Function-as-a-Service for the Cloud-to-Thing continuum: a Systematic Mapping Study

Bárbara da Silva Oliveira[1][a], Nicolas Ferry[1][b], Hui Song[2][c], Rustem Dautov[2][d], Ankica Barišić[1][e], and Atslands Rego da Rocha [3][f]

[1]*Université Côte d'Azur, I3S/INRIA Kairos, Sophia Antipolis, France*

[2]*SINTEF Digital, Oslo, Norway*

[3]*Departamento de Engenharia de Teleinformática, Universidade Federal do Ceará, Brasil*

{*barbara.da-silva-oliveira, nicolas.ferry, ankica.barisic*}*@inria.fr,* {*hui.song, rustem.dautov*}*@sintef.no, astlands@ufc.br*

Keywords: Internet of Things; Cloud; Cloud-to-Thing Continuum; Function-as-a-Service; Systematic Mapping Study

Abstract: Until recently, Internet of Things applications were mainly seen as a means to gather sensor data for further processing in the Cloud. Nowadays, with the advent of Edge and Fog Computing, digital services are dragged closer to the physical world, with data processing and storage tasks distributed across the whole Cloud-to-Thing continuum. Function-as-a-Service (FaaS) is gaining momentum as one of the promising programming models for such digital services. This work investigates the current research landscape of applying FaaS over the Cloud-to-Thing continuum. In particular, we investigate the support offered by existing FaaS platforms for the deployment, placement, orchestration, and execution of functions across the whole continuum using the Systematic Mapping Study methodology. We selected 33 primary studies and analyzed their data, bringing a broad view on the current research landscape in the area.

1 INTRODUCTION

Nowadays, with the advent of Edge and Fog Computing, digital services are dragged closer to the physical world, with data processing tasks distributed across the whole Cloud-to-Thing continuum. Function-as-a-Service (FaaS) is gaining momentum as one of the promising programming models for such digital services. It allows developers to focus on the software development while the Cloud provider manages the underlying infrastructure. Serverless functions are simple operations expressing parts of the application logic, which are triggered on the occurrence of specific events. This event-driven nature is a natural fit for IoT event and data processing (Cheng et al., 2019).

However, challenges hinder the broader adoption of FaaS solutions for the whole Cloud-to-Thing continuum, hereafter called FaaS4C2T for short. For example, in contrast to classical Cloud infrastructures, Edge and Thing infrastructures are typically largely heterogeneous, including devices such as gateways, base stations, and tiny microcontrollers, with a wide geographical distribution. Serverless functions typically need to be deployed close to the data source and tailored to their host's cyber-physical context, which typically cannot be done homogeneously using standard deployment solutions (*e.g.*, not all devices can support the same virtualization technology).

Several platforms such as PAPS (Baresi and Quattrocchi, 2021) and AuctionWhisk (Bermbach et al., 2022) have emerged to enable FaaS on Edge infrastructures. However, there needs to be more understanding of their coverage and support of the IoT end (*i.e.*, the Thing layer). Consequently, there is an urgent need to investigate the diverse features and properties of current FaaS4C2T platforms. In this context, we conducted a Systematic Mapping Study to provide a holistic, yet concise overview of the FaaS4C2T landscape and the existing support for the whole Cloud-to-Thing continuum, including research gaps and limitations. The main contributions of this work are the answers to the following Research Questions (RQs):

- **RQ1:** What are the research publication trends in

[a] https://orcid.org/0009-0007-8455-9627
[b] https://orcid.org/0000-0003-2036-0508
[c] https://orcid.org/0000-0002-9748-8086
[d] https://orcid.org/0000-0002-0260-6343
[e] https://orcid.org/0000-0001-7513-7907
[f] https://orcid.org/0000-0002-3069-132X

the domain of FaaS platforms for the Cloud-to-Thing Continuum?

- **RQ2:** What are the specificities and limitations of FaaS platforms for the Cloud-to-Thing continuum?
- **RQ3:** What are the open research issues to be further investigated in this field?

We have systematically processed an extensive number of relevant papers from four online publication databases to finally obtain a set of 33 primary studies from which we extracted and synthesized data to answer our RQs. The results show gaps in research (*e.g.*, lack of customizable function allocation strategies, managing functions' concurrent accesses to shared resources) that need to be addressed to enable proper application of the FaaS programming model to the Cloud-to-Thing continuum.

The remainder of the paper is organized as follows. Section 2 introduces some background definitions. Section 3 details the methodology adopted. Section 4 presents the results and the analysis of the data extraction. Section 5 presents related work and Section 6 concludes our work, presenting and discussing the main findings.

2 BACKGROUND

In this section, we describe the main concepts and definitions adopted in this paper.

2.1 Cloud-to-Thing Continuum

The Cloud-to-Thing continuum refers to the extension of the Cloud capabilities towards low-end devices, *i.e.*, Things. This work considers the Cloud-to-Thing continuum composed of four main layers: Cloud, Intermediary, Edge and Things layers. The Intermediary and Edge layers can also be grouped and referred to as Fog, according to the definition of the OpenFog consortium (Byers and Swanson, 2017). There is still no consensus about the definition of Edge and Fog computing; in some cases, Edge and Fog can be used interchangeably.

According to the National Institute of Standards and Technology (NIST), Cloud computing is a computing model enabling ubiquitous network access to a shared and virtualized pool of computing capabilities (*e.g.*, network, storage, processing, and memory) that can be rapidly provisioned with minimal management effort (Mell and Grance, 2001). IoT systems typically exploit the **Cloud layer** for centralized and resource-demanding data processing and storage.

The Open Glossary from the Linux Foundation defines Edge computing as *"the delivery of computing capabilities to the logical extremes of a network in order to improve the performance, operating cost and reliability of applications and services. By shortening the distance between devices and the cloud resources that serve them and reducing network hops, edge computing mitigates the latency and bandwidth constraints of today's Internet, ushering in new classes of applications"* (LF Edge, 2019). Conceptually, Edge computing is closer to the bottom IoT layer, rather than to Cloud-based services (Mahmud et al., 2018). The **Edge layer** thereby consists of Edge devices such as routers, gateways, base stations, etc. This missing link between Edge and Cloud is often implemented as an extra layer, which we denote as **Intermediary Layer**, following the Fog computing architecture definition by the OpenFog consortium (Byers and Swanson, 2017). It is typically composed of Fog servers or lightweight cloud servers such as Cloudlets (Pang et al., 2015). The **Intermediary** and **Edge** layers are often grouped as a single **Fog layer**.

The **Thing layer** includes low-level leaf nodes within IoT systems and is defined as *"physical objects that are capable of sensing or acting on their environment and able to communicate with each other"*[1]. Things cannot always provide virtualization support and might not be directly connected to the Internet (*i.e.*, Internet connection is granted via other devices).

2.2 Serverless and Function-as-a-Service

As explained in (Baldini et al., 2017), *"Serverless describes a programming model and architecture where small code snippets are executed in the Cloud without any control over the resources on which the code runs"*. The Serverless programming model stands closer to the Platform-as-a-Service (PaaS) model, where developers have access to a prepared runtime software stack maintained by the Cloud provider and focus on writing application code, rather than to the Infrastructure-as-a-Service (IaaS) model where the developer has full control and is accountable for managing the application and its runtime software stack. However, in contrast to PaaS, serverless applications are meant to be scalable by design, and developers expect the operation of these applications to be fault-tolerant and quickly auto-scalable (Xie et al., 2021). Serverless applications are often wrapped in software containers as a lightweight virtualization solution that can be easily and rapidly provisioned. (Baldini et al.,

[1] As defined by the EU: https://www.europarl.europa.eu/RegData/etudes/BRIE/2015/557012/EPRS_BRI(2015)557012_EN.pdf

2017) defines a Serverless platform as *"an event processing system. The service must manage a set of user-defined functions, take an event sent over HTTP or received from an event source, determine which function(s) to dispatch the event, find an existing instance of the function or create a new instance, send the event to the function instance, wait for a response, gather execution logs, make the response available to the user, and stop the function when it is no longer needed"*. This implies that Serverless platforms include at least an event broker as well as a solution to dispatch the events and manage the service life-cycle.

FaaS is a form of Serverless computing in which the Cloud provider manages the resources, life-cycle, and event-driven execution of user-provided functions (Van Eyk et al., 2018), denoted as serverless functions in the rest of the paper.

2.3 Deployment and Orchestration

To **deploy** an application on a selected target environment, its components need to be **placed** to hosting resources. More precisely, the implementation of those components needs to be placed and deployed. The notion of a deployable artifact supports exactly the reference between logical components and connectors to their implementations (Bergmayr et al., 2018). **Placement** selects the resources on which the deployable artifacts will be hosted, and a **placement strategy** defines how to drive this selection. **Deployment** refers to the action of provisioning (or de-provisioning) the resources (when possible), installing, configuring, and starting (or stopping) deployable artefacts. How to enact a deployment is specified in a **deployment model**.

The NIST defines **orchestration** as *"the sequence and conditions in which one Web service invokes other Web services to realize some useful function"*. In the FaaS context, orchestration involves management and coordination of serverless functions. An **orchestration model** provides a programmatic way to combine simple, well-defined serverless functions into more complex applications (Bocci et al., 2021).

3 REVIEW METHODOLOGY

This section describes the adopted review methodology, which is based on the established guidelines from (Kitchenham et al., 2011; Kitchenham and Charters, 2007; Petersen et al., 2015).

3.1 Research Questions

This study investigates ***how FaaS platforms and associated mechanisms support the Cloud-to-Thing continuum***. To this end, we classify the main characteristics and capabilities of existing platforms, including the toolset they offer, and we identify the gaps and future research directions. The overall objective is underpinned by the three RQs listed in Section 1.

As part of **RQ1**, we intend to understand the current status of the research, which includes general statistics, such as **RQ1.1 -** the number of studies published by year and the venue type; **RQ1.2 -** the level of maturity of the research; **RQ1.3 -** the focus of the research (*e.g.*, Deployment, Placement); **RQ1.4 -** what FaaS platforms are frequently used for the Cloud, Fog, and Thing layers? With **RQ2**, we want to understand the mechanisms and properties of FaaS for the Cloud-to-Thing continuum and how they differentiate from FaaS platforms for the Cloud. This includes the specific subquestions: **RQ2.1 -** what are the platforms' architectural patterns; **RQ2.2 -** on which layers of the Cloud-to-Thing continuum serverless functions can be deployed; **RQ2.3 -** what are the mechanisms used to deploy, package and run serverless functions in isolation; **RQ2.4 -** what mechanisms are responsible for the serverless function placement and scalability? **RQ3** also includes two sub-questions. **RQ3.1 -** What are the open issues in FaaS4C2T research? **RQ3.2 -** What research directions can be recommended for tackling the open issues? These subquestions suggest potential future work directions.

3.2 Inclusion and Exclusion Criteria

The Inclusion (ICs) and Exclusion Criteria (ECs) applied in this systematic mapping are listed in Table 1.

3.3 Selection Strategy

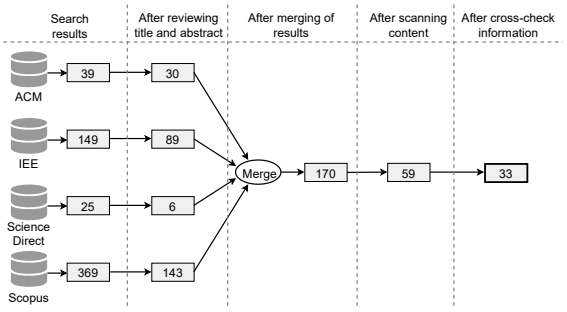


Figure 1: Study selection process.

Using search engines of online publication

Table 1: Inclusion (ICx) and Exclusion (ECx) Criteria.

IC1	The study presents a FaaS platform or extends an existing FaaS platform.
IC2	FaaS is applied to the Cloud-to-Thing continuum, *i.e.*, covering not only the Cloud, but also the domains Edge, Fog, robotics, IoT, as well as their possible applications.
IC3	The study provides details about serverless function deployment, placement, or scheduling.
EC1	The study is bounded to a specific Fog and/or IoT environment, which cannot be applied in more use cases.
EC2	The paper is published before the year 2008. This year, the concepts of serverless programming and FaaS started to appear.
EC3	When a solution is reported in several papers (*e.g.*, overall approach, empirical study, evaluation), we include all these papers but treat them as a single approach.
EC4	We excluded papers not written in English, not peer-reviewed, as well as short papers (*i.e.*, only accessible as extended abstracts, posters, or presentations).
EC5	The study has a length of fewer than four pages in double-column format or six pages in single-column format.

databases is the established practice to search for essential primary studies, as well as follow-up supplemental studies (Kitchenham and Charters, 2007). We used the following four popular publication databases: IEEE Xplore[2], ACM Digital Library[3], ScienceDirect[4], and Scopus[5]. Scopus and ACM DL already index SpringerLink[6] (Tran et al., 2017). Following the guidelines from (Kitchenham and Charters, 2007), we have defined our search terms based on the RQs and the keywords used in related articles. The search query was adjusted to fit the search engines of the four publication databases.

Listing 1: Query string used to perform the database search.

```
TITLE-ABS ("Serverless" OR "FaaS" OR
    "Function as a Service" OR
    "Function-as-a-Service")
AND
TITLE-ABS ("IoT" OR "Internet of Things" OR
    "Things" OR "Edge" OR "Fog" OR "CPS" OR
    "Cyber*Physical System*" OR "Robot*"))
AND
PUBYEAR > 2008
```

[2]https://ieeexplore.ieee.org
[3]https://dlnext.acm.org
[4]https://sciencedirect.com/
[5]https://scopus.com
[6]https://www.springer.com

The study selection process is illustrated in Figure 1. First, we searched the databases using the defined query string. Next, for every candidate paper, we reviewed the title and abstract, and long-listed papers according to the ICs and ECs. Since an article may appear in more than one database, we merged the results, thus removing duplicates. After that, we short-listed the studies based on full texts, again using the the ICs and ECs. To do this, the papers were distributed among the contributing authors, who had to specify which ICs and ECs they considered for short-listing the papers. For quality assurance purposes, we also performed another round of peer review to reach the consensus on the short-listing decisions. When there were conflicts in judgments between the reviewers, we discussed the decisions to reach the final agreement.

3.4 Comparison Framework

The review framework contemplates the essential aspects of the FaaS4C2T presented in the literature. In particular, we emphasized characteristics regarding the deployment and placement of serverless functions, taking into account the differences between only involving the Cloud and other layers within the Cloud-to-Thing continuum. The primary purpose of the framework is to extract and compare data from the primary studies so that they can help to answer the RQs. To develop a comprehensive review framework, we extensively explored and discussed the core concepts in the field of FaaS when applied to the Cloud-to-Thing continuum. The resulting framework is documented in two categories: Architecture and Runtime Capabilities.

3.4.1 Architecture

This category focuses on the architecture of the FaaS platforms. We survey the components that form the core of the FaaS platforms and the typical architectural patterns they adopt when applied to the Cloud-to-Thing continuum.

- *Platform distribution*: All the software components that form a FaaS platform may be centralized or some of them may be distributed. We specifically investigate if these components can, or need to, be deployed on Fog and Things layers.
- *Multi-Tenancy*: When multiple independent tenants operate in the same platform, the same instance or resources are used for distinct users.
- *Multi-Application*: When two or more different applications from the same tenant operate in the same environment.

- *Multi-Cloud*: When the platform services, functions, and software components can be deployed across multiple computing and storage services from different Cloud providers.
- *Entities for Functions Life Cycle Management*: We investigate services and components participating in the life-cycle management of serverless functions (Aslanpour et al., 2021). This includes aspects such as the monitoring, configuration, deployment, termination, etc. of serverless functions. We also aim to understand whether the management of functions on the Fog and Thing layers requires specific support.

3.4.2 Runtime Capabilities

We want here to explore the runtime capabilities offered by the FaaS platform for the management, deployment, placement of serverless functions, and mechanisms for function isolation.

Deployment Capabilities:

- *Deployable Artefact Type*: What is the type of deployable artefacts accepted by the platform (*e.g.*, the project code, the binaries, scripts).
- *Bootstrap*: Bootstrap refers to all the necessary software modules on the devices required to the function deployment. (Arcangeli et al., 2015).
- *Target Infrastructure*: We survey the layers where serverless functions can be deployed.
- *Long-lived tasks*: Serverless functions are naturally suitable for short-lived tasks. For those tasks, there is a small amount of data to be processed, and they are rapidly finished. We investigate the support for long-lived tasks, which involve substantial amounts of data, and require a longer period to be processed (Baldini et al., 2017; Aslanpour et al., 2021).

Placement:

- *Placement Strategy*: The Placement Strategy may be to choose a specific node for the placement, for example, the closest node to the IoT device. It can also be defined as an optimization problem, considering various constraints, such as energy consumption or bandwidth.
- *Dynamism*: We investigate if the placement algorithm is executed dynamically. One possibility is that the placement can be carried out by the arrival of new inputs in the system. The placement can also be triggered by changes in the infrastructure, *e.g.*, when new virtual machines are created (Mahmud et al., 2020).
- *Replication Strategy*: We investigate the strategy adopted for function replication, which plays a part in the serverless functions scalability mechanism.

Infrastructure Management:

- *Resource Monitoring*: The deployment of the serverless functions and the quantity of workload that can be employed depends on the available resources. Thus, the FaaS platform must bring mechanisms to provide the status of the device's capabilities (Costa et al., 2022), determining whether it is possible or not to execute the incoming functions and services in these devices.
- *Device Information*: The kind of device information is being considered, *e.g.*, memory, CPU, or bandwidth available in the system devices.
- *Runtime Information*: The kind of runtime information is being considered for the monitoring mechanisms. For example, it can be the status of the running processes in computing Fog nodes.

Scalability: The standard types of scaling in Cloud modeling include Horizontal and Vertical Scalability (Xie et al., 2021). Horizontal scaling is done by deploying new instances of the serverless function on existing or newly added machines in the resource pool. In Vertical scaling, more resources, such as CPUs and memory, are added to an existing server, or the function is moved to a more powerful machine.

Isolation: Isolation is a fundamental mechanism for the secure and effective execution of functions and services. It aims to separate different tenants and functions of the same application, which allows Multi-tenancy and Multi-Application use (Li et al., 2022). Traditionally, for Cloud applications, possible solutions are Virtual Machines and Containers, which can be heavy for devices in constrained scenarios in the Fog or Thing layers.

4 ANALYSIS OF THE RESULTS

Table 2 details the list of primary FaaS4C2T studies. Based on the comparison framework, we have extracted and synthesized the data from the primary studies to answer our RQs.

4.1 Answering RQ1: Publication trends

To answer RQ1, we extracted information about the publication years, venue type, author's affiliation, etc.

Table 2: Overview of the primary FaaS4C2T studies.

#	Title*	Year	v
1	Kappa: Serverless IoT Deployment	2017	C
2	Calvin Constrained — A Framework for IoT Applications in Heterogeneous Environments	2017	W
3	NFaaS: Named function as a service	2017	C
4	Lite-Service: A Framework to Build and Schedule Telecom Applications in Device, Edge and Cloud	2018	C
5	Dynamic Allocation of Serverless Functions in IoT Environments	2018	C
6	Towards a Serverless Platform for Edge Computing	2019	C
7	Towards a serverless platform for edge AI	2019	C
8	Fog Function: Serverless Fog Computing for Data Intensive IoT Services	2019	C
9	An Execution Model for Serverless Functions at the Edge	2019	C
10	A unified model for the mobile-edge-cloud continuum	2019	J
11	tinyFaaS: A Lightweight FaaS Platform for Edge Environments	2020	C
12	Device-driven On-demand Deployment of Serverless Computing Functions	2020	W
13	Allocation priority policies for serverless function-execution scheduling optimisation	2020	C
14	A NodeRED-based dashboard to deploy pipelines on top of IoT infrastructure	2020	C
15	NanoLambda: Implementing Functions as a Service at All Resource Scales for the Internet of Things	2020	C
16	Sledge: A Serverless-First, Light-Weight Wasm Runtime for the Edge	2020	C
17	μ actor: Stateful Serverless at the Edge	2021	C
18	Self-balancing architectures based on liquid functions across computing continuums	2021	C
19	Optimized container scheduling for data-intensive serverless edge computing	2021	J
20	Operating Latency Sensitive Applications on Public Serverless Edge Cloud Platforms	2021	J
21	Mu actor: Stateful Serverless at the Edge	2021	C
22	PAPS: A Serverless Platform for Edge Computing Infrastructures	2021	J
23	Colony: Parallel Functions as a Service on the Cloud-Edge Continuum	2021	C
24	Latency-Sensitive Edge/Cloud Serverless Dynamic Deployment over Telemetry-Based Packet-Optical Network	2021	J
25	In the Fog: Application Deployment for the Cloud Continuum	2021	C
26	Function delivery network: Extending serverless computing for heterogeneous platforms	2021	J
27	LaSS: Running Latency Sensitive Serverless Computations at the Edge	2021	C
28	Dyninka: A FaaS Framework for Distributed Dataflow Applications	2021	W
29	A Decentralized Framework for Serverless Edge Computing in the Internet of Things	2021	J
30	Real-Time FaaS: Towards a Latency Bounded Serverless Cloud	2022	J
31	Light weight serverless computing at fog nodes for internet of things systems	2022	J
32	AuctionWhisk: Using an auction-inspired approach for function placement in serverless fog platforms	2022	J
33	Towards Efficient Processing of Latency-Sensitive Serverless DAGs at the Edge	2022	W

[v] Venue type: J = Journal (19), C = Conference (37), W = Workshop (13).
* The titles are clickable to link to the corresponding publications

Publication Years and Venue type. Figure 2 depicts the publication years according to the venue type. We can observe a growth in the interest in the FaaS4C2T research area. The earliest relevant study was published in 2017. The years 2020 and 2021 show a rise in the number of publications. This demonstrates the need for research on FaaS4C2T and the research area is gaining more attention. In the last year, we collected fewer publications since we did the database search of papers on April 2022.

From Figure 2, we also observe that conference paper is the most popular publication type. The journal articles have a lower percentage and started appearing in 2019. It is worth noting that the high number of journal articles in 2021 might be partly due to the COVID-19 pandemic. Overall, this reflects the paper's maturity level in the field, which is still in its early stages.

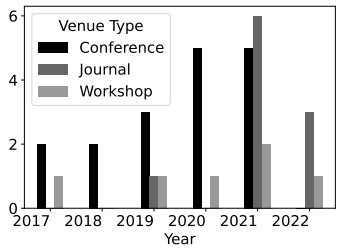


Figure 2: Distribution of publication types per year.

Author's Affiliation: Figure 3 depicts the author's affiliation. Most authors are from academia, followed by a collaboration of academia and industry. There are few studies conducted by the industry, probably due to the novelty of this research area.

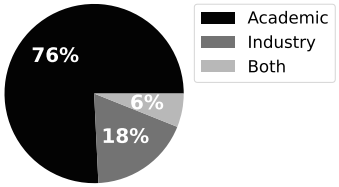


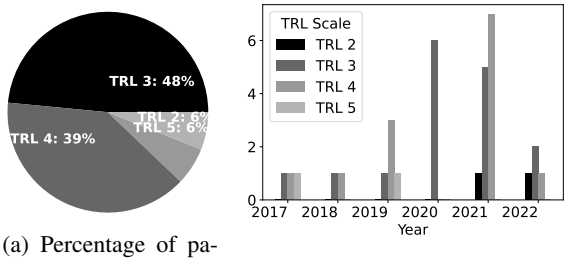
Figure 3: Author's affiliation.

Case study for evaluation: Regarding the type of case study used for evaluating the FaaS4C2T approaches, to the best of our understanding, all case studies were defined by academics and not by industrial such as motivational examples, prototypes, or simulations developed by researchers for discussing or evaluating. Finally, it is nice to see that only two primary studies do not provide any evaluation details. In the future, we expect to see more industry-driven case studies as the maturity of the research increase.

Technology Readiness Levels: The Technology Readiness Levels (TRL) is a concept used to determine a given technology's maturity level. We used the TRL scale defined by the EC[7] to determine the maturity of the FaaS4C2T platforms. Figure 4a depicts the TRL of our primary studies. We did not find any work at TRL 1 and only a few at TRL 2, as such work is more difficult to be accepted by conferences and journals. Most of the FaaS4C2T platforms are in the early stages of TRL 3, and 4 whilst

[7] https://ec.europa.eu/research/participants/data/ref/h2020/wp/2014_2015/annexes/h2020-wp1415-annex-g-trl_en.pdf

only a few reach TRL 5. The works at TRL 2 present technology concepts without connecting components or performing many experiments. At TRL 3, we typically find experimental studies with at most prototypes and some evaluation metrics. At TRL 4, results are validated in labs typically with small use case environments. The works at stage 5 are more advanced, implementing the FaaS platforms in large-scale environments and applications in more advanced projects. This represents that we are still distant from the actual implementation of the platform in an operational environment. Figure 4b represents the distribution of the works according to year and scale. We note that more works in stages 3-5 started to appear in the last three years, which corroborates our presumption that this application area is essential and with growing interest and development.



(a) Percentage of papers per TRL (b) TRL per year

Figure 4: Analysis of the TRLs.

Main Focus of the Primary Studies: Figure 5 depicts the main focus of the selected studies. Given that all studies are about FaaS platforms for the Cloud-to-Thing continuum (see ICs and ECs), we evaluated whether their contribution was primarily on serverless functions deployment, placement, or orchestration. Indeed, serverless functions placement and deployment are critical for the proper execution of an orchestration. Placement is the main topic for most primary studies as classical serverless function placement strategies in Cloud infrastructures are, in most cases, impractical in the Cloud-to-Thing continuum. Indeed, in the later cases, the placement strategies must consider *(i)* the high heterogeneity of the infrastructure - *i.e.*, resources have largely different capabilities and functions must be placed accordingly, as well as *(ii)* the geographical location and distribution of the devices at the Edge and Thing layers - *i.e.*, functions must be placed and replicated in the vicinity of all sensors of a certain type.

Original or extension of existing FaaS Platforms: As depicted in Figure 6, there is a good balance between the studies that developed their original FaaS

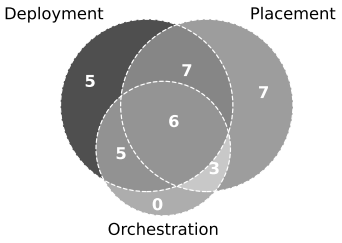


Figure 5: Main focus of the primary studies.

platform and studies extending existing ones. In the later cases, the studies proposed modifications, rules, and mechanisms for improving widely used platforms. OpenWhisk is the most preferred among the open-source platforms, since OpenWhisk was one of the first comprehensive open-source FaaS platforms. For example, Bermbach et al. developed *AuctionWhisk*, (Bermbach et al., 2022), and propose a placement mechanism on top of OpenWhisk. For AWS Lambda, (Pelle et al., 2021) propose including a service that provides real-time evaluations of the system and dynamic modification of placement decisions based on current performance. There are studies extending OpenFaaS and one based on OpenStack. It is also worth mentioning that not all platforms offer the same features. In particular, some studies leverage Kubernetes and KNative, which are not pure FaaS platforms but rather advanced container management systems on top of which solutions such as OpenWhisk and OpenFaaS can be built.

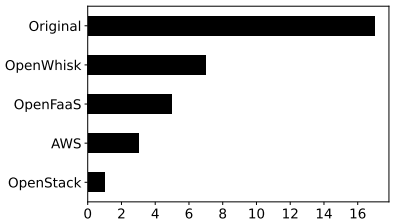


Figure 6: FaaS platforms used.

4.2 Answering RQ2: Support for the Cloud-to-Thing continuum

To investigate the peculiarities and limits of existing FaaS platforms when applied to the Cloud-to-Thing continuum we analyzed the primary studies using the comparison framework defined in Section 3.4.

How are the FaaS platforms and the application using the platforms architectured? Figure 7 indicates on which layers of the Cloud-to-Thing continuum the FaaS4C2T platforms components are typically distributed. In most primary studies, the plat-

form's components are located at the Fog and Cloud layers. (Pelle et al., 2021) implemented a system with FaaS platform components responsible for Deployment and Data Storage running both on the Edge and Cloud Layers. In (Elkholy and Marzok, 2022), some Manager and Working nodes at the Fog layer are responsible for the serverless function life-cycle management and execution, whilst at the Cloud layer, a Working node offers support for executing offloaded tasks from the Fog. In general, Cloud resources are privileged for resource demanding tasks and Fog resources when low latency is required. We found studies (24%) that only considered the Fog layer. In (Mittal et al., 2021), an extension of the KNative Platform, all the components (*i.e.*, Load Balancer, Auto-scaler, Placement Engine, and other standards modules from KNative) are located in the Fog Layer. Lastly, we identified studies that also included the Things layer (15%), adding to this layer component related to the execution of functions. Overall, it is natural that the FaaS platform, besides supporting the deployment of applications across the Cloud-to-Thing continuum, also exploits the benefit of the Fog layer. Among others, the main benefits exhibited in the primary studies are reliability, reduced latency, and the ability to process data close to the source.

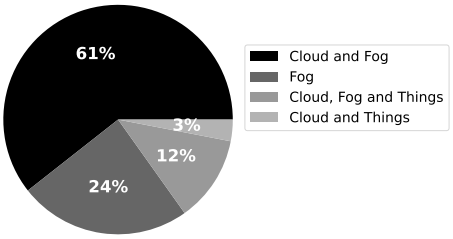


Figure 7: FaaS platform distribution.

Serverless Function Deployment Location: As depicted in Figure 8, most primary studies (66%) support deployment of serverless functions over Cloud and Fog infrastructures whilst 19% support deployment on Fog infrastructures only. A total of 15% of the FaaS4C2T approaches offer support for deployment on the Thing layer, in combination with Cloud and Fog layers or not. The deployment of function on Things is only motivated by a few use cases, typically to prepare and preprocess the data in the Thing before sending it to the other layers. An example is when a sensor must detect changes in an environment (*e.g.*, humidity in a greenhouse) for management decisions (*e.g.*, irrigating the plantation). It is inefficient that the sensor sends all the data it detects (*e.g.*, insignificant changes in the humidity indicator) for the Fog or Cloud Layer to be processed. A solution would be to execute a function on the IoT device or close to it, which classifies if the data is relevant. This strategy could reduce bandwidth in the network and energy consumption. Furthermore, in some cases, when the Internet connection is unstable, deploying functions on IoT devices would prevent the interruption of tasks. (Persson and Angelsmark, 2017) add that the use of sensors is a continuous process, implying the necessity of a continuous execution of functions, the so called long-lived functions.

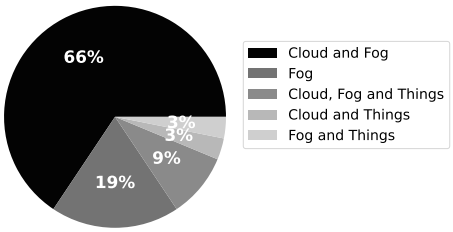


Figure 8: Function deployment location.

What are the mechanisms offered for functions virtualization, deployment and isolation? As depicted in Figure 9, most of the FaaS4C2T approaches leverage virtualization techniques such as Docker containers, WebAssembly, and other lightweight virtualization techniques to wrap and deploy serverless functions. The isolation offered by these techniques is essential in multi-tenancy and multi-application scenarios. Only (4/33) studies rely on original solutions. (Persson and Angelsmark, 2017) developed the *Kappa* platform, in which they leverage the concept of *actors*, deployment units of a single operation or data processing. For supporting the deployment in IoT devices, (George et al., 2020) propose isolating different serverless functions by running them in separate instances of Python Virtual Machines, a solution similar to traditional Linux containers. In (Hall and Ramachandran, 2019), the authors use WebAssembly, which has an inherent strong, yet lightweight, sandboxed environment for function execution.

Sometimes functions are designed to interact with other software and hardware resources (in some cases directly as part of the function code – *e.g.*, a function retrieving data from a sensor; or indirectly – *e.g.*, messages being sent to a broker before reaching a software service). At the Fog and Thing layers, this includes hardware such as sensors and actuators, which cannot be embedded as part of the virtualized environment. The management of such shared resources is of tremendous importance, ranging from business to safety critical in the case of actuators. (Hall and Ramachandran, 2019), partly consider this issue by

defining different access to function patterns, including a case of multiple clients accessing one serverless function. As an example, they consider the case of multiple smart cameras (sensors), each belonging to a different client requesting image processing to the same serverless function. We did not find any primary study tackling the opposite case where several functions need to concurrently access a shared resource such as an actuator.

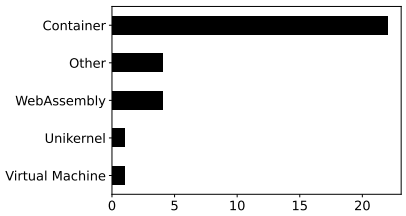


Figure 9: Isolation techniques.

How serverless functions are placed and scaled? We observed that placement strategy is a key research topic in the field as the extension of the FaaS to the Cloud-to-Thing continuum calls for novel strategies better exploiting the specificities of the Fog and Thing infrastructures – *i.e.*, heterogeneity, limited computing resources, and geographical distribution of devices. In particular, we identified that most of the proposed placement strategies consider the availability of resources and device capacity (*e.g.*, CPU, memory, bandwidth) as key parameters to decide whether a serverless function can be deployed on a device.

(Pelle et al., 2021) propose considering the data from the system monitoring for modifying placement decisions. A monitoring module computes network statistics, infrastructure status, and function deployment information (*e.g.*, execution time and invocation rates). According to this data, there is a reconfiguration of the function placement after each deployment. Another study that considers function information is (Elkholy and Marzok, 2022). The functions related to real-time processing are placed first and executed in nodes of greater capacity than others.

Only a few primary studies allow developers to customize the placement strategy parameters. In (Bermbach et al., 2022), the authors propose using bids, attached with the functions to be executed. Those bids are composed of two parameters, the price for the function's execution and a customized parameter, in which the developers determine the price they are willing to pay for storing the function's executable. In (Pelle et al., 2020), there is support for developers to send as input data constraints for the placement, such as latency and critical (graph) paths, among other conditions. (Mittal et al., 2021) focus on placing functions to avoid container fragmentation and provide fairness among multiple tenants, whereas (Cicconetti et al., 2020) implement flexible placement, which can incorporate different algorithms to fit specific use cases based on the developer's choice.

Regarding scalability, we found that (21/33) of the FaaS4C2T platforms mention support for scalability. More specific to the Cloud-to-Thing continuum, we noted large support for offloading techniques (16/21) among the (21/33) FaaS4C2T platforms that mention scaling mechanisms. Offloading either refers to forwarding the function from a local layer to other devices in the same layer (*e.g.*, between Fog devices), commonly referred to as horizontal offloading, or to devices in different layers (*e.g.*, from the Fog Layer to the Cloud), designated as vertical offloading. Figure 10 depicts the types of offloading techniques considered by the primary studies. (Baresi et al., 2019) claims that this is an efficient way of improving users' Quality of Experience when facing a computation-intensive and latency-sensitive application. Computing devices at the Fog and Thing layers may experience some degradation. Hence, offloading tasks to another Fog device or server in the Cloud can prevent service interruptions. This conclusion is also supported by (Lordan et al., 2021), which adds that it is also beneficial for workload balancing, resulting in less power consumption and time response of applications. (George et al., 2020) is an example of FaaS4C2T platform supporting vertical offloading, which proposes an algorithm for estimating when better latency can be achieved by leveraging execution from the Things Layer to the Edge or Cloud Layer. (Hetzel et al., 2021) is an example of FaaS4C2T platform supporting horizontal offloading, distributing the workload on the Edge devices to overcome resource limitations.

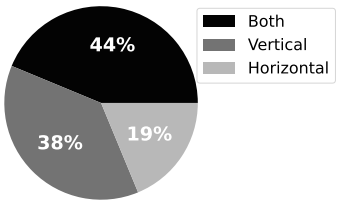


Figure 10: Offloading technique.

4.3 Answering RQ3: Open issues

In the following, we discuss open issues we believe should be further investigated. We try to illustrate some of these issues and suggest research directions.

Regarding the support offered by existing FaaS

platforms for the deployment of serverless functions over the whole Cloud-to-Thing continuum, we identified that only a few primary studies provide means to deploy serverless functions on devices at the Thing layer. These approaches are still far from offering the same features are Cloud FaaS platforms (*e.g.*, in terms of isolation and replication). We also observe the lack of proper use cases, and in particular industrial use cases, that motivate and clearly describe the benefits this would bring compared to more deployment on solely Cloud and Fog layers, where the Fog layer is typically used as a means to integrate Things.

Serverless functions often need to be deployed together with other services and software components such as message brokers and data stores. Software systems, including Edge and IoT systems, are rarely built entirely from serverless components, but rather follow a hybrid approach where functions are combined with other types of software components (Ferry et al., 2022). When dealing with systems deployed over the whole Cloud-to-Thing continuum, these supporting services typically need to be deployed in the vicinity of the function, close to the data source (*e.g.*, on the same edge device). To the best of our knowledge, there is little support and freedom offered to developers for deploying and customizing such services as they deploy serverless functions (*i.e.*, existing platforms come with a fixed and limited set of services and do not support the deployment of supporting services together with the functions).

We found that there can be limits to isolation and virtualization as sometimes resources cannot be virtualized and must be shared between tenants and functions. This is particularly the case at the Fog and Things layers where the function might want to concurrently access hardware such as actuators (*e.g.*, GPIO on a Raspberry Pi) or accelerators (*e.g.*, Coral.ai USB accelerator), which eventually cannot be virtualized. Since actuators can impact the physical environment, managing access to these shared resources is essential to guarantee the trustworthiness of the system, including safety, security, and performance aspects. To the best of our knowledge, there are currently no proper solutions to this issue. A few studies proposed mechanisms tackling this aspect for shared access to software components. (Jang et al., 2021) developed an architecture with a sharing deployment module in the same edge device, in which two or more functions are deployed in the same container. For this scenario, the authors considered in the platform design a control plane, with customized options for the platform users, and a data plane, guaranteeing the deployment of functions in the module by controlled queue requests.

Regarding the placement of serverless functions, while our study shows that this topic is gaining interest in the research community, we believe there are still several areas for improvement. In particular, most of the work focuses on allocation strategies based on resource (*e.g.*, CPU, Memory, network) usage optimization. While a few studies also consider aspects such as the geographical location of Fog and Things, there is still room for greater consideration of devices' context. We also found that most strategies are static and cannot be easily customized or even specified by end-users. We believe such a feature would be highly beneficial and work in this direction is required (including the definition of a tool supported domain specific languages for end-user to specify their allocation strategies).

5 RELATED WORK

In recent years, several surveys were conducted to investigate different aspects of FaaS platforms for the Cloud. For instance, (Taibi et al., 2020) focuses on patterns for serverless functions. There have also been some recent surveys focusing on some aspects of serverless and infrastructures within the Cloud-to-Thing continuum. Some provide an overview whilst others focus on specific topics; however, to the best of our knowledge, no single comprehensive review answers our RQs.

Closest to our work, (Cassel et al., 2022) presents a systematic literature review about Serverless computing for the Internet of Things. In particular, this study investigates on which layer (Cloud, Fog, Edge) serverless functions are usually placed and provides an overview of the landscape of runtime used to execute these functions. We distinguish from this work by extending this investigation to the whole Cloud-to-Thing continuum, thereby considering the placement of serverless functions on Things. In addition, we also investigate how the placement, and the actual deployment of the serverless functions are performed. (Bocci et al., 2021) explores the secure orchestration of serverless functions in Fog computing environments. This work complements our investigation to understand how developers can specify how to deploy serverless functions. However, by contrast, we consider the whole Cloud-to-Thing continuum and cover other aspects such as allocation mechanisms, and scalability.

In (Yussupov et al., 2019), the authors conduct a systematic mapping study to identify and discuss challenges related to engineering FaaS platforms. In contrast to our work, FaaS platforms are considered

independent of their application domain. The review does not focus on identifying specific aspects of their application to the Cloud-to-Thing continuum. By contrast, (Kjorveziroski et al., 2021) focuses on applying FaaS to the IoT domain. This work investigates aspects complementary to our study such as support for security, and performance. In contrast to our study, there is no focus on the deployment and allocation aspects, and the 'Edge', 'Fog', and 'CPS' keywords are not part of the study query string, reducing their scope.

(Leitner et al., 2019) presents a multivocal systematic literature review intending to characterize the use of FaaS. The study does not focus on the design of FaaS platforms for the Cloud-to-Thing. However, it provides valuable insight into the typical architectures and patterns for building FaaS-based applications.

As noted, the previous works only partially cover our RQs, typically with a different perspective – *i.e.*, not focusing on the allocation and deployment of serverless functions over the Cloud-to-Things continuum.

6 CONCLUSION

In this work, we have examined the research landscape of Function-as-a-Service platforms for the Cloud-to-Thing continuum. After systematically identifying and reviewing 33 primary studies out of hundreds of relevant papers in this field, we have found out that there has been a growth in interest and maturity level of the FaaS4C2T platforms over the years, yet maturity still remains low. There are still gaps in the research and several challenges remain to be tackled with still a very limited coverage of the Cloud-to-Thing continuum for the deployment and placement of serverless functions. In particular, we identified that there is little coverage of the Things layer, and there is no clear study exhibiting all the possible benefits of applying serverless functions to the Things layer. We believe this is due to its diverse technical challenges, in particular regarding the support for virtualization and dynamic deployment.

ACKNOWLEDGEMENTS

This work was partially funded by the uropean Union under the Horizon Europe grant 101070455 (DYNABIC). Bárbara da Silva Oliveira is supported by the BRAFITEC program at Université Côte d'Azur, financed by CAPES within the Ministry of Education of Brazil, and by the scholarship PIBIC, financed by Universidade Federal do Ceará.

REFERENCES

Arcangeli, J.-P., Boujbel, R., and Leriche, S. (2015). Automatic deployment of distributed software systems: Definitions and state of the art. *Journal of Systems and Software*, 103:198–218.

Aslanpour, M. S., Toosi, A. N., Cicconetti, C., Javadi, B., Sbarski, P., Taibi, D., Assuncao, M., Gill, S. S., Gaire, R., and Dustdar, S. (2021). Serverless edge computing: vision and challenges. In *2021 Australasian Computer Science Week Multiconference*, pages 1–10. ACM.

Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., et al. (2017). Serverless computing: Current trends and open problems. In *Research advances in cloud computing*, pages 1–20. Springer.

Baresi, L., Mendonça, D. F., Garriga, M., Guinea, S., and Quattrocchi, G. (2019). A unified model for the mobile-edge-cloud continuum. *ACM Transactions on Internet Technology (TOIT)*, 19(2):1–21.

Baresi, L. and Quattrocchi, G. (2021). PAPS: A serverless platform for edge computing infrastructures. *Frontiers in Sustainable Cities*, 3.

Bergmayr, A., Breitenbücher, U., Ferry, N., Rossini, A., Solberg, A., Wimmer, M., Kappel, G., and Leymann, F. (2018). A systematic review of cloud modeling languages. *ACM Computing Surveys (CSUR)*, 51(1):1–38.

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

Bocci, A., Forti, S., Ferrari, G.-L., and Brogi, A. (2021). Secure FaaS orchestration in the fog: how far are we? *Computing*, 103(5):1025–1056.

Byers, C. and Swanson, R. (2017). OpenFog Reference Architecture for Fog Computing. Technical report, OpenFog Consortium.

Cassel, G. A. S., Rodrigues, V. F., da Rosa Righi, R., Bez, M. R., Nepomuceno, A. C., and da Costa, C. A. (2022). Serverless computing for Internet of Things: A systematic literature review. *Future Generation Computer Systems*, 128:299–316.

Cheng, B., Fuerst, J., Solmaz, G., and Sanada, T. (2019). Fog function: Serverless fog computing for data intensive IoT services. In *2019 IEEE International Conference on Services Computing*, pages 28–35. IEEE.

Cicconetti, C., Conti, M., and Passarella, A. (2020). A decentralized framework for serverless edge computing in the internet of things. *IEEE Transactions on Network and Service Management*, 18(2):2166–2180.

Costa, B., Bachiega Jr, J., de Carvalho, L. R., and Araujo, A. P. (2022). Orchestration in fog computing: A com-

prehensive survey. *ACM Computing Surveys (CSUR)*, 55(2):1–34.

Elkholy, M. and Marzok, M. A. (2022). Light weight serverless computing at fog nodes for internet of things systems. *Indones. J. Electr. Eng. Comput. Sci*, 26(1):394–403.

Ferry, N., Dautov, R., and Song, H. (2022). Towards a Model-Based Serverless Platform for the Cloud-Edge-IoT Continuum. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 851–858. IEEE.

George, G., Bakir, F., Wolski, R., and Krintz, C. (2020). NanoLambda: Implementing functions as a service at all resource scales for the internet of things. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 220–231. IEEE.

Hall, A. and Ramachandran, U. (2019). An execution model for serverless functions at the edge. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, pages 225–236. ACM.

Hetzel, R., Kärkkäinen, T., and Ott, J. (2021). μactor: Stateful serverless at the edge. In *Proceedings of the 1st Workshop on Serverless mobile networking for 6G Communications*, pages 1–6. ACM.

Jang, S. Y., Kostadinov, B., and Lee, D. (2021). Microservice-based edge device architecture for video analytics. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 165–177. IEEE.

Kitchenham, B. A., Budgen, D., and Brereton, O. P. (2011). Using mapping studies as the basis for further research–a participant-observer case study. *Information and Software Technology*, 53(6):638–651.

Kitchenham, B. A. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University.

Kjorveziroski, V., Filiposka, S., and Trajkovik, V. (2021). IoT Serverless Computing at the Edge: A Systematic Mapping Review. *Computers*, 10(10).

Leitner, P., Wittern, E., Spillner, J., and Hummer, W. (2019). A mixed-method empirical study of Function-as-a-Service software development in industrial practice. *Journal of Systems and Software*, 149:340–359.

LF Edge (2019). Open glossary of edge computing, version 2.0. Technical report, Linux Foundation.

Li, Y., Lin, Y., Wang, Y., Ye, K., and Xu, C.-Z. (2022). Serverless computing: State-of-the-art, challenges and opportunities. *IEEE Transactions on Services Computing*, pages 1–1.

Lordan, F., Lezzi, D., and Badia, R. M. (2021). Colony: Parallel Functions as a Service on the Cloud-Edge Continuum. In *Euro-Par 2021: Parallel Processing*, pages 269–284. Springer.

Mahmud, R., Kotagiri, R., and Buyya, R. (2018). Fog computing: A taxonomy, survey and future directions. In *Internet of everything*, pages 103–130. Springer.

Mahmud, R., Ramamohanarao, K., and Buyya, R. (2020). Application management in fog computing environments: A taxonomy, review and future directions. *ACM Computing Surveys (CSUR)*, 53(4):1–43.

Mell, P. and Grance, T. (2001). The NIST Definition of Cloud Computing. Technical Report 800-145, National Institute of Standards and Technology.

Mittal, V., Qi, S., Bhattacharya, R., Lyu, X., Li, J., Kulkarni, S. G., Li, D., Hwang, J., Ramakrishnan, K., and Wood, T. (2021). Mu: An efficient, fair and responsive serverless framework for resource-constrained edge clouds. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 168–181. ACM.

Pang, Z., Sun, L., Wang, Z., Tian, E., and Yang, S. (2015). A survey of cloudlet based mobile computing. In *2015 International Conference on Cloud Computing and Big Data (CCBD)*, pages 268–275. IEEE.

Pelle, I., Czentye, J., Dóka, J., Kern, A., Gerő, B. P., and Sonkoly, B. (2020). Operating Latency Sensitive Applications on Public Serverless Edge Cloud Platforms. *IEEE Internet of Things Journal*, 8(10):7954–7972.

Pelle, I., Paolucci, F., Sonkoly, B., and Cugini, F. (2021). Latency-sensitive edge/cloud serverless dynamic deployment over telemetry-based packet-optical network. *IEEE Journal on Selected Areas in Communications*, 39(9):2849–2863.

Persson, P. and Angelsmark, O. (2017). Kappa: Serverless IoT deployment. In *Proceedings of the 2nd International Workshop on Serverless Computing*, pages 16–21.

Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18.

Taibi, D., El Ioini, N., Pahl, C., and Niederkofler, J. R. S. (2020). Patterns for serverless functions (function-as-a-service): A multivocal literature review. In *Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER)*, pages 181–192. SciTePress.

Tran, N. K., Sheng, Q. Z., Babar, M. A., and Yao, L. (2017). Searching the Web Of Things: state of the art, challenges, and solutions. *ACM Computing Surveys (CSUR)*, 50(4):55.

Van Eyk, E., Toader, L., Talluri, S., Versluis, L., Uță, A., and Iosup, A. (2018). Serverless is more: From paas to present cloud computing. *IEEE Internet Computing*, 22(5):8–17.

Xie, R., Tang, Q., Qiao, S., Zhu, H., Yu, F. R., and Huang, T. (2021). When serverless computing meets edge computing: architecture, challenges, and open issues. *IEEE Wireless Communications*, 28(5):126–133.

Yussupov, V., Breitenbücher, U., Leymann, F., and Wurster, M. (2019). A systematic mapping study on engineering function-as-a-service platforms and tools. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, pages 229–240.