developers may have personalized requirements, i.e., custom configuration. For example, developers try to specify a new response template with HTTP status to a redirect web page [29]. However, they find "*I can do it manually with AWS lambda user interface but I need to do it with serverless framework version v1.*". It also reflects the challenges in writing the configuration file compared to the convenient user interface provided by cloud providers.

> **Finding 7**: The challenges on *Low-Code Development* account for the second-largest percentage (i.e., 19.6%) of the total challenges. *Provider/Function Configuration* is the top category, accounting for 56.5% of the challenges in *Low-Code Development*.

Discussion and implication: Low-code development is a feature of developing serverless-based applications, effectively simplifying the integration of cloud services and management of functions. However, the related challenges even account for 19.6%, which is the second-largest category in proportion. Answerers mainly solve these problems by providing code snippets of specific configurations or modifying existing errors. It shows that developers lack a comprehensive configuration specification to guide them, and they are in urgent need of support tools. For cloud providers, they can provide more instructions and examples, especially the event and permission settings, to satisfy various configuration requirements. For framework developers, they can abstract the underlying differences of different serverless platforms, and provide a unified standard so that developers do not need to spend much time on browsing tedious documentations. For researchers, they can summarize the common issues of low-code developments, and extend existing code analysis tools and validation tools to help developers better find the bugs and mistakes of configurations.

6.5 Application Implementation (E)

Developers focus on the application logic without management underlying infrastructure in serverless computing. Any application code written by developers can be executed on serverless platforms. In our taxonomy, the challenges asked by developers most frequently are application implementation accounting for 35.4% of the total challenges. There are 3 inner categories (i.e., *E.2*, *E.4* and *E.6*) and 11 leaf categories. Detailed categories are explained as follows.

Functionality Implementation (E.1) This category refers to challenges in the implementation of specific functionalities. We find that there are two types in this category, including both serverless-unrelated and serverless-related functionality implementations. For example, a developer makes a mistake in reading a CSV file [30], and it is a serverless-unrelated functionality implementation problem.

Another example is a serverless-related functionality implementation like "*updating AWS S3 object metadata through boto3*" [31].

Package Integration (E.2) This category represents challenges encountered in using third-party libraries in functions. Specifically, when developers prepare to use the method of a certain library, they must import its package beforehand but may encounter import-related challenges (*E.2.1*). They are mainly in the import of external libraries [32], which is generally solved through modifying the package path. After importing required packages, especially when using database services, developers need to establish a connection in advance [33], and challenges on configuring and connecting such services (*E.2.2*) occur. Besides, developers may be incorrect to use methods of third-party libraries(*E.2.3*).

Asynchronous Processing (E.3) This category covers challenges in dealing with asynchronous activities in a function. Due to the event-driven nature of serverless computing, functions written in *JavaScript* often apply asynchronous callbacks with *async*, *await*, and *promise*. However, many developers fail to correctly use these features to construct their functions [34].

Data Processing (E.4) This category covers questions about data processing. Converting raw data into the required format(*E.4.1*) is an important factor so that functions can successfully resolve data from triggered events [35]. Meanwhile, formatting the result (*E.4.2*) is critical so that users can obtain the correct response [36]. There are also some incorrect data processing problems caused by syntax errors (*E.4.3*) [37].

Debugging and Testing (E.5) This category represents challenges on debugging and testing of functions. Most questions are about debugging and they are often described in the format of "*how to debug*" [38].

Function Invocation (E.6) Developers can incorporate multiple functions in a single serverless-based application, and these functions can be invoked internally. This category represents challenges in invoking functions with different invocation methods and parameters. Most functions are configured to be triggered by Amazon API Gateway, so most challenges are related to supporting various HTTP methods (*E.6.1*), e.g., "*POST*" and "*GET*" [39]. Other challenges are to transmit data parameters to functions (*E.6.2*), e.g., JSON-format payload [40] or binary data [41].

> **Finding 8**: Most (i.e. 35.4%) of the challenges occur in the *Application Implementation* stage, covering a wide spectrum of 11 leaf categories. *Package Integration* and *Functionality Implementation* are the top two categories, accounting for 30.8% and 21.4% of challenges in this stage, respectively.

Discussion and implication: Our results show that package integration is the most common problem in the application implementation stage. Especially, developers have to pay more attention to keeping the connection of database service in order to conform to the timeout limit of functions. Such problems may even not throw any exceptions [42], which are difficult to detect and debug for developers. For cloud providers, they can further improve their log/monitoring services (e.g., *Amazon CloudWatch*) to provide detailed logs and descriptions of error information, which can help developers better locate their bad code. For researchers, they can extend the existing analysis and diagnosis technologies with consideration of characteristics of serverless-based applications. Meanwhile, the unified programming model also needs to be studied to simplify developers' development issues on multiple different serverless platforms.

We also find that another big bottleneck is functionality implementation. In this category, the first two authors manually check these posts and find that serverless-unrelated and serverless-related functionality implementations account for 33.3% and 66.7%, respectively. It illustrates that the serverless computing paradigm increases the programming difficulty of developers, although reducing most operational tasks. On the one hand, cloud providers can further simplify the programming model for serverless-based applications. On the other hand, cloud providers can also provide mature development and debugging tools to assist developers' development. The challenge of debugging and testing tools is also found in the category (*E.5*) of our taxonomy. Traditional tools are not fully suitable for serverless-based applications, and new approaches are needed. Additionally, developers should avoid common pitfalls obtained from asynchronous processing and data processing, e.g., use of asynchronous statements, data parsing of the event, and format setting of the response, etc.

6.6 Application Deployment (F)

After implementing serverless-based applications, developers can deploy their applications to serverless platforms. Developers can package their functions and dependent libraries into a single bundle, and upload them to serverless platforms. Therefore, this category is deployment-related problems, accounting for 4.4% of the total challenges. Detailed leaf categories are explained as follows.

Procedure (F.1) This category describes questions about the procedure of a specific deployment, such as whether it is possible to deploy the same package to different cloud providers [43].

Packaging (F.2) This category represents challenges in packaging all code and dependent libraries of a serverless-based application. Developers often fail to package all related files in the correct format [44], and may also encounter the problem of exceeding the package size limit.

Deployment Tool Usage (F.3) This category refers to challenges in correctly using commands of deployment tools. For instance, the command "*sls resources deploy*" used by developers is incorrect [45], and instead the command "*sls deploy*" should be used.

> **Finding 9**: *Application Deployment* is the development stage where developers ask the least questions. It accounts for 4.4% of the total challenges, and contains *Procedure*, *Packaging*, and *Deployment Tool Usage* categories.

Discussion and implication: We further analyze *Procedure* and *Deployment Tool Usage* categories and find that developers have diverse deployment requirements, e.g., deploying with a specific version, deploying with an environment variable, or deploying the same package to multiple cloud providers, etc. For cloud providers, they can think about how to facilitate various deployment requirements. For example, cloud providers provide a custom deployment template to fill in the necessary and optional information (e.g., version and environment variable), and then apply the deployment tool

to complete the deployment tasks with this template. In addition, for the size limit of the deployment package, researchers can work on more analysis techniques to identify and remove unused code or files to reduce the package size of serverless-based applications.

6.7 Non-functional Properties (G)

Developers may consider some non-functional properties in the development of serverless-based applications. 10.4% of the total challenges are about non-functional properties. Specific categories are introduced as follows.

Networking (G.1) A function might integrate with external services (internal or publically hosted Web services, cloud services, databases, etc.) via the network. Thus, challenges related to networking occur, and they are mainly in Virtual Private Cloud (VPC) configuration. Some services can only be accessed by a function under the same VPC. For instance, Lambda functions connect to the Amazon RDS (i.e., a relational database) instances through the public Internet, but it fails due to a wrong VPC configuration [7].

Monitoring (G.2) This category represents questions about monitoring for serverless-based applications, e.g., whether all functions are completed normally [46].

Version Control (G.3) This category is mainly involved with the version management of serverless-based applications. Developers may be looking forward to an update or rollback version for applications [47]. Thus, serverless-based applications require version management like traditional applications using *Git*.

Cost (G.4) This category covers challenges about the billing model of serverless computing. Although serverless computing charges in a pay-as-you-go manner, there are still some concerns for developers. It is more difficult to predict the cost of a function, such as calculating the cost of the main function for a nested application when multiple functions are orchestrated in the main function [48].

Concurrency (G.5) This category represents questions about dealing with concurrent requests. They are often limited by the inherent concurrency strategy of different serverless platforms, such as concurrency number [49] and execution time [50].

Performance (G.6) This category represents questions about application performance. First, for execution latency (*G.6.1*), developers try to present a certain strategy to improve the execution efficiency of functions. For example, developers can speed up the date loading and save the network bandwidth by compressing data [51]. Second, for resource utilization (*G.6.2*), developers find competition between resources, such as I/O and CPU [52]. Third, for cold startup (*G.6.3*), when an invocation of a function occurs for the first time, this invocation is experiencing a *cold startup* and creating a function container. This container remains active and available for subsequent invocations for at least a few minutes before it is terminated. Developers often invoke a function periodically to reduce cold startup, i.e., "keeping warm" [53]. Finally, for throughput (*G.6.4*), developers look for ways to improve throughput, e.g., the read capacity of a database [54].

Security (G.7) Serverless computing hides infrastructure management from developers, but such opacity may increase concerns on security. They are mainly in the endpoint protection of functions [55] and data protection between functions [56]. Answerers mainly solve such questions by applying authentication and encryption on the endpoint and communication.

> **Finding 10**: 10.4% of the total challenges are in *Non-Functional Properties* covering 10 leaf categories. *Networking* and *Performance* (53.0% in total) are the top two categories in *Non-Functional Properties*.

Discussion and implication: Our results show that two biggest challenges are networking and performance in non-functional properties. Specifically, networking-related challenges are VPC settings. Developers can strengthen the networking knowledge, especially the VPC configuration process. For performance-related challenges, developers expect to improve performance from different perspectives, i.e., execution latency, resource utilization, cold startup, and throughput. Particularly, the performance of serverless platforms is a broad topic in academia. For example, some measurement work [96, 107, 110] has evaluated the performance of different serverless platforms from the execution latency of functions, resource utilization, throughput, etc. Moreover, some studies [77, 99, 103] have minimized cold startup to improve execution efficiency through system optimization or sandboxing design. By comparison, we find that academia and industry have similar focuses in terms of performance. In this situation, researchers can consider improving performance from the perspective of the developer practice, e.g., using data compression to reduce network overhead [51]. In addition, we find that developers ask fewer questions about *Performance* (2.4%) than *Application Implication* (35.4%). This observation coincides with the observation of other work [76], i.e., developers ask more about the correctness of their programs than performance.

Under the serverless computing paradigm, the version control and billing model still face new challenges, e.g., the version update and rollback of a project (or development environment or configuration), and cost prediction for function retry. Thus, to promote the use of serverless computing, researchers can design version control tools that conform to this new programming paradigm, and cloud providers try to provide a more user-friendly billing model to reduce the cost of non-real function execution.

7 THREATS TO VALIDITY

In this section, we discuss threats to the validity of our study.

Selection bias of data source. Similar to previous research [76, 78, 80, 95, 100, 109, 111], our work uses SO as the only data source to study the challenges that developers encounter, which may overlook useful insights from other data sources. Thus, we plan to expand our analysis to other data sources in future work, further verifying our findings. Nevertheless, because SO data contains posts from both novices and experts [112], we believe that our findings and implications are still valid.

Construction of tag set. We use a set of tags to extract SO questions related to serverless computing. However, since the chosen thresholds for significance and relevance metrics may omit some tags related to serverless computing, we cannot guarantee the tag set is complete. To mitigate this threat, we adopt the lowest thresholds used in previous work [80, 95] to include as many relevant

tags as possible. Furthermore, we refine the tag set through further manual inspection to ensure the precision.

Subjectivity of researchers. In this study, we adopt manual analysis to construct the taxonomy of challenges. The manual analysis may pose threats to the validity of our taxonomy. In order to minimize this threat, the first two authors and an experienced arbitrator discuss and reach an agreement for conflicts. Meanwhile, we calculate the inter-rater agreement as 0.836, which is relatively high and indicates the reliability of our labeling procedure.

Saturation of taxonomy. When classifying the remaining 30% of questions, we discover five new categories that are added to the final taxonomy. It may raise the uncertainty about the level of completeness of our taxonomy. However, the number of newly-added categories in this study is smaller than that in previous work [84], and these categories contain few questions (no more than five). Thus, our taxonomy is relatively complete. Meanwhile, the taxonomy is carefully checked by the third arbitrator with seven years of cloud computing experience.

8 RELATED WORK

In this section, we summarize related work of serverless computing as well as empirical studies on challenges that developers encounter in the widely used techniques.

Studies on serverless computing. Serverless computing is a new trending paradigm of cloud computing, and lots of related studies have been proposed. First, some work [96, 103, 107, 110] has evaluated different serverless platforms from various aspects. For example, Wang *et al.* [107] conducted a measurement study to characterize the architectures, performance, and resource efficiency for commercial serverless platforms. Mohanty *et al.* [96] compared the features and performance of open-source implementations of serverless computing. Second, some work [88, 94] has analyzed the characteristics of serverless-based applications to guide the design of approaches related to serverless computing. Furthermore, some literature reviews have been conducted to understand the gap related to serverless computing in academia and industry [101, 105]. Different from the previous work of serverless computing, our study focuses on the specific challenges that developers encounter in developing serverless-based applications.

Studies on developers' challenges. Many studies have investigated the software developers' perspectives by mining Q&A websites. Particularly, several empirical studies have leveraged SO data on different domains such as machine learning [78], deep learning deployment [84], mobile [100], security [109], big data [80], concurrency [76], etc. Specifically, Alshangiti *et al.* [78] conducted an empirical study of machine learning-related posts on SO to understand the challenges that developers face in developing machine learning applications. Chen *et al.* [84] presented a comprehensive study to explore what specific challenges developers face when deploying DL software. Ahmed *et al.* [76] understood the concurrent programming problem that developers often encounter when writing concurrent code. In similar, Bagherzadeh *et al.* [80] investigated interests and difficulties of big data developers by analyzing SO posts, while Rosen *et al.* [100] examined what mobile developers ask about. Considering the increasing popularity of serverless computing, in this work, we make the first attempt to investigate challenges that developers related to serverless computing may encounter.

9 CONCLUSION

Through analyzing SO questions related to serverless computing, we have found that the topic of serverless computing is gaining increasing attention among software engineers. Furthermore, we have demonstrated that questions related to serverless computing are more difficult to answer than other challenging topics in SE, e.g., Web development, concurrency, mobile, which motivates us to identify the specific challenges behind them. We have manually inspected 619 sampled questions related to serverless computing and constructed a comprehensive taxonomy, including 11 inner categories and 36 leaf categories, representing challenges that developers encounter in developing serverless-based applications. In addition, our study also provide a series of practical findings and actionable implications for developers, researchers, and cloud providers, intending to highlight good practices and interesting research avenues in adopting the serverless computing paradigm.

ACKNOWLEDGMENTS

This work was supported by the R&D Projects in Key Areas of Guangdong Province under the grant number 2020B010164002, the National Natural Science Foundation of China under the grant number 61725201, the Beijing Outstanding Young Scientist Program under the grant number BJJWZYJH01201910001004, and the PKU-Baidu Fund Project under the grant number 2020BD007.

REFERENCES

[1] [n.d.]. https://stackoverflow.com/questions/49172437/how-serverless-like-aws-lambda-and-google-cloud-function-work-on-infrastructure. Retrieved on February 16, 2021.
[2] [n.d.]. https://stackoverflow.com/questions/54760261/what-is-the-best-approach-for-generating-thumbnails-and-uploading-to-s3-bucket-i. Retrieved on February 16, 2021.
[3] [n.d.]. https://stackoverflow.com/questions/44678952/moonmail-installation-issue-command-project-not-found. Retrieved on February 16, 2021.
[4] [n.d.]. https://stackoverflow.com/questions/46765894/can-we-use-cloudwatch-events-on-s3-object-with-serverless. Retrieved on February 16, 2021.
[5] [n.d.]. https://stackoverflow.com/questions/57618689/how-do-i-use-aws-secret-manager-with-nodejs-lambda. Retrieved on February 16, 2021.
[6] [n.d.]. https://stackoverflow.com/questions/54233577/way-to-include-exlude-directories-controlling-aws-lambda-size-serverless. Retrieved on February 16, 2021.
[7] [n.d.]. https://stackoverflow.com/questions/55019736/aws-lambda-in-vpc-with-rds-and-internet-connection. Retrieved on February 16, 2021.
[8] [n.d.]. https://stackoverflow.com/questions/58372902/how-to-use-request-module-in-node-js-lambda. Retrieved on February 16, 2021.
[9] [n.d.]. https://stackoverflow.com/questions/62005562/aws-lambda-the-function-is-idle. Retrieved on February 16, 2021.
[10] [n.d.]. https://stackoverflow.com/questions/55017141/aws-lambda-extract-large-data-and-upload-to-s3. Retrieved on February 16, 2021.
[11] [n.d.]. https://stackoverflow.com/questions/57125277/bring-machine-learning-to-live-production-with-aws-lambda-function. Retrieved on February 16, 2021.
[12] [n.d.]. https://stackoverflow.com/questions/41664708/cognito-user-pool-authorizer-with-serverless-framework. Retrieved on February 16, 2021.
[13] [n.d.]. https://stackoverflow.com/questions/45692168/web-scraping-with-aws-lambda. Retrieved on February 16, 2021.
[14] [n.d.]. https://stackoverflow.com/questions/51095849/how-can-run-rails-web-app-in-aws-lambda-server. Retrieved on February 16, 2021.
[15] [n.d.]. https://stackoverflow.com/questions/44842070/aws-serverless-resources-deploy-from-v0-5-to-v1-0. Retrieved on February 16, 2021.
[16] [n.d.]. https://stackoverflow.com/questions/61548553/aws-lambda-with-both-python-and-java-language-support. Retrieved on February 16, 2021.
[17] [n.d.]. https://stackoverflow.com/questions/45308893/sharing-code-across-swift-openwhisk-actions. Retrieved on February 16, 2021.

[18] [n.d.]. https://stackoverflow.com/questions/43520629/unable-to-install-serverless-framework-on-macosx. Retrieved on February 16, 2021.
[19] [n.d.]. https://stackoverflow.com/questions/60808256/update-eventsourcearn-through-aws-cli. Retrieved on February 16, 2021.
[20] [n.d.]. https://stackoverflow.com/questions/51925672/sls-dynamodb-start-throws-spawn-java-enoent. Retrieved on February 16, 2021.
[21] [n.d.]. https://stackoverflow.com/questions/40569505/proper-request-template-mapping-or-process-in-order-to-upload-a-photo-to-s3-usin. Retrieved on February 16, 2021.
[22] [n.d.]. https://stackoverflow.com/questions/47249256/accessdenied-error-message-when-calling-aws-s3-buckets-from-serverless-lambda-fu. Retrieved on February 16, 2021.
[23] [n.d.]. https://stackoverflow.com/questions/40640433/what-is-the-syntax-in-serverless-yml-file-to-deploy-lambda-to-multiple-regions. Retrieved on February 16, 2021.
[24] [n.d.]. https://stackoverflow.com/questions/58504542/how-to-fix-timed-out-error-using-python-in-aws-lambda-functions-when-i-am-using. Retrieved on February 16, 2021.
[25] [n.d.]. https://stackoverflow.com/questions/58224566/serverless-framework-ignoring-authorizer-block-in-lambda-proxy-setup. Retrieved on February 16, 2021.
[26] [n.d.]. https://stackoverflow.com/questions/54581575/conditional-resource-in-serverless. Retrieved on February 16, 2021.
[27] [n.d.]. https://stackoverflow.com/questions/47327765/creating-two-dynamodb-tables-in-serverless-yml. Retrieved on February 16, 2021.
[28] [n.d.]. https://stackoverflow.com/questions/44032664/reference-function-from-within-serverless-yml. Retrieved on February 16, 2021.
[29] [n.d.]. https://stackoverflow.com/questions/39793242/serverless-response-template-with-status-code. Retrieved on February 16, 2021.
[30] [n.d.]. https://stackoverflow.com/questions/56849240/how-to-read-csv-file-from-s3-bucket-in-aws-lambda. Retrieved on February 16, 2021.
[31] [n.d.]. https://stackoverflow.com/questions/64998595/signature-error-while-updating-s3-object-metadata-through-boto3. Retrieved on February 16, 2021.
[32] [n.d.]. https://stackoverflow.com/questions/37169377/serverless-framework-how-to-add-external-npm-packages. Retrieved on February 16, 2021.
[33] [n.d.]. https://stackoverflow.com/questions/35969178/serverless-framework-with-node-mysql. Retrieved on February 16, 2021.
[34] [n.d.]. https://stackoverflow.com/questions/60506343/why-would-this-aws-lambda-cause-error-warning-callback-response-already-delive. Retrieved on February 16, 2021.
[35] [n.d.]. https://stackoverflow.com/questions/56712973/passing-a-json-file-in-event-event-cannot-read-keys-only-values. Retrieved on February 16, 2021.
[36] [n.d.]. https://stackoverflow.com/questions/49328315/how-to-send-back-non-stringified-data-in-serverles-aws. Retrieved on February 16, 2021.
[37] [n.d.]. https://stackoverflow.com/questions/63476921/lambda-function-in-python-returning-configuration-error. Retrieved on February 16, 2021.
[38] [n.d.]. https://stackoverflow.com/questions/45668631/debugging-aws-serverless-lambda-functions-with-dynamodbevents-in-c-sharp. Retrieved on February 16, 2021.
[39] [n.d.]. https://stackoverflow.com/questions/53626962/what-is-the-correct-way-to-get-the-origin-header-in-a-serverless-deployed-lambda. Retrieved on February 16, 2021.
[40] [n.d.]. https://stackoverflow.com/questions/60381331/aws-lambda-function-cloudwatch-how-to-pass-the-right-parameters-to-a-timed-ex. Retrieved on February 16, 2021.
[41] [n.d.]. https://stackoverflow.com/questions/51850931/openwhisk-and-binary-data-from-google-flatbuffers. Retrieved on February 16, 2021.
[42] [n.d.]. https://stackoverflow.com/questions/52465530/sequelize-connection-timeout-while-using-serverless-aurora-looking-for-a-way-to. Retrieved on February 16, 2021.
[43] [n.d.]. https://stackoverflow.com/questions/49045256/serverless-deploying-to-aws-azure-or-gcp. Retrieved on February 16, 2021.
[44] [n.d.]. https://stackoverflow.com/questions/56575448/why-i-missing-dependencies-in-aws-lambda-when-deploy-packages-in-python. Retrieved on February 16, 2021.
[45] [n.d.]. https://stackoverflow.com/questions/48315409/serverless-command-resources-not-found. Retrieved on February 16, 2021.
[46] [n.d.]. https://stackoverflow.com/questions/63002322/how-to-monitor-if-all-aws-lambda-functions-executions-finish-correctly. Retrieved on February 16, 2021.
[47] [n.d.]. https://stackoverflow.com/questions/51005379/how-do-you-manage-updates-rollbacks-and-multiples-versions-with-appsync-and-serv. Retrieved on February 16, 2021.
[48] [n.d.]. https://stackoverflow.com/questions/51854491/running-a-graphql-app-on-aws-lambda. Retrieved on February 16, 2021.
[49] [n.d.]. https://stackoverflow.com/questions/52210036/openwhisk-increase-number-of-concurrent-requests. Retrieved on February 16, 2021.
[50] [n.d.]. https://stackoverflow.com/questions/52280344/openwhisk-request-has-not-yet-finished. Retrieved on February 16, 2021.
[51] [n.d.]. https://stackoverflow.com/questions/48309760/how-to-send-gzipped-json-response-from-google-cloud-functions. Retrieved on February 16, 2021.
[52] [n.d.]. https://stackoverflow.com/questions/47373834/ffmpeg-azure-function-consumption-plan-low-cpu-availability-for-high-volume-requ. Retrieved on February 16, 2021.
[53] [n.d.]. https://stackoverflow.com/questions/54895958/how-can-i-keep-warm-an-aws-lambda-invoked-from-api-gateway-with-proxy-integratio. Retrieved on February 16, 2021.
[54] [n.d.]. https://stackoverflow.com/questions/51703030/determine-read-capacity-unit-for-dynamodb-table. Retrieved on February 16, 2021.
[55] [n.d.]. https://stackoverflow.com/questions/33001798/how-to-protect-serverless-framework-endpoints-from-abuse-dos. Retrieved on February 16, 2021.
[56] [n.d.]. https://stackoverflow.com/questions/48206257/protect-data-in-transit-when-two-lambda-communicate-via-aws-sns-simple-notifica. Retrieved on February 16, 2021.
[57] [n.d.]. Amazon API Gateway. https://aws.amazon.com/api-gateway/. Retrieved on February 16, 2021.
[58] [n.d.]. Amazon DynamoDB. https://aws.amazon.com/dynamodb/?nc1=h_ls. Retrieved on February 16, 2021.
[59] [n.d.]. Amazon RDS. https://aws.amazon.com/rds/. Retrieved on February 16, 2021.
[60] [n.d.]. Amazon S3. https://aws.amazon.com/s3/. Retrieved on February 16, 2021.
[61] [n.d.]. AWS Lambda. https://docs.aws.amazon.com/lambda/latest/dg/welcome.html. Retrieved on February 16, 2021.
[62] [n.d.]. Azure Functions. https://docs.microsoft.com/en-us/azure/azure-functions/. Retrieved on February 16, 2021.
[63] [n.d.]. The CIO's guide to serverless computing. https://www.gartner.com/smarterwithgartner/the-cios-guide-to-serverless-computing/. Retrieved on February 16, 2020.
[64] [n.d.]. Function-as-a-Service market. https://www.marketsandmarkets.com/Market-Reports/function-as-a-service-market-127202409.html. Retrieved on February 16, 2021.
[65] [n.d.]. Google Cloud Functions. https://cloud.google.com/functions. Retrieved on February 16, 2021.
[66] [n.d.]. Infrastructure as code. https://en.wikipedia.org/wiki/Infrastructure_as_code. Retrieved on February 16, 2021.
[67] [n.d.]. LocalStack: a fully functional local AWS cloud stack. https://github.com/localstack/localstack. Retrieved on February 16, 2020.
[68] [n.d.]. Low-code development. https://en.wikipedia.org/wiki/Low-code_development_platform. Retrieved on February 16, 2021.
[69] [n.d.]. OpenFaaS. https://www.openfaas.com/. Retrieved on February 16, 2021.
[70] [n.d.]. Openwhisk. https://openwhisk.apache.org/. Retrieved on February 16, 2021.
[71] [n.d.]. R documentation. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/prop.test. Retrieved on February 16, 2020.
[72] [n.d.]. Serverless Framework. https://www.serverless.com/. Retrieved on February 16, 2021.
[73] [n.d.]. Stack Exchange Data Dump. https://archive.org/details/stackexchange. Retrieved on February 16, 2021.
[74] Emad Aghajani, Csaba Nagy, Olga Lucero Vega-Márquez, Mario Linares-Vásquez, Laura Moreno, Gabriele Bavota, and Michele Lanza. 2019. Software documentation issues unveiled. In *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019*. 1199–1210. https://doi.org/10.1109/icse.2019.00122
[75] Md Ahasanuzzaman, Muhammad Asaduzzaman, Chanchal K Roy, and Kevin A Schneider. 2018. Classifying stack overflow posts on API issues. In *Proceedings of 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering, SANER 2018*. 244–254. https://doi.org/10.1109/saner.2018.8330213
[76] Syed Ahmed and Mehdi Bagherzadeh. 2018. What do concurrency developers ask about? A large-scale study using Stack Overflow. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2018*. 30:1–30:10. https://doi.org/10.1145/3239235.3239524
[77] Istemi Ekin Akkus, Ruichuan Chen, Ivica Rimac, Manuel Stein, Klaus Satzke, Andre Beck, Paarijaat Aditya, and Volker Hilt. 2018. SAND: Towards high-performance serverless computing. In *Proceedings of the 2018 USENIX Annual Technical Conference, ATC 2018*. 923–935.
[78] Moayad Alshangiti, Hitesh Sapkota, Pradeep K. Murukannaiah, Xumin Liu, and Qi Yu. 2019. Why is developing machine learning applications challenging? A study on Stack Overflow posts. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement, ESEM 2019*. 1–11. https://doi.org/10.1109/esem.2019.8870187
[79] Lixiang Ao, Liz Izhikevich, Geoffrey M Voelker, and George Porter. 2018. Sprocket: A serverless video processing framework. In *Proceedings of the 2018 ACM Symposium on Cloud Computing, SoCC 2018*. 263–274. https://doi.org/10.1145/3267809.3267815

[80] Mehdi Bagherzadeh and Raffi Khatchadourian. 2019. Going big: A large-scale study on what big data developers ask. In *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2019*. 432–442. https://doi.org/10.1145/3338906.3338939

[81] Anton Barua, Stephen W Thomas, and Ahmed E Hassan. 2014. What are developers talking about? An analysis of topics and trends in Stack Overflow. *Empirical Software Engineering* 19, 3 (2014), 619–654. https://doi.org/10.1007/s10664-012-9231-y

[82] Stefanie Beyer, Christian Macho, Massimiliano Di Penta, and Martin Pinzger. 2018. Automatically classifying posts into question categories on Stack Overflow. In *Proceedings of the 2018 IEEE/ACM 26th International Conference on Program Comprehension, ICPC 2018*. 211–21110. https://doi.org/10.1145/3196321.3196333

[83] Joao Carreira, Pedro Fonseca, Alexey Tumanov, Andrew Zhang, and Randy Katz. 2019. Cirrus: A serverless framework for end-to-end ML workflows. In *Proceedings of the ACM Symposium on Cloud Computing, SoCC 2019*. 13–24. https://doi.org/10.1145/3357223.3362711

[84] Zhenpeng Chen, Yanbin Cao, Yuanqiang Liu, Haoyu Wang, Tao Xie, and Xuanzhe Liu. 2020. A comprehensive study on challenges in deploying deep learning based software. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2020*. 750–762. https://doi.org/10.1145/3368089.3409759

[85] Zhenpeng Chen, Huihan Yao, Yiling Lou, Yanbin Cao, Yuanqiang Liu, Haoyu Wang, and Xuanzhe Liu. 2021. An empirical study on deployment faults of deep learning based mobile applications. In *Proceedings of the 43rd International Conference on Software Engineering, ICSE 2021*. 674–685. https://doi.org/10.1109/icse43902.2021.00068

[86] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20, 1 (1960), 37–46. https://doi.org/10.1177/001316446002000104

[87] Alex Cummaudo, Rajesh Vasa, Scott Barnett, John Grundy, and Mohamed Abdelrazek. 2020. Interpreting cloud computer vision pain-points: A mining study of Stack Overflow. In *Proceedings of the 41st International Conference on Software Engineering, ICSE 2020*. 1584–1596. https://doi.org/10.1145/3377811.3380404

[88] Simon Eismann, Joel Scheuner, Erwin van Eyk, Maximilian Schwinger, Johannes Grohmann, Nikolas Herbst, Cristina L Abad, and Alexandru Iosup. 2020. A review of serverless use cases and their characteristics. *arXiv preprint arXiv:2008.11110* (2020).

[89] Lang Feng, Prabhakar Kudva, Dilma Da Silva, and Jiang Hu. 2018. Exploring serverless computing for neural network training. In *Proceedings of the IEEE 11th international conference on cloud computing, CLOUD 2018*. 334–341. https://doi.org/10.1109/cloud.2018.00049

[90] Sadjad Fouladi, Riad S Wahby, Brennan Shacklett, Karthikeyan Vasuki Balasubramaniam, William Zeng, Rahul Bhalerao, Anirudh Sivaraman, George Porter, and Keith Winstein. 2017. Encoding, fast and slow: Low-latency video processing using thousands of tiny threads. In *Proceedings of the 2017 USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017*. 363–376.

[91] Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht. 2017. Occupy the cloud: Distributed computing for the 99%. In *Proceedings of the ACM Symposium on Cloud Computing, SoCC 2017*. 445–451. https://doi.org/10.1145/3127479.3128601

[92] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, Joseph E. Gonzalez, Raluca Ada Popa, Ion Stoica, and David A. Patterson. 2019. Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383* (2019).

[93] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics* 33, 1 (1977), 159–174. https://doi.org/10.2307/2529310

[94] Philipp Leitner, Erik Wittern, Josef Spillner, and Waldemar Hummer. 2019. A mixed-method empirical study of Function-as-a-Service software development in industrial practice. *Journal of Systems and Software* 149 (2019), 340–359. https://doi.org/10.1016/j.jss.2018.12.013

[95] Yiling Lou, Zhenpeng Chen, Yanbin Cao, Dan Hao, and Lu Zhang. 2020. Understanding build issue resolution in practice: Symptoms and fix patterns. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2020*. 617–628. https://doi.org/10.1145/3368089.3409760

[96] Sunil Kumar Mohanty, Gopika Premsankar, and Mario Di Francesco. 2018. An evaluation of open source serverless computing frameworks. In *Proceedings of the IEEE 10th International Conference on Cloud Computing Technology and Science, CloudCom 2018*. 115–120. https://doi.org/10.1109/cloudcom2018.2018.00033

[97] Stefan Nastic and Schahram Dustdar. 2018. Towards deviceless edge computing: Challenges, design aspects, and models for serverless paradigm at the edge. In *The Essence of Software Engineering*. Springer, Cham, 121–136. https://doi.org/10.1007/978-3-319-73897-0_8

[98] Robert G. Newcombe. 1998. Interval estimation for the difference between independent proportions: Comparison of eleven methods. *Statistics in Medicine* 17, 8 (1998), 873–890. https://doi.org/10.1002/(sici)1097-0258(19980430)17:8<873::aid-sim779>3.0.co;2-i

[99] Edward Oakes, Leon Yang, Dennis Zhou, Kevin Houck, Tyler Harter, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2018. SOCK: Rapid task provisioning with serverless-optimized containers. In *Proceedings of the 2018 USENIX Annual Technical Conference, ATC 2018*. 57–70.

[100] Christoffer Rosen and Emad Shihab. 2016. What are mobile developers asking about? A large scale study using Stack Overflow. *Empirical Software Engineering* 21, 3 (2016), 1192–1223. https://doi.org/10.1007/s10664-015-9379-3

[101] Joel Scheuner and Philipp Leitner. 2020. Function-as-a-Service performance evaluation: A multivocal literature review. *Journal of Systems and Software* 170 (2020), 110708. https://doi.org/10.1016/j.jss.2020.110708

[102] Carolyn B. Seaman. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering* 25, 4 (1999), 557–572. https://doi.org/10.1109/32.799955

[103] Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. 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, ATC 2020*. 205–218.

[104] Vaishaal Shankar, Karl Krauth, Kailas Vodrahalli, Qifan Pu, Benjamin Recht, Ion Stoica, Jonathan Ragan-Kelley, Eric Jonas, and Shivaram Venkataraman. 2020. Serverless linear algebra. In *Proceedings of the 2020 ACM Symposium on Cloud Computing, SoCC 2020*. 281–295. https://doi.org/10.1145/3419111.3421287

[105] Davide Taibi, Nabil El Ioini, Claus Pahl, and Jan Raphael Schmid Niederkofler. 2020. Serverless cloud computing (function-as-a-service) patterns: A multivocal literature review. In *Proceedings of the 2020 International Conference on Cloud Computing and Services Science, CLOSER 2020*. https://doi.org/10.5220/0009578501810192

[106] Ali Tariq, Austin Pahl, Sharat Nimmagadda, Eric Rozner, and Siddharth Lanka. 2020. Sequoia: Enabling quality-of-service in serverless computing. In *Proceedings of the 2020 ACM Symposium on Cloud Computing, SoCC 2020*. 311–327. https://doi.org/10.1145/3419111.3421306

[107] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. 2018. Peeking behind the curtains of serverless platforms. In *Proceedings of the 2018 USENIX Annual Technical Conference, ATC 2018*. 133–146.

[108] Sebastian Werner, Jörn Kuhlenkamp, Markus Klems, Johannes Müller, and Stefan Tai. 2018. Serverless big data processing using matrix multiplication as example. In *Proceedings of the IEEE International Conference on Big Data, Big Data 2018*. 358–365. https://doi.org/10.1109/bigdata.2018.8622362

[109] Xinli Yang, David Lo, Xin Xia, Zhiyuan Wan, and Jian-Ling Sun. 2016. What security questions do developers ask? A large-scale study of Stack Overflow posts. *Journal of Computer Science and Technology* 31, 5 (2016), 910–924. https://doi.org/10.1007/s11390-016-1672-0

[110] Tianyi Yu, Qingyuan Liu, Dong Du, Yubin Xia, Binyu Zang, Ziqian Lu, Pingchao Yang, Chenggang Qin, and Haibo Chen. 2020. Characterizing serverless platforms with serverlessbench. In *Proceedings of the 2020 ACM Symposium on Cloud Computing, SoCC 2020*. 30–44. https://doi.org/10.1145/3419111.3421280

[111] Tianyi Zhang, Cuiyun Gao, Lei Ma, Michael R. Lyu, and Miryung Kim. 2019. An empirical study of common challenges in developing deep learning applications. In *Proceedings of the 30th IEEE International Symposium on Software Reliability Engineering, ISSRE 2019*. 104–115. https://doi.org/10.1109/issre.2019.00020

[112] Yuhao Zhang, Yifan Chen, Shing-Chi Cheung, Yingfei Xiong, and Lu Zhang. 2018. An empirical study on TensorFlow program bugs. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2018*. 129–140. https://doi.org/10.1145/3213846.3213866