**Institute of Architecture of Application Systems**

# A Systematic Mapping Study on Engineering Function-as-a-Service Platforms and Tools

Vladimir Yussupov, Uwe Breitenbücher, Frank Leymann, Michael Wurster

Institute of Architecture of Application Systems,
University of Stuttgart, Germany,
{yussupov, breitenbuecher, leymann, wurster}@iaas.uni-stuttgart.de

BibTeX:

```
@inproceedings{Yussupov2019_SystematicMappingStudyFaaS,
  author    = {Vladimir Yussupov and Uwe Breitenb{\"u}cher and Frank Leymann
               and Michael Wurster},
  title     = {{A Systematic Mapping Study on Engineering
               Function-as-a-Service Platforms and Tools}},
  booktitle = {Proceedings of the 12th IEEE/ACM International Conference on
               Utility and Cloud Computing (UCC 2019)},
  publisher = {ACM},
  year      = 2019,
  month     = dec,
  pages     = {229--240},
  doi       = {10.1145/3344341.3368803}
}
```



Universität Stuttgart
Germany

# A Systematic Mapping Study on Engineering Function-as-a-Service Platforms and Tools

Vladimir Yussupov
Institute of Architecture of Application Systems
University of Stuttgart, Germany
yussupov@iaas.uni-stuttgart.de

Uwe Breitenbücher
Institute of Architecture of Application Systems
University of Stuttgart, Germany
breitenbuecher@iaas.uni-stuttgart.de

Frank Leymann
Institute of Architecture of Application Systems
University of Stuttgart, Germany
leymann@iaas.uni-stuttgart.de

Michael Wurster
Institute of Architecture of Application Systems
University of Stuttgart, Germany
wurster@iaas.uni-stuttgart.de

## ABSTRACT

Function-as-a-Service (FaaS) is a novel cloud service model allowing to develop fine-grained, provider-managed cloud applications. In this work, we investigate which challenges motivate researchers to introduce or enhance FaaS platforms and tools. We use a systematic mapping study method to collect and analyze the relevant scientific literature, which helps us answering the three clearly-defined research questions. We design our study using well-established guidelines and systematically apply it to 62 selected publications. The collected and synthesized data provides useful insights into the main challenges that motivate researchers to work on this topic and can be helpful in identifying research gaps for future research.

## CCS CONCEPTS

• **Software and its engineering → Cloud computing**.

## KEYWORDS

Serverless; FaaS; Function-as-a-Service; Systematic Mapping Study

**ACM Reference Format:**
Vladimir Yussupov, Uwe Breitenbücher, Frank Leymann, and Michael Wurster. 2019. A Systematic Mapping Study on Engineering Function-as-a-Service Platforms and Tools. In *Proceedings of the IEEE/ACM 12th International Conference on Utility and Cloud Computing (UCC '19), December 2–5, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3344341.3368803

## 1 INTRODUCTION

The landscape of cloud computing [18] service models is changing constantly. In general, cloud service models evolve in the direction of offering finer-grained compute models with reduced maintenance efforts, e.g., from Infrastructure- to Platform- and Software-as-a-service. Serverless computing [4] follows the overall direction of making infrastructure management a provider's responsibility. Serverless applications comprise provider-managed components, which creates a wrong impression that servers are absent. A recent newcomer in the line of *as-a-service* offerings called Function-as-a-Service (FaaS) is often associated with the term serverless, as it allows developers to compose applications using arbitrary, event-driven functions managed by providers. Prominent commercial and open source offerings include AWS Lambda [2], Microsoft Azure Functions [19], Apache Openwhisk [25], and OpenFaaS [20]. Moreover, FaaS employs an event-driven computing paradigm, meaning that FaaS-hosted functions are typically triggered by events. Another important aspect is a *scale-to-zero* property that helps saving costs by scaling idle instances to zero, which is typically not the case in, e.g., Platform-as-a-Service offerings. While there are works formulating open challenges for FaaS platforms [4, 26, 27], the rapid technology evolution complicates understanding why new platforms and tools are introduced in research instead of using existing technologies. Furthermore, the large amount of available proprietary and open source solutions raises a question if and how the introduced research concepts complement existing solutions.

In this systematic mapping study, we aim to answer the following research question: *what are the challenges and drivers motivating researchers to engineer new or extend existing FaaS platforms and platform-specific tools.* We rely on the well-known guidelines for conducting systematic mapping studies [14, 22, 23] and structure our work based on several existing systematic mapping studies [7, 8]. The main contributions of this work are: (i) a reproducible and reusable study design and classification framework and (ii) a systematically-obtained map of state-of-the-art research on the topic of FaaS platform and tooling engineering combined with analysis of relations of the proposed concepts to existing solutions published until June of 2019. We design and carefully follow a reproducible data search, selection, extraction, and synthesis process. The initial search is conducted using six well-known electronic databases, including IEEE Xplore, ACM Digital Library, and Springer Link. The resulting set of 62 publications is obtained via a multiphase process using precise selection criteria and techniques such as snowballing. Using a systematically-defined classification framework, we carefully synthesize and present the final results.

The remaining paper is structured as follows. In Sections 2 and 3 we describe the fundamentals and related work. Section 4 provides a detailed description of the study design. Sections 5 to 7 present the answers on the stated research questions. Finally, Section 8 describes the threats to validity, and Section 9 concludes this work.


*UCC '19, December 2–5, 2019, Auckland, New Zealand*

ACM ISBN 978-1-4503-6894-0/19/12.
https://doi.org/10.1145/3344341.3368803

## 2 FUNDAMENTALS

The term *serverless* is used in various contexts, e.g., to describe p2p networks, RFID-based protocols, and, more recently, a cloud paradigm that focuses on developing applications comprising provider-managed components [10]. FaaS is a novel cloud service model, which allows developers to deploy arbitrary, event-driven code snippets managed by cloud providers, that is commonly linked with the idea of serverless. Essentially, with FaaS the infrastructure is abstracted away giving an impression that servers are absent. This, however, is misleading, as resource management, monitoring, scaling, and fault tolerance become instead a burden of cloud service providers [4]. Frequently advertised features of FaaS include (i) a fine-grained programming model focusing on "boilerplate-less" business logic development, (ii) an automated and effortless scaling ensured by providers, (iii) a *scale to zero* property, which also results in a flexible cost model that ignores the idle periods. In spite of its advantages, FaaS has certain drawbacks [4, 13], which might influence the final decision on adopting it. For example, function execution is typically *limited in time*, with the actual limit depending on the cloud provider. Ephemeral and stateless FaaS functions often need to *persist the state* in shared storage systems, which imposes additional requirements on developers. Moreover, the *cold start* problem affects the startup time of first function instances, since the function's runtime must be equipped with dependencies required for running the function. Typically, function's compute resources are not terminated right after the execution, to allow reusing them for next calls. In case if there are no calls for a certain amount of time, compute resources are eventually scaled to zero.

## 3 RELATED WORK

To the best of our knowledge, there is no study analyzing the published research on engineering FaaS platforms and platform-specific tooling. Sadaqat et al. [24] conduct a multivocal literature review on serverless computing. The authors analyze core technological components, benefits and challenges (operational and for developers), and evolution trends for serverless computing. The study focuses on general aspects of serverless computing and is not sufficient for answering the research questions we define in this study.

Numerous works evaluate commercial and open source FaaS solutions [11, 15, 17] focusing on analysis of supported features and comparison based on the identified feature sets. Leitner et al. [16] presents a mixed-method empirical study focusing on FaaS development in industrial practice, which combines a gray literature analysis with interviews and online questionnaire. Multiple works discuss open challenges for FaaS platforms [4, 27] including software engineering, system (operational), and performance challenges. In addition, an initial version of the reference architecture for FaaS platforms is introduced by van Eyk et al. [26]. Several systematic mapping studies [1, 7, 8, 21] analyze various aspects of microservice architecture development such as research trends and challenges, various aspects related to architecting microservices, and the potential for industrial adoption of the introduced concepts. Di Francesco et al. [6] investigate aspects of migrating applications to microservice architectures by means of an industrial survey. Ghofrani and Lübke [12] conduct an empirical survey on the state of the practice in microservice architectures.

## 4 STUDY DESIGN

The goal of this research is to systematically collect and analyze the existing state-of-the-art research literature that focuses on developing new or enhancing existing FaaS platforms and tools. Essentially, the main research question is formulated as follows: "*What are the main challenges and drivers behind the newly-introduced or enhanced FaaS platforms and tools and how the proposed solutions are related to existing production-ready software?*" The study design follows the existing guidelines for conducting systematic mapping studies [5, 14, 22, 23], and its structure and content organization is influenced by existing, published systematic mapping studies on the topic of microservice architectures [7, 8]. In the following, we describe the study design in-detail.

### 4.1 Research Questions

To answer the main research question, we formulate more precise and finer-grained research questions as follows:

***RQ1***. "*What are the publication trends in published research focusing on developing new or modifying existing FaaS platforms and tools?*" By answering this question, we intend to understand the trends in state-of-the-art scientific research including (i) general statistics on years and publication types, which helps understanding the intensity of research on development and modification of FaaS platforms and tools, (ii) highlighting venues chosen most frequently to publish on the topic, and (iii) demonstrating the trends in preferred research strategies.

***RQ2***. "*What are the main challenges and drivers behind the published research on developing new or modifying existing FaaS platforms and tools?*" With this research question, we intend to explore the existing scientific work and to identify the challenges and drivers that motivate researchers to work on this topic as well as to cluster and analyze the correlations among proposed solutions. The results aim to (i) help in identifying the research gaps, (ii) serve as a basis for current or future research on FaaS platforms and tools, and (iii) identify features missing in existing production-ready solutions.

***RQ3***. "*What are the connections between the published research and industry?*" By answering this question, we want to understand how industry is involved, both in a form of company participation in authoring publications and reuse of existing production-ready FaaS solutions. These data might shed some light on how realistic is it to bring new concepts into production, e.g., more chances for mixed and industry-only publications, and how much effort does it take to achieve it, i.e., if the prototypes are available and can be reused. This includes statistics on: (i) industrial participation in scientific publications, (ii) commercial and open source FaaS platform and tooling usage in the proposed concepts, and (iii) availability of research prototypes for exploration and reuse.

### 4.2 Data Sources and Search Strategy

The initial release of a pioneering FaaS platform from Amazon Web Services called AWS Lambda happened in November of 2014 [3]. We chose to include all literature appearing five years before AWS Lambda was announced to capture any potential related work prior to beginning of the FaaS trend. As a result, the chosen time frame

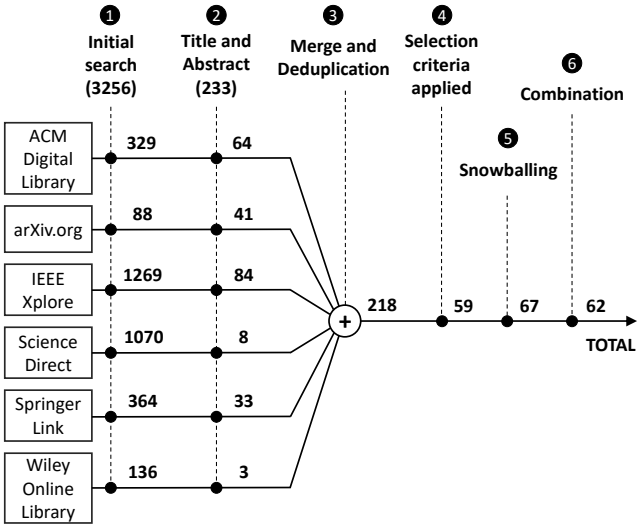


**Figure 1. A multiphase search and selection process (figure is based on the idea by Di Francesco et al. [8])**

for publications search is set for a time period from 2009 to 2019. Following the existing guidelines [14, 22, 23] and examples from multiple related works [5, 7, 9] we use a multiphase process, to have a better control over the final results. Figure 1 depicts an overview of the used multiphase search and selection process. In the following, we describe each phase in-detail starting with the description of initial search organization.

*4.2.1 Initial search.* For this systematic mapping study, we choose the electronic databases recommended in well-established guidelines [14, 22, 23], which are also used in numerous related studies [5, 7, 9]. To conduct the initial search, we queried six major scientific electronic databases, namely: (i) ACM Digital Library, (ii) arXiv.org, (iii) IEEE Xplore, (iv) Science Direct, (v) Springer Link, and (vi) Wiley Online Library. To increase the list of possible candidates, we define a generic search string, which contains high-level, FaaS-related keywords, as shown in Listing 1.

```
(serverless OR "Function-as-a-Service"
  OR FaaS OR "Function as a Service")
```

**Listing 1. A query string used for initial literature search**

The large number of false positives from IEEE Xplore and Science Direct is due to the fact that the keyword *FaaS* is used in multiple domains with different meanings, e.g., *flame atomic absorption spectrometry (FAAS)*. In addition, the stemming used by some search engines results in multiple false positives that refer to *Federal Aviation Administration (FAA)*. As a result, we search the results in arXiv.org, Springer Link, and Wiley Online Library using a Computer Science subject filter as it is supported by the respective search engines. The initial search resulted in collecting 3256 entries in total.

*4.2.2 Screening based on the titles and abstracts.* In this search phase, we pruned the set of results obtained during the previous phase based on their relevance to the topics of serverless computing and FaaS. The majority of the results were analyzed based on the title and abstract. However, when the topic of the paper was close and its relevance could not be determined from the title and abstract, we also used adaptive reading depth [23] to decrease the number of false negatives. After pruning the initial search results, we obtained 233 entries identified as *related to serverless or FaaS*.

*4.2.3 Merge and De-duplication.* After identifying the set of publications relevant to the topics of FaaS and serverless, we merged all results into a single dataset and identified duplicate entries based on the combination of a title, author names, publication year and venue. In cases when both, a pre-print and a published version were available, the preference was given to the latter. In total, after merge and de-duplication the dataset contained 218 entries.

*4.2.4 Applying the Selection Criteria.* In this search phase, the selected works are filtered based on a set of precise selection criteria listed in the following. For the majority of selected studies we used adaptive reading depth [23] as it was sufficient to determine the work's relevance. The entire dataset was separately screened by three researchers. All records of dubious relevance, e.g., a FaaS-based application, which incorporates a FaaS platform as its component for internal use, were marked with a *to-be-resolved* tag and further discussed until the consensus was reached. After applying the selection criteria, the size of the dataset reduced to 59 entries. The set of selection criteria defines which publications are to be included (✓) and which are to be excluded (×):

- ✓ Publications introducing novel general-purpose FaaS platforms and tooling, e.g., tooling related to testing, monitoring, or deployment automation, or extend existing ones with additional features.
- ✓ Publications that focus on architectural solutions, methods, algorithms, and optimization techniques targeting specific aspects of FaaS platforms or tools, e.g., the cold start problem or support for long-running tasks.
- ✓ Publications that are written in English
- × Publications on development of FaaS-based applications, and also solutions, which embed existing FaaS platforms as domain-specific job scheduling components.
- × Publications that evaluate and compare existing FaaS platforms and tools without proposing any modifications.
- × Secondary or tertiary studies, e.g., systematic literature reviews and surveys.
- × Publications not available as full-text or not in the form of a full research paper, e.g. extended abstract, presentation, tutorial, PhD research proposal, demo paper, as they do not provide enough details.

*4.2.5 Snowballing.* In the next phase, to increase the set of relevant literature we applied a complementary activity referred to as snowballing technique [29]. The main idea of backward snowballing is to search for related papers in the references section of each selected paper. In forward snowballing, for each selected paper one need to analyze all papers that cite it. We applied forward (using Google Scholar) and closed recursive backward snowballing. The new studies were checked against the selection criteria, which resulted in increase of the dataset's size to 67 entries.

*4.2.6 Combination.* As several works might be related to the same concept, e.g., a journal extension of a previously-published conference paper, in the next phase, we analyze and link related papers [8, 30]. This activity was conducted after the snowballing phase to increase the chances of finding more connections between selected papers. As a result, we identified a final set of 62 publications after completing all the phases.

## 4.3 Data Extraction and Synthesis

With respect to every research question, we extract the corresponding information as described in the following.

*4.3.1 Publication trends (RQ1).* To answer this question, we extract these data: (i) publication year, (ii) venue, e.g., conference, journal, or pre-print, (iii) name of the venue, e.g., IEEE CLOUD or USENIX ATC, and finally, (iv) research strategy.

*4.3.2 Challenges and drivers (RQ2).* To develop an initial classification framework, we apply the keywording technique [22]. The overall idea of keywording is to systematically define a classification scheme for categorizing the selected publications [23]. Keywords are assigned to the concepts identified in the publications and used to create an overall structure for categorizing the selected publication. The keywording process for defining a set of classification categories includes the following sequence of steps:

*a) Select an initial set of publications.* Two researchers randomly selected ten research papers each. This set of publications was used to derive an initial set of keywords.

*b) Identify keywords and concepts.* By reading the full text of publications from the initial set, a set of keywords describing related concepts was derived by combining all the collected information obtained from every publication in the initial set. The gathered information provided an initial overview of the landscape of targeted challenges and proposed solutions.

*c) Cluster keywords and formulate a starting list of categories.* The collected keywords were analyzed with respect to the open challenges described in existing scientific works [4, 26, 27] and clustered into an initial classification framework. Examples of obtained categories include *"Function execution"* or *"Testing and observability"*.

*d) Complete data extraction.* During this step, all remaining studies were analyzed based on the previously-defined framework. Additional relevant data missing in the framework was collected for refinement, which resulted either in reassessing the classification results or updating the framework.

*4.3.3 Connections with industry (RQ3).* To answer this question, we extract the following statistical data: (i) industrial participation in academic publications, (ii) production-ready FaaS solutions usage, and (iii) availability of prototypical implementations. The obtained information aims to show how often companies participate in published research, which FaaS solutions are frequently used, and how often the prototypes are available for reuse and how they are related to existing FaaS solutions, e.g., are based on them.

*4.3.4 Data Synthesis.* Finally, we analyze and summarize the final results by classifying them into a broader set of categories and using the narrative synthesis, i.e., using primarily the textual description for summarizing and explaining the findings in detail [8].

## 4.4 Information Management and Reproducibility of the Study

To facilitate the reproducibility of this mapping study, the self-contained replication bundle is made publicly-available [34]. The bundle comprises raw data for each step, the final list of selected research papers, and the data extraction forms.

For literature collection and de-duplication, we use JabRef[1], an open source bibliography reference manager. For collaboration and data extraction we use shared Google Docs and Google Sheets in particular. To automate the data extraction and avoid manual collection errors, we used extraction scripts created by means of Google Apps Script, which allows creating custom functions for Google Sheets. In all cases, scripts were used for simple traversal of the given sheet and manipulation with the collected data, e.g., counting total numbers based on the required conditions or concatenating multiple values and printing them in the desired format.

# 5 RESULTS: PUBLICATION TRENDS (RQ1)

In this section, we present the results obtained after synthesizing the data extracted for the first research question.

***Publication years.*** In Figure 2, which depicts the distribution of publication types over the years with respect to quarters, we can observe a growing interest in the topic of FaaS platforms and tools. The first relevant publications started to appear in 2016 (2/62), with one focusing on developing an open source research FaaS platform (P13) and another focusing on the cost modeling for microservices and FaaS-based applications. In the next three years the amount of publications increased drastically, with the sharp rise in 2018 (33/62). Figure 2 shows the presence of eleven publications in the second and the third quarters of 2019, which, however, represent those quarters in the final set of selected publications only partially, since the search was completed by mid June 2019.

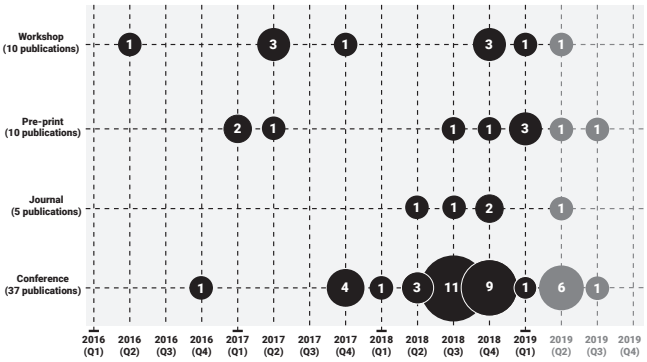


**Figure 2. Distribution of publication types per year**

***Publication types.*** The most popular publication type is a *conference paper* (37/62), while the second place is shared between the *workshop paper* (10/62) and *pre-prints* (10/62) via arXiv.org. As both workshop papers and pre-prints are considered to be a good option for getting the feedback on early work, the increase in numbers might serve as an additional indicator of a growing interest in the

[1] https://www.jabref.org

Table 1. The list of publications with respect to their research strategy

| Research Strategy | Count | Ratio | Publications |
|---|---|---|---|
| **Validation research** | 42 | 68% | P5, P7, P9, P10, P11, P12, P14, P15, P16, P17, P19, P20, P22, P23, P24, P27, P28, P31, P32, P33, P36, P37, P40, P41, P42, P44, P45, P46, P48, P49, P50, P51, P53, P54, P55, P56, P57, P58, P59, P60, P61, P62 |
| **Solution proposal** | 10 | 16% | P2, P6, P13, P18, P21, P25, P29, P30, P39, P52 |
| **Evaluation research** | 9 | 14% | P1, P3, P4, P8, P26, P35, P38, P43, P47 |
| **Philosophical paper** | 1 | 2% | P34 |

topic. Finally, *journal article* (5/62) publication type is starting to appear only in 2018, which can be explained by the longer duration of a journal publishing process.

***Publication venues.*** In general, the list of venues preferred by researchers is large and fragmented, with the 42 distinct venues in total. Apart from the pre-prints published directly to arXiv.org (10 publications), the top three venues include International Workshop on Serverless Computing (WoSC) (6 publications), IEEE CLOUD (3 publications), and USENIX ATC (3 publications). Overall, the venues span multiple fields, mostly under the umbrella of cloud computing, but also including such topics as edge or fog computing, service-oriented computing, distributed systems, Internet of Things, general software engineering, parallel processing and high-performance computing. Such heterogeneity results in no clear leader in the list of preferred venues and indicates that the topic of FaaS is being investigated from different angles. A few remarks have to be added about the Workshop on Serverless Computing, which is dedicated explicitly to the topic of serverless computing. Starting from 2017, WoSC was a part of such conferences as IEEE ICDCS, ACM/IFIP Middleware, IEEE CLOUD, and IEEE/ACM UCC. While in the obtained publications' metadata WoSC was explicitly mentioned only for one publication (P55), it was implied in several publications from the final list [31–33]. For example, publications presented at IEEE ICDCSW 2017 (P23, P25) and UCC 2018 Companion publications (P44, P52, P54) were actually presented as a part of WoSC agenda. A large number of papers published via a specialized workshop indicates a keen interest of the community in a specialized venue focused entirely on the topic of serverless computing.

***Research strategies.*** To identify a research strategy, we use the classification approach by Wieringa et al. [28] due to its relative simplicity and wide acceptance among researchers [1, 8]. Table 1 and Figure 3 demonstrate the statistics on research strategies.

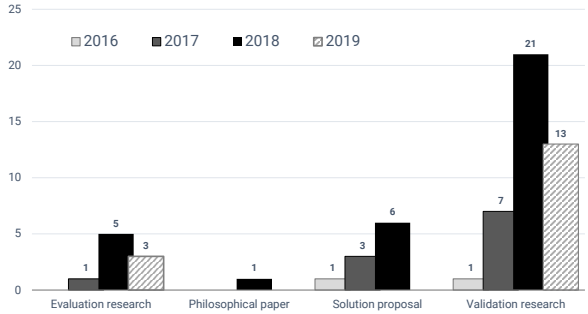


Figure 3. Distribution of research strategies per year

The majority of publications use *validation research* strategy (42/62), in which new techniques are described and validated, e.g., via lab experiments, but not implemented in practice, with prototypes and lab experiments being the most popular validation approaches for our chosen topic. Such a large representation of validation research showcases the strong focus on verifying and confirming the introduced hypotheses and highlights the practical orientation of research on the topic. The *solution proposal* (10/62) is the second popular strategy, in which new solutions or significantly-extended techniques are proposed, with small examples or good argumentations showing their benefits. One possible reason for presenting less validated ideas is that FaaS is still relevantly new, with a large space of open problems and their possible solutions. The *evaluation research* (9/62) strategy, i.e., where a technique is implemented and evaluated in practice, e.g., an industrial case study or experiment with practitioners, is on the third place. The biggest increase in evaluation research happens in 2018 and 2019, which might indicate a rising interest for industrial participation in research publications, also, considering the fact that results for 2019 cover only the first 5 months of the year. Finally, the *philosophical paper* (1/62) strategy, which proposes a new way of observing existing concepts, e.g., by introducing a conceptual framework, is the least frequent research strategy for the topic of this study.

**Main findings: Publication trends**

- ➔ Relevant publications first appear in 2016, with the rapid increase of interest in the next years.
- ➔ Conference is the most preferable (68%) venue type for presenting the research on this topic.
- ➔ The list of venues spans multiple different fields without a clear leader. At the same time, multiple papers are published via a serverless-specific workshop, which indicates a keen interest in having a dedicated serverless venue.
- ➔ Presented research is practice-oriented, with most concepts (82%) validated/evaluated via prototypes, lab experiments, or case studies.
- ➔ Rising evaluation research numbers, which increased from one publication in 2017 to five in 2018 and three by mid June 2019 might indicate a keen interest in industrial collaborations.

## 6 RESULTS: CHALLENGES & DRIVERS (RQ2)

In this section, we present and discuss the main challenges and drivers for publishing research on engineering FaaS platforms and tools identified by analyzing the final set of selected scientific publications. Table 2 shows the distribution of publications with respect to the list of categories defined using the method described in Section 4. In general, the research challenges are heterogeneous yet often tightly-coupled and even overlapping. For example, enhancement of a function runtime in a FaaS platform might try tackling multiple challenges simultaneously, e.g., improving the performance and optimizing costs. Moreover, the described solution might be simultaneously related to several categories, e.g., secure execution of functions can be tackled by combining enhanced function containers (compute resource) with security-optimized API Gateway, which routes encrypted inputs and outputs (function routing). To facilitate understanding of the global picture, we also link the identified challenges with layers of the FaaS platform reference architecture described in one of the selected publications (P34).

***FaaS platform reference architecture.*** One of the selected publications (P34) introduces the FaaS platform reference architecture [26] consisting of five layers. The *Resource layer* is the lowest layer that comprises compute, e.g., virtual machines or containers, network, and storage resources. Next, the *Resource Orchestration layer* manages these resources including such tasks as scheduling and resource allocation. Often, production-ready solutions use container orchestration platforms for this layer, e.g., Kubernetes. The *Function Management layer* is responsible for managing function's lifecycle (scaling, deleting instances), maintaining functions registry, routing I/O, and scheduling executions. The *Function Orchestration layer* is responsible for composing functions, e.g., by means of workflows. Finally, the *DevOps layer* groups cross-cutting concerns such as monitoring or benchmarking and often fits nicely as a grouping entity for platform-specific tools.

***Function execution.*** The largest segment of selected publications focuses on various aspects related to *function execution* in FaaS platforms. This category groups together multiple closely-related concepts such as function scheduling, resource allocation and management, or function runtime, which includes underlying compute resources, e.g., Docker containers and the actual language runtimes, e.g., Java Virtual Machine, etc. In the following, we summarize the challenges and drivers related to this category, with the two largest categories being related to performance optimizations and security enhancements as shown in Table 2.

*a) Performance optimization.* In the context of function execution, the most frequently-targeted research challenge is performance optimization. Essentially, various strategies targeting different layers of reference architecture are proposed to optimize the performance. It is worth mentioning that multiple publications which try to optimize execution performance, typically, tackle several challenges simultaneously, e.g., reducing the startup overhead and at the same time improving performance isolation (P12). One common way of optimizing execution performance described in multiple works is to modify or reconsider the underlying *compute resource* used in existing FaaS platforms (e.g., P1, P6, P12, P19, P26). As an option, compute resources are suggested to be (i) *enhanced* (P1, P19, P26), e.g., by improving some internal characteristics of the compute resource, or (ii) *replaced (P6, P12)*, e.g., run functions using WebAssembly or Rust language's runtime instead of containers. One introduced enhancement for compute resources is to optimize characteristics of Linux containers (P26), e.g., by using bind-mounts and a pool of prepared cgroups to speed up the overall startup process. Another examples include (i) usage of containers only for application-level isolation (P1), while functions of the same application can have weaker isolation mechanisms, and (ii) adding a support for GPU-based computations in Docker containers (P19), which accelerates compute-intensive jobs, e.g., training deep learning models.

Essentially, the publications trying to reduce function startup overhead (also by means of optimized compute resources) are implicitly connected with the *cold start* problem even if this is not explicitly stated. This FaaS-specific limitation attracts a lot of attention and some publications focus exclusively on tackling this problem, e.g., by maintaining a *pool of warm nodes* (P22) or using *checkpointing* to optimize JVM cold start overhead (P35). Another solution tackles the startup overhead by introducing package caching mechanisms for instantiating pre-initialized Python runtimes (P25).

The next big segment of publications aiming to optimize the function execution performance is related to *function scheduling* and *resources allocation*. Here, researchers try to optimize resource utilization and reduce response time overhead (e.g., P5, P17, P18, P23, P32), or minimize the redundant function calls (P56), i.e., multiple idempotent calls with the same inputs. In some cases, function scheduler component is implicitly affected as a part of a larger concept, e.g., to optimize the execution of functions the scheduling logic has to be adapted (P1). Essentially, researchers focus either on the function runtime in general (e.g., P9, P11) or on particular resource types such as CPU (P20) or storage (P46). The majority of proposed concepts use proactive and predictive (P11, P14, P15, P20, P28) techniques, which highlights the interest in improved dynamism of resource orchestration in FaaS platforms. Typically, proposed function scheduling solutions are implemented on the function management layer, with the function scheduler component being the target for changes. As a side note, one scheduling-related driver (P10) is to support defining custom algorithms in the platform, which can simplify developing new scheduling algorithms.

One of the observations made during the analysis of performance-focused publications is that traditional operating systems such as Linux (together with its container virtualization) are considered to be not efficient enough for the serverless computations. The idea of implementing a *serverless operating system* (P2, P21) where the underlying abstractions are tailored for serverless workloads might also be considered as an additional confirmation of this observation.

*b) Security.* The majority of concepts trying to enhance security properties such as *confidentiality*, *integrity*, or *accountability*, use Intel SGX (P3, P7, P27, P33), a hardware-backed trusted execution environment, which can be utilized for running functions in private memory regions called enclaves. Typically, also the API Gateway is proposed to be secured as well as the function's inputs and outputs, emphasizing the privacy aspects of function interactions. Composite approaches, e.g., combining Intel SGX and a WebAssembly-based isolation for function runtime, are also discussed (P27).

Table 2. The connections among selected publications and the defined categories (one publication might simultaneously be related to several categories)

| Category | Count | Total share | Publications |
|---|---|---|---|
| **Function execution** | **36 / 62** | 58% | |
| • **Performance** | 26 / 36 | | P1, P2, P5, P6, P9, P10, P11, P12, P14, P15, P17, P18, P19, P20, P21, P22, P23, P25, P26, P28, P32, P35, P36, P42, P46, P55 |
| • **Security** | 5 / 36 | | P3, P4, P7, P27, P33 |
| • **Long-running tasks support** | 2 / 36 | | P16, P30 |
| • **Fault tolerance mechanisms** | 1 / 36 | | P8 |
| • **Function composition support** | 1 / 36 | | P45 |
| • **Language runtime support** | 1 / 36 | | P57 |
| **Deployment environments support** | **6 / 62** | 10% | P12, P16, P29, P36, P42, P58 |
| **Testing & observability** | **5 / 62** | 8% | P39, P50, P51, P53, P61 |
| **Benchmarking** | **5 / 62** | 8% | P37, P41, P44, P52, P56 |
| **Costs optimization** | **5 / 62** | 8% | P38, P40, P48, P49 |
| **Programming models** | **3 / 62** | 5% | P24, P45, P54 |
| **Research-centric platforms** | **3 / 62** | 5% | P10, P13, P31 |
| **Deployment automation** | **2 / 62** | 3% | P47, P62 |
| **Migration** | **2 / 62** | 3% | P59, P60 |
| **CI/CD pipelines** | **1 / 62** | 2% | P43 |
| **Reference architecture** | **1 / 62** | 2% | P34 |

*c) Other challenges.* This segment comprises challenges that are tackled less frequently. Several papers focus on tackling the problem of *long-running tasks*, e.g., by using Docker's checkpointing features to pause/resume such tasks (P16) or by moving the function with its state, i.e., strong code mobility, to another node for continuing computations when the allowed time limit is reached (P30). Other challenges include enhancing runtime's *fault tolerance mechanisms* (P8), *supporting function composition* (P45), and adding support for a new *language runtime* (P57) – in this case allowing to execute containers based on Docker images in AWS Lambda.

Most solutions related to function execution focus on the Resource layer (compute resources), while also affecting the Resource orchestration and Function Management layers, e.g., by modifying function/resource scheduler of function router components. Publications focusing on resources allocation mainly concentrate the efforts on the resource orchestration layer. In rare cases, the proposed changes affect all layers, e.g., serverless OS or are related to Function Composition (P45) or DevOps layers (P57).

***Deployment environments support***. Supporting new deployment environments, e.g., multi- or hybrid-cloud, edge/fog computing is another challenge, which is also influenced by the advent of Internet of Things. The properties of FaaS allow using such platforms for running tasks on edge or IoT devices (P12, P16, P29, P58), e.g., data pre-processing jobs, but require additional enhancements, e.g., supporting pausing/resuming computations for long-running tasks (P16) to help reducing the power consumption. Other challenge related to deployment environments is to support scheduling functions in different target environments, e.g., multiple cloud providers (P36), where functions can be scheduled in multiple environments depending on their performance monitored in real time, or selecting a performance- or cost-optimal cloud service model (P42), e.g., FaaS platform or a hosted virtual machine. The introduced concepts target different architecture layers, e.g., DevOps layer (P36, P42) by proposing a new tool, or going more to platform-specific layers such as Resource, Resource orchestration, and Function Management layers (P12, P16, P29, P58).

***Testing and observability***. The publications in this category are concerned with providing support for testing, debugging, monitoring, and logging of FaaS-based applications. The proposed solutions target exclusively the DevOps layer, with the majority focusing on development of a specialized tool. One of the frequent challenges appearing in identified works is providing means for dependency tracing (P50, P51, P61). Here, presented works introduce provider-agnostic (P50) or provider-specific (P51) tools, or focus on the modeling approaches that can be used for tracing the dependencies as well as, e.g., for testing (P61). Other solutions discuss how to use logging for monitoring and debugging FaaS-based applications (P53) or how logs can be visualized (P39). All proposed concepts are validated by means of prototypes that are related to the DevOps layer of the reference architecture.

***Benchmarking***. This category includes publications focused on benchmarking approaches (P37, P41, P44, P52, P56). Typically, the main target of the proposed benchmarks is the performance and costs. As a subtype of performance-related benchmarks, the cold start influencing factors are being benchmarked in one of the studies (P52). The proposed benchmarks and tools target the DevOps layer of the reference architecture.

***Costs optimization***. As a logical follow-up, the cost optimization (P38, P40, P48, P49) is a challenge that also attracts the interest of researchers. The proposed solutions, however, are not always FaaS-specific, but rather focus on microservices and cloud in general. The proposed concepts are implemented exclusively in the DevOps layer, typically, by introducing a new tool, which might also implement an introduced model.

***Programming models.*** Another category is related to new programming models for FaaS, e.g., retroactive programming for FaaS-based applications (P24) which allows modifying and replaying execution histories using event sourcing. Another examples include formal foundations and serverless programming language for function orchestration (P45) or visual programming language for developing FaaS-based applications (P54). Here, the solutions target the DevOps layer except for the function composition prototype, which targets platform's layers up to Function Orchestration layer.

***Research-centric platforms.*** Public availability of the source code is important, as it helps researchers to avoid reimplementing the wheel by building on top of working solutions. The challenge of having an open source and easy to use research platform is highlighted in several publications (P10, P13, P31). One example (P10) aims to allow defining custom scheduling algorithms in the platform, which can simplify developing new scheduling algorithms in the future. Since the outcome of this challenge is a FaaS platform, all platform-specific layers of the reference architecture are affected.

***Deployment automation.*** The topic of deployment automation is highly-relevant to the serverless application model. The main aspects investigated by researchers are how to support modeling, e.g., by using cloud modeling languages such as TOSCA or CAMEL [5], and automate the deployment of the FaaS-based applications using deployment technologies. The solutions tackling this challenge are implemented exclusively in the DevOps layer (P47, P62).

***Migration.*** Another category we identified is related to migration of legacy application to FaaS service model (P59, P60). Here, the focus of researchers is how to facilitate *faasification* of functions in a chosen application. Identifying FaaS-compatible functionalities and automating the migration process are interesting topic in this area. The DevOps layer is used for implementing the concepts, since the specific tooling is needed that can analyze, extract, and deploy functions of the application to a FaaS platform.

***CI/CD pipelines.*** Setting up a working continuous integration and continuous delivery pipeline is the main subject in one publication (P43), where the CI/CD pipeline is implemented and evaluated by means of an industrial case study. The solution is related to the DevOps layer of the reference architecture.

**Main findings: Challenges and drivers**

- ➔ Most publications (58%) focus on challenges of function execution in FaaS platforms.
- ➔ Function execution performance is the most popular challenge (42%) discussed in 26 publications with solutions proposed in different layers of a FaaS platform's architecture.
- ➔ Frequently-used performance optimizations for function execution include modification of the underlying compute resources, function scheduling, and resources allocation.
- ➔ Multiple challenges are tackled by introducing FaaS-specific tools, which might indicate a need in extending FaaS tooling landscape with more tools, e.g., for testing, benchmarking etc.

# 7 RESULTS: LINKS WITH INDUSTRY (RQ3)

In this section, we present and discuss the results of data analysis required to answer the third research question.

***Industry participation in research publications.*** To analyze industry participation, we record authors' affiliations and group publications into three categories, namely academic-only, industry-only and mixed publications, depending on the presence of a company names in listed authors' affiliations as shown in Table 3.

**Table 3. Affiliation types in published research**

| Affiliation type | Count | Ratio | Publications |
|---|---|---|---|
| **Academic-only** | 39 / 62 | 63% | P5, P7, P9, P10, P11, P12, P13, P14, P15, P17, P19, P20, P22, P23, P24, P27, P28, P29, P30, P31, P33, P36, P37, P40, P41, P42, P44, P45, P48, P49, P50, P52, P53, P57, P58, P59, P60, P61, P62 |
| **Mixed** | 16 / 62 | 26% | P2, P3, P4, P6, P8, P16, P25, P26, P34, P38, P43, P46, P47, P51, P54, P56 |
| **Industry-only** | 7 / 62 | 11% | P1, P18, P21, P32, P35, P39, P55 |

One of the main finding obtained after analyzing the affiliations shows that the industrial and mixed contributions constitute more than one third of the collected publications. In particular, industry-only (7/62) and mixed (16/62) publications constitute approximately 37% of the total number of publications. Despite the fact that this topic was initially driven by industry, the companies are interested in publishing FaaS-related research, with such notable names as Nokia Bell Labs, IBM Watson Research Center, or Huawei being among the listed affiliations. This might indicate an opportunity to increase the amount of evaluation research segment in future by having more industrial collaborations.

***Use of existing FaaS software.*** To analyze the usage of existing FaaS software, we counted the number of distinct mentions of existing FaaS solutions in the validation and evaluation sections, which resulted in a general picture on preferred FaaS technologies. Table 4 demonstrates the overall usage statistics for existing FaaS solutions. The leading FaaS platform mentioned either as a part of the implementation, e.g., benchmarking tool, or as a part of the evaluation is AWS Lambda (22/62). The second place is taken by Apache Openwhisk (10/62), an open source FaaS platform which is also used by IBM in their commercial offering called IBM Cloud Functions (4/62). Therefore, these two platform choices are strongly-related. The third place is shared by Microsoft Azure Functions (5/62) and OpenLambda (5/62). An interesting point here is that OpenLambda was originally introduced as a research platform and multiple concepts are implemented using this platform, which also resonates with the respective challenge described in Section 6. As seen previously, a popular open source alternative such as Apache Openwhisk can be

a good choice for evaluating proposed concepts with reduced efforts, since it is also represented as a commercial offering – IBM Cloud Functions. FaaS platforms such as Google Cloud Functions (3/62), Huawei FunctionStage (2/62), OpenFaaS (1/62), IronFunctions (1/62) are mentioned less frequently.

**Table 4. Production-ready FaaS software usage**

| Name | Count | Ratio | Publications |
|---|---|---|---|
| **AWS Lambda** | 22 / 62 | 36% | P4, P9, P23, P31, P36, P37, P40, P41, P43, P44, P46, P47, P48, P50, P51, P53, P54, P57, P59, P60, P61, P62 |
| **Apache Openwhisk** | 10 / 62 | 16% | P1, P3, P4, P12, P15, P23, P33, P36, P39, P45 |
| **Microsoft Azure Functions** | 5 / 62 | 8% | P37, P41, P44, P47, P50 |
| **OpenLambda** | 5 / 62 | 8% | P5, P25, P26, P27, P28 |
| **IBM Cloud Functions** | 4 / 62 | 7% | P36, P37, P41, P55 |
| **Google Cloud Functions** | 3 / 62 | 5% | P23, P37, P41 |
| **Huawei FunctionStage** | 2 / 62 | 3% | P8, P35 |
| **OpenFaaS** | 1 / 62 | 2% | P56 |
| **IronFunctions** | 1 / 62 | 2% | P19 |

***Prototypes availability for proposed concepts.*** After analyzing the descriptions of prototype and evaluation sections, we identified that 73% (45/62) of publications provide a description of the prototypical implementation with varying level of details. Table 5 shows the overall statistics on the description and accessibility of the prototypical implementations. However, out of these 45 publications, slightly more than a half (23/45 or 51%) provide an accessible link to the source code, with most of them being hosted on Github. While slightly more than half of the described prototypes are available, our observation demonstrates that in many cases reuse or verification of proposed concepts is not directly possible, as it requires additional efforts, i.e., contacting the authors, and might not be feasible at all in cases when the concepts have to be reimplemented from scratch. Figure 4 demonstrates the ratio between described and accessible prototypes per affiliation type. It is easy to notice that less than half of publications actually provide prototype links regardless of affiliation type. Although prototypes might be less relevant depending on the publication type, e.g., solution proposals or philosophical papers will less likely provide a prototypical implementation, in our dataset most publications are either validation or evaluation research publications. During the data extraction phase, we observed that a fair amount of papers focusing on the evaluation of proposed concepts, e.g., by means of performance measurements, while highlighting various technical details of implemented prototypes and evaluation setups tend to omit prototype links; however, further investigation of potential correlations related to this issue was not in the scope of this work.

**Table 5. Description and accessibility of prototypical implementations**

| Status | Count | Ratio | Publications |
|---|---|---|---|
| **Prototype described** | 45 / 62 | 73% | P1, P3, P4, P5, P7, P10, P12, P13, P14, P15, P16, P19, P20, P22, P23, P24, P27, P28, P31, P32, P33, P35, P36, P37, P38, P39, P41, P44, P45, P46, P47, P48, P49, P50, P51, P53, P54, P55, P56, P57, P58, P59, P60, P61, P62 |
| **Prototype accessible** | 23 / 62 | 37% | P4, P5, P10, P23, P24, P31, P37, P38, P39, P41, P44, P46, P47, P48, P49, P53, P54, P57, P58, P59, P60, P61, P62 |

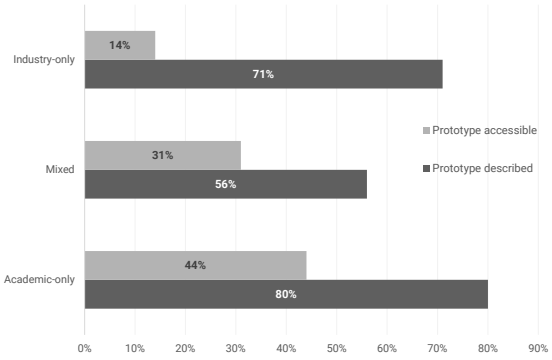


**Figure 4. Prototype description and accessibility per affiliation type**

**Main findings: Connections with industry**

- ➔ More than one third of publications (37%) are authored together with industrial partners.
- ➔ The most popular FaaS platform used in validations and evaluations is AWS Lambda (mentioned in 36% of publications).
- ➔ The second most popular option is Apache Openwhisk (16%), which might be helpful for research prototypes, as it is also used in a commercial offering from IBM.
- ➔ While 73% of the publications describe prototypical implementations, roughly half of them are accessible and can be verified or reused.

## 8 THREATS TO VALIDITY

To assess the quality of our systematic study, we used the guidelines for computing the score described by Petersen et al. [23]. The main idea is to follow the described checklist and compute the ratio of the applied activities compared to the total number of suggested

activities in the checklist. The total score in our case is equal to 58%, which is considered to be a good score for systematic mapping studies, with the median value for such studies in the literature being 33% [8]. In this section, we reflect on the methodology of this study and discuss potential threats to validity and the measures we applied to mitigate these threats.

***Selection bias.*** To tackle one of the main threats to this study, i.e., if the primary set of publications is not representative enough, we used a multi-phase search strategy with initial search conducted against six well-established electronic research databases. The initial search was performed using corresponding databases' search engines and based on the publication's meta-data including abstract, title, and author-defined keywords. One possible threat here is to miss studies that do not explicitly have any of the used search terms in their meta-data. To mitigate this risk, we intentionally defined the search string using only high-level, generic keywords to reduce the amount of false negatives, which resulted in a relatively-large size of the initial dataset compared to the final dataset's size. In addition, the search strategy also included the forward and backward snowballing to increase the final set of primary publications. During the next phases, we used a deterministic set of selection criteria, which was initially validated before starting this study in a set of trial runs and refined based on the obtained results. To decrease the number of false negatives, we also used adaptive reading depth [23] in cases when it was not possible to determine the relevance of the study based only on title and abstract. The study selection was separately performed by three researchers, with the consecutive resolution of conflicts until the consensus was reached. There were no cases, when the agreement was not reached.

***Data extraction and internal validity.*** Laborious and tedious data extraction process can pose another important threat to this study, namely a potential inaccuracy in data extraction. To mitigate this threat, we defined our research protocol based on the well-established guidelines [14, 22, 23] and existing published systematic mapping studies [7, 8]. As one of the important steps for extracting the data relevant to the set of predefined research questions, we designed data extraction forms to facilitate the analysis and synthesis of the information needed for answering these questions. The designed classification framework was iteratively refined using a keywording technique while the existing related work was carefully taken into consideration. The extraction and synthesis of the data was partially automated, especially in the error-prone parts, e.g., where the list of the publications related to a certain category had to be retrieved. The data extraction process was conducted by two researchers, one being the main data extractor, whereas the second was verifying the extracted data. In cases of any disagreements, researchers resolved them via discussions to ensure the correctness of the extracted data.

***Reproducibility.*** To facilitate the reproducibility and verifiability of this study, we carefully documented all steps of our search and selection strategy, and made our results publicly-available. However, there is a potential threat related to the reproducibility aspect. The search engines of the electronic research databases typically do not provide public web APIs for automating the search and offer varying sets of features, e.g., export of the results in specific formats such as .bib is not supported by Springer Link. Thus, the results obtained during the initial search require significant post-processing efforts, either manual or required for automating the entire process.

***External validity.*** We designed our study focusing on a combination of peer-reviewed academic works *and* pre-prints available via arXiv.org, which makes it possible to generalize our work only for these types of publications. We did not intend to include gray literature in a form of blog posts as well as the industrial efforts not published via academic research channels. This decision was motivated by the fact that relevant entries might not be available due to the high dynamics of such data sources [16], which might increase the chance of introducing systematic errors and affecting the overall quality of the mapping study [9].

## 9 CONCLUSION

In this work, we investigate which challenges and drivers motivate researchers to engineer new or extend existing FaaS platforms and platform-specific tools, as formulated in Section 1. We split this research question into three sub-questions and addressed them by synthesizing the data extracted from 62 publications obtained using a systematic multiphase search and selection process. In the *first research question* we analyzed the publication trends on the topic of FaaS platforms and tools engineering. The initial works on the topic appear in 2016, with the significant increase in number of publications in 2018. Most works are being presented at conferences and the list of chosen venues spans multiple different domains. Presented concepts are often validated, and there is an increasing number of evaluation research involving companies, which serves as an indicator of a practice-orientation of the topic.

In the *second research question*, we classified publications based on the challenges and drivers motivating researchers to engineer new or extend existing FaaS platforms and tools. Essentially, the challenges tackled by researchers are heterogeneous and target different layers of the FaaS platform. The majority of works focus on optimizing the performance of function execution aspects using various strategies, e.g., optimize or replace function runtime, improve function scheduling and resources allocation. Multiple works focus on challenges that require introducing new FaaS tools, e.g., for benchmarking FaaS providers, for testing and monitoring, or automating the deployment of FaaS-based applications.

The *third research question* focused on how industry is connected with the published research. More than one third of publications have one author with an industrial affiliation, which indicates a strong industrial influence on the topic. Multiple proposed concepts also rely on the production-ready solutions either for validation or evaluation purposes, with AWS Lambda being the most popular one. While a significant segment of publications describes the details on prototypical implementations of the proposed concepts, roughly a half of them provides an accessible link, which complicates the process of reusing and verifying them.

## ACKNOWLEDGMENTS

This work is partially funded by the European Union's Horizon 2020 research and innovation project *RADON* (825040). We would also like to thank the anonymous referees, whose insightful feedback helped to improve this paper.

## Appendix A THE FINAL LIST OF SELECTED PUBLICATIONS

| ID | Authors | Publication Title | Year |
|---|---|---|---|
| P1 | Akkus et al. | SAND: Towards High-performance Serverless Computing | 2018 |
| P2 | Al-Ali et al. | Making Serverless Computing More Serverless | 2018 |
| P3 | Alder et al. | S-FaaS: Trustworthy and Accountable Function-as-a-Service using Intel SGX | 2018 |
| P4 | Alpernas et al. | Secure Serverless Computing Using Dynamic Information Flow Control | 2018 |
| P5 | Aumala et al. | Beyond Load Balancing: Package-Aware Scheduling for Serverless Platforms | 2019 |
| P6 | Boucher et al. | Putting the "Micro" Back in Microservice | 2018 |
| P7 | Brenner and Kapitza | Trust More, Serverless | 2019 |
| P8 | Chan et al. | BalloonJVM: Dynamically Resizable Heap for FaaS | 2019 |
| P9 | Christoforou and Andreou | An effective resource management approach in a FaaS environment | 2018 |
| P10 | Danayi and Sharifian | openCoT: The opensource Cloud of Things platform | 2019 |
| P11 | Danayi and Sharifian | PESS-MinA: A Proactive Stochastic Task Allocation Algorithm for FaaS Edge-Cloud environments | 2018 |
| P12 | Hall and Ramachandran | An Execution Model for Serverless Functions at the Edge | 2019 |
| P13 | Hendrickson et al. | Serverless Computation with openLambda | 2016 |
| P14 | HoseinyFarahabady et al. | A QoS-Aware Resource Allocation Controller for Function as a Service (FaaS) Platform | 2017 |
| P15 | HoseinyFarahabady et al. | A Model Predictive Controller for Managing QoS Enforcements and Microarchitecture-Level Interferences in a Lambda Platform | 2018 |
| P16 | Karhula et al. | Checkpointing and Migration of IoT Edge Functions | 2019 |
| P17 | Kesidis | Temporal Overbooking of Lambda Functions in the Cloud | 2019 |
| P18 | Kim and Cha | Design of the Cost Effective Execution Worker Scheduling Algorithm for FaaS Platform Using Two-Step Allocation and Dynamic Scaling | 2018 |
| P19 | Kim et al. | GPU Enabled Serverless Computing Framework | 2018 |
| P20 | Kim et al. | Dynamic Control of CPU Usage in a Lambda Platform | 2018 |
| P21 | Koller and Williams | Will Serverless End the Dominance of Linux in the Cloud? | 2017 |
| P22 | Lin and Glikson | Mitigating Cold Starts in Serverless Platforms: A Pool-Based Approach | 2019 |
| P23 | McGrath and Brenner | Serverless Computing: Design, Implementation, and Performance | 2017 |
| P24 | Meissner et al. | Retro-$\lambda$: An Event-sourced Platform for Serverless Applications with Retroactive Computing Support | 2018 |
| P25 | Oakes et al. | Pipsqueak: Lean Lambdas with Large Libraries | 2017 |
| P26 | Oakes et al. | SOCK: Rapid Task Provisioning with Serverless-optimized Containers | 2018 |
| P27 | Qiang et al. | Se-Lambda: Securing Privacy-Sensitive Serverless Applications Using SGX Enclave | 2018 |
| P28 | Saha and Jindal | EMARS: Efficient Management and Allocation of Resources in Serverless | 2018 |
| P29 | Soltani et al. | Towards Distributed Containerized Serverless Architecture in Multi Cloud Environment | 2018 |
| P30 | Soltani et al. | A Migration-based Approach to execute Long-Duration Multi-Cloud Serverless Functions. | 2018 |
| P31 | Spillner | Snafu: Function-as-a-Service (FaaS) Runtime Design and Implementation | 2017 |
| P32 | Stein | The Serverless Scheduling Problem and NOAH | 2018 |
| P33 | Trach et al. | Clemmys: Towards Secure Remote Execution in FaaS | 2019 |
| P34 | van Eyk et al. | A SPEC RG Cloud Group's Vision on the Performance Challenges of FaaS Cloud Architectures | 2018 |
| P35 | Wang et al. | Replayable Execution Optimized for Page Sharing for a Managed Runtime Environment | 2019 |
| P36 | Aske and Zhao | Supporting Multi-Provider Serverless Computing on the Edge | 2018 |
| P37 | Back and Andrikopoulos | Using a Microbenchmark to Compare Function as a Service Solutions | 2018 |
| P38 | Boza et al. | Reserved, on demand or serverless: Model-based simulations for cloud budget planning | 2017 |
| P39 | Chang and Fink | Visualizing serverless cloud application logs for program understanding | 2017 |
| P40 | Elgamal | Costless: Optimizing Cost of Serverless Computing through Function Fusion and Placement | 2018 |
| P41 | Figiela et al. | Performance evaluation of heterogeneous cloud functions | 2018 |
| P42 | Horovitz et al. | FaaStest - Machine Learning Based Cost and Performance FaaS Optimization | 2019 |
| P43 | Ivanov and Smolander | Implementation of a DevOps Pipeline for Serverless Applications | 2018 |
| P44 | Jackson and Clynch | An Investigation of the Impact of Language Runtime on the Performance and Cost of Serverless Functions | 2018 |
| P45 | Jangda et al. | Formal Foundations of Serverless Computing | 2019 |
| P46 | Klimovic et al. | Pocket: Elastic Ephemeral Storage for Serverless Analytics | 2018 |
| P47 | Kritikos and Skrzypek | Towards an Optimized, Cloud-Agnostic Deployment of Hybrid Applications | 2019 |
| P48 | Kuhlenkamp and Klems | Costradamus: A Cost-Tracing System for Cloud-Based Software Services | 2017 |
| P49 | Leitner et al. | Modelling and managing deployment costs of microservice-based cloud applications | 2016 |
| P50 | Lin et al. | Tracing Function Dependencies across Clouds | 2018 |
| P51 | Lin et al. | Tracking Causal Order in AWS Lambda Applications | 2018 |
| P52 | Manner et al. | Cold Start Influencing Factors in Function as a Service | 2018 |
| P53 | Manner et al. | Troubleshooting Serverless functions: a combined monitoring and debugging approach | 2019 |
| P54 | Moczurad and Malawski | Visual-Textual Framework for Serverless Computation: A Luna Language Approach | 2018 |
| P55 | Nadgowda et al. | The Less Server Architecture for Cloud Functions | 2017 |
| P56 | Pellegrini et al. | Function-as-a-Service Benchmarking Framework | 2019 |
| P57 | Pérez et al. | Serverless computing for container-based architectures | 2018 |
| P58 | Pinto et al. | Dynamic Allocation of Serverless Functions in IoT Environments | 2018 |
| P59 | Spillner | Transformation of Python Applications into Function-as-a-Service Deployments | 2017 |
| P60 | Spillner and Dorodko | Java Code Analysis and Transformation into AWS Lambda Functions | 2017 |
| P61 | Winzinger and Wirtz | Model-based analysis of serverless applications | 2019 |
| P62 | Wurster et al. | Modeling and Automated Deployment of Serverless Applications Using TOSCA | 2018 |

# REFERENCES


[1] Nuha Alshuqayran, Nour Ali, and Roger Evans. 2016. A systematic mapping study in microservice architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE, 44–51.

[2] Amazon Web Services, Inc. 2019. AWS Lambda. https://aws.amazon.com/lambda

[3] Amazon Web Services, Inc. 2019. AWS Lambda Releases History. https://docs.aws.amazon.com/lambda/latest/dg/history.html

[4] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, et al. 2017. Serverless computing: Current trends and open problems. In *Research Advances in Cloud Computing*. Springer, 1–20.

[5] Alexander Bergmayr et al. 2018. A Systematic Review of Cloud Modeling Languages. *ACM Comput. Surv.* 51, 1, Article 22 (Feb. 2018), 38 pages. https://doi.org/10.1145/3150227

[6] Paolo Di Francesco, Patricia Lago, and Ivano Malavolta. 2018. Migrating towards microservice architectures: an industrial survey. In *2018 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 29–2909.

[7] Paolo Di Francesco, Patricia Lago, and Ivano Malavolta. 2019. Architecting with microservices: A systematic mapping study. *Journal of Systems and Software* 150 (2019), 77–97.

[8] Paolo Di Francesco, Ivano Malavolta, and Patricia Lago. 2017. Research on architecting microservices: trends, focus, and potential for industrial adoption. In *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 21–30.

[9] Abdessalam Elhabbash, Faiza Samreen, James Hadley, and Yehia Elkhatib. 2019. Cloud Brokerage: A Systematic Survey. *ACM Computing Surveys (CSUR)* 51, 6 (2019), 119.

[10] Geoffrey C Fox, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. 2017. Status of serverless computing and function-as-a-service (faas) in industry and research. *arXiv preprint arXiv:1708.08028* (2017).

[11] P. García López, M. Sánchez-Artigas, G. París, D. Barcelona Pons, Á. Ruiz Ollobarren, and D. Arroyo Pinto. 2018. Comparison of FaaS Orchestration Systems. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. 148–153. https://doi.org/10.1109/UCC-Companion.2018.00049

[12] Javad Ghofrani and Daniel Lübke. 2018. Challenges of Microservices Architecture: A Survey on the State of the Practice.. In *ZEUS*. 1–8.

[13] Joseph M Hellerstein, Jose Faleiro, Joseph E Gonzalez, Johann Schleier-Smith, Vikram Sreekanti, Alexey Tumanov, and Chenggang Wu. 2018. Serverless computing: One step forward, two steps back. *arXiv preprint arXiv:1812.03651* (2018).

[14] Barbara Kitchenham and Pearl Brereton. 2013. A systematic review of systematic review process research in software engineering. *Information and software technology* 55, 12 (2013), 2049–2075.

[15] H. Lee, K. Satyam, and G. Fox. 2018. Evaluation of Production Serverless Computing Environments. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. 442–450.

[16] 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.

[17] T. Lynn, P. Rosati, A. Lejeune, and V. Emeakaroha. 2017. A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms. In *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. 162–169.

[18] Peter M. Mell and Timothy Grance. 2011. *SP 800-145. The NIST Definition of Cloud Computing*. Technical Report. Gaithersburg, MD, United States.

[19] Microsoft. 2019. Microsoft Azure Functions. https://azure.microsoft.com/en-us/services/functions

[20] OpenFaaS Ltd. 2019. OpenFaaS. https://www.openfaas.com

[21] Claus Pahl and Pooyan Jamshidi. 2016. Microservices: A Systematic Mapping Study.. In *CLOSER (1)*. 137–146.

[22] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic mapping studies in software engineering.. In *Ease*, Vol. 8. 68–77.

[23] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64 (2015), 1–18.

[24] Mubashra Sadaqat, Ricardo Colomo-Palacios, and Lars Emil Skrimstad Knudsen. 2018. Serverless computing: a multivocal literature review. (2018).

[25] The Apache Software Foundation. 2019. Apache OpenWhisk. https://openwhisk.apache.org

[26] Erwin van Eyk, Alexandru Iosup, Cristina L. Abad, Johannes Grohmann, and Simon Eismann. 2018. A SPEC RG Cloud Group's Vision on the Performance Challenges of FaaS Cloud Architectures. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE '18)*. ACM, New York, NY, USA, 21–24.

[27] Erwin van Eyk, Alexandru Iosup, Simon Seif, and Markus Thömmes. 2017. The SPEC Cloud Group's Research Vision on FaaS and Serverless Architectures. In *Proceedings of the 2Nd International Workshop on Serverless Computing (WoSC '17)*. ACM, New York, NY, USA, 1–4.

[28] Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. 2006. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements engineering* 11, 1 (2006), 102–107.

[29] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. Citeseer, 38.

[30] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.

[31] WoSC. 2017. First International Workshop on Serverless Computing (WoSC). https://www.serverlesscomputing.org/wosc17

[32] WoSC. 2018. Fourth International Workshop on Serverless Computing (WoSC). https://www.serverlesscomputing.org/wosc4

[33] WoSC. 2018. Third International Workshop on Serverless Computing (WoSC). https://www.serverlesscomputing.org/wosc3

[34] Vladimir Yussupov, Uwe Breitenbücher, Frank Leymann, and Michael Wurster. 2019. *Systematic mapping study on engineering FaaS platforms and tools*. https://doi.org/10.5281/zenodo.3520163