Literature Research: Function as a Service and Serverless Computing in the Cloud industry

Ilias Daia — (2622922 & 12000574)

Vrije Universiteit, Amsterdam, The Netherlands & Universiteit van Amsterdam

Abstract. Cloud computing services are always evolving. The Function as a Service (FaaS) paradigm is a recent trend that forces developers to consider whether adopting this new paradigm will benefit parts of their application. Users can define functions in a high-level programming language using FaaS. They or others can then call that function one or more times, relying on the cloud provider's elastic infrastructure provisioning. Many factors, however, influence or even prevent the use of FaaS. The potential and applications of FaaS and serverless computing are discussed in this study, along with how FaaS might advance in the field of cloud computing.

Keywords: Cloud · Cloud computing · FaaS · Serverless computing · IT-services

1 INTRODUCTION

For many businesses today, developing apps that run in a cloud environment is the standard. However, the choices for how applications are created and run in the cloud are numerous, and they continue to evolve as cloud providers introduce new services. Cloud computing is a computer paradigm that has revolutionized enterprise and Internet computing. [1] Cloud computing is now synonymous with on-demand provisioning and delivery of IT services in a pay-as-you-go model in practically all industries throughout the world. The long-term efforts of the computing research community around the world are attributed with cloud computing's success as a technology. The three major cloud product sectors are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Each of these product categories has an impact on a variety of industries. [2]

Cloud computing is defined as ”a model for enabling ubiquitous, accessible, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be instantly supplied and released with no administrative effort or service provider contact. The concept of cloud computing has developed as a viable option for exploiting on-demand computational and software resources. It delegated to service providers the complex and time-consuming resource and software management duties that customers used to handle. As a result, the ability to respond to shifting demand is improved. It also reduces the cost of infrastructure and the cost of coordinating IT resources. Investment and operations costs are lowered due to the Cloud's economies of scale. As a result, cloud-based systems have created quite a stir in the IT industry. Another essential characteristic is that it appeals to small research organizations in the field of computational science and engineering, as well as corporate clients. The most popular method of hosting an application is through cloud-based platforms, which offer a wide range of services. There are many distinct types of services that are built on top of one another. [3]
Serverless computing, with the Function as a Service (FaaS) offering at its core, is a recent trend. A recent development in cloud client-server systems is serverless architecture, also known as Function as a Service. A thin-client request, in the traditional approach, invokes some server-side services. The server is a single unit that implements all of the logic for authentication, page navigation, searching, and transactions. Developers create a serverless function by writing code that processes an input and provides an output according to a predetermined interface, which may be integrated with other code artefacts (e.g., libraries). After that, the serverless function can be deployed to a FaaS platform, which allows it to run. The FaaS platform saves the function code and registers it with one or more triggers so that it can be executed. The triggers that are available are determined by the platform. [4]

Build up of the paper There are five sections to this study. First, in the Introduction, the background information is presented. The research questions and motives for selecting them are then outlined under Methodology. Following that, in the Discussion of Literature portion, the literature is divided into three sections in order to address the research questions. The conclusion of this paper includes a summary, followed by Reflection and Limitation. Last but not least, the Appendix explains the terminologies used in this paper.

2 METHODOLOGY

The title of this literature study is ”Function as a Service in Cloud,” which refers to the use of the cloud to perform small processes. As a result, the research questions and rationale for the stated research questions are as follows:
Research Questions:

- **RQ1: What are considered the challenges for FaaS?**
  This is due to the fact that FaaS is a relatively young cloud computing technology, making it prone to errors. Given the current state of FaaS, practically every concept requires a great deal of trial and error before reaching the point of being reliable. As a result, there is room for development and certain hurdles to overcome. This is also found in the academic publications that were researched.
- **RQ2: In which application domains does FaaS have the biggest potential?**
  Given that cloud computing is primarily employed in the business world, it would be worthwhile to investigate which application domains it can be applied to. The use of FaaS is currently confined to platform vendors. On the other hand, certain vendors, such as OpenFaas, provide the open-source freedom.
- **RQ3: Does FaaS have a long-lasting future in the cloud landscape?**
  Given how new FaaS is in the cloud domains, the question remains whether it will last long enough to be considered durable. A number of different businesses have been significantly impacted by cloud computing, which has been around for approximately 15 years. With this in mind, the answer to the provided question might be interesting.

2.1 Selected papers

To address the setup research questions, relevant information was chosen from web databanks. The query phrase was then entered into Google Scholar to look for preliminary results on previously published articles on the subject. In order to reduce the scope of the search, inclusion and exclusion criteria were devised based on the research goal and questions, which facilitated in the removal of irrelevant studies and the selection of primary studies. The primary papers were

then thoroughly evaluated, and pertinent data was obtained to help answer the study objectives.
The inclusion [I] and exclusion [E] criteria are listed as follows:

- **I1** Studies that concern cloud computing
- **I2** Studies that are specific concerning FaaS & Serverless
- **I3** White papers from cloud vendors
- **I4** Studies in the business application domain of Cloud
- **E1** Studies that are not complete or require a paid subscription.
- **E2** Studies of cloud computing that do no mention the use of Serverless
- **E3** Studies that don't mention the use of FaaS

3 BACKGROUND RESEARCH

3.1 What is Serverless?

Serverless computing is a novel notion that allows developers to concentrate on the core functionality of their code while abstracting the underlying infrastructure. To manage server-side logic and state, the term "serverless" was initially used to describe applications that heavily or completely include third-party, cloud-hosted apps and services. These are often "rich client" applications, such as single-page web apps or mobile apps, that take advantage of the wide ecosystem of cloud-accessible databases, authentication services, and other services. [5]
Serverless apps are those in which the application developer still writes server-side logic, but it is run in stateless compute containers that are event-triggered, ephemeral (lasting only one invocation), and fully managed by a third party, as opposed to traditional architectures.
Serverless computing is a type of cloud computing that enables customers to run event-driven and granularly billed applications without having to worry about operational logic. [6]
Physical servers continue to exist, thus this does not imply that they are no longer available. As a result, developers no longer have to bother about server management. The key advantage is that developers don't have to worry about managing underlying services and operating systems because the FaaS platform takes care of that for them, as well as providing transparent scaling options.

When no function is running, platforms automatically scale down to zero, preventing resource waste. [7]

3.2 What is FaaS?

FaaS is a new method of creating and deploying server-side software applications, based on individual functions as a deployment unit. Examples of FaaS providers are AWS Lambda (Amazon Web Services) [8], Google Cloud Functions (Google) [9], IBM Cloud Functions (IBM) [10], Microsoft Azure Functions (Microsoft) [11] and Apache OpenWisk (Apache) [12]. The most widely adopted FaaS implementation currently available is AWS Lambda (Amazon Web Services), many people refer to FaaS implementations as serverless computing [13]. However, FaaS is just one implementation of the serverless computing concept. [14]
Function-as-a-Service (FaaS) allows developers to define, orchestrate and run modular event-based pieces of code on virtualised resources, without the burden of managing the underlying infrastructure nor the life-cycle of such pieces of code. [15] Indeed, FaaS providers offer resource auto-provisioning, auto-scaling and pay-per-use billing at no costs for idle time. This makes it easy to scale running code, and it represents an effective and increasingly adopted way to deliver software. As in the case of AWS Lambda or Azure Functions, FaaS platforms are often managed by cloud providers. These cloud providers also offer direct connectivity with their respective cloud services, such as database or messaging services, to act as triggers or dependencies for functions. Open-source systems such as OpenFaaS3 and Knative4 are also available.

FaaS platforms have proved wildly successful in industry as a way to reduce costs and the need to manage infrastructure. The FaaS model is a type of serverless computing. The FaaS provider is in charge of the lifecycle and event-driven execution of the application. A FaaS platform manages the functions in the FaaS model. The FaaS platform is in charge of deploying and ensuring that user-provided functions work properly. Instrumenting and monitoring the function's performance, as well as automatically scaling (autoscaling) the resources given to the function based on the incoming workload, are among its operational tasks. [16]

We broadly categorize platforms as commercial, open source, or academic, and further compare them based on the following categories:

Table 1. Categories of platforms [15]

Categories	Description
Languages:	The programming languages that can be used to define functions.
Infrastructure:	Where the FaaS platform is deployed and where functions are executed, e.g., cloud, Kubernetes.
Virtualization:	The virtualization technology used to isolate and deploy functions.
Triggers:	How functions are invoked and whether specific event sources are supported.
Walltime:	How long functions are permitted to execute.
Billing:	What billing models are used to recoup costs.

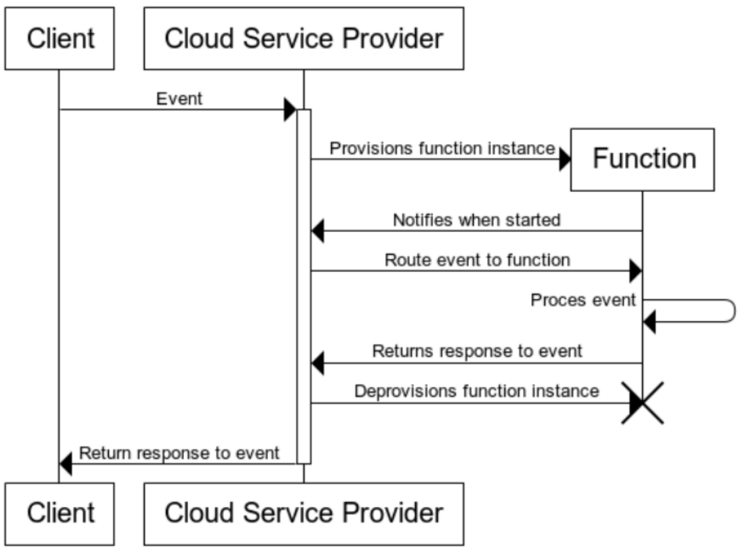

Fig. 1. Sequence diagram of basic FaaS consumption [17]

Figure 1 depicts the process of consuming a serverless function, which can be stated as follows at a high abstraction level: 1. An incoming event is detected by the cloud service provider. 2. The function will be executed in a container. 3. Once the event has begun, it will be sent to this container. 4. The event will be processed by the function. 5. The container will then be terminated. [17]
This entire procedure is completed in a relatively short period of time. Although it is dependent on other aspects such as programming language, dependencies, and available memory. The time it takes the cloud service provider to establish the container and launch the function that the incoming event requires is the most time-consuming aspect of this process [18]. Most cloud service providers address this issue by keeping the function ”warm” for an arbitrary number of minutes after it has been used; in other words, the container, including the function, is not directly destroyed after usage, but is kept idle for some time. Because the container may be reused, the cold start delay is eliminated.

3.3 Differnce FaaS and Serverless

Many people confuse serverless and functions-as-a-service (FaaS), however FaaS is a subset of serverless. The term ”serverless” refers to any service category where the configuration, management, and payment of servers are hidden from the end user, such as compute, storage, database, messaging, api gateways, and so on. FaaS, on the other hand, focuses on the event-driven computing paradigm, in which application code, or containers, only run in response to events or requests, and is likely the most central technology in serverless architectures. [16]
The core deployment unit of FaaS is a serverless function. FaaS allows individual deployment of functions or activities, whereas traditional cloud computing considers huge all-encompassing software programs (or services in the case of Service Oriented Architectures (SOA) or microservice architectures) as the basic deployment unit. These functions are all deployed independently in lightweight containers that may be launched (provisioned) as soon as the cloud service provider gets an event requesting that specific software application function. [19]
As indicated in section 3.1, this does not mean the absence of physical servers, which still exist. As a result, developers no longer have to bother about server management. The main benefit is that developers do not need to manage underlying services and operating systems, as this is the responsibility of the FaaS platform that also provides transparent scalability mechanisms. Platforms auto-

matically scale to zero when there is no function running, thus avoiding waste of resources. [20]
The idea of the Internet of Things (IoT) is also intriguing when discussing the absence of backend hardware. It has emerged as a fairly new trending concept that is closely related to ubiquitous computing, which allows physical objects (things) to "talk" to each other while exchanging information through the Internet. Although originally proposed for the cloud, serverless computing can also be a great ally for IoT as FaaS platforms can be placed physically closer to sensors and actuators in order to run functions with lower latency. This is specially important for mission-critical scenarios that cannot rely on Internet connection. For other scenarios, when intermittent connection is not a problem, functions can also be executed on more than a single layer and benefit from traditional cloud serverless platforms, that provide the illusion of infinite resources for processing huge workloads of IoT data produced by sensors. [21]

3.4 What is considered the biggest challenge for FaaS?

FaaS functions have significant constraints when it comes to local (machine/instance-bound) state, such as data kept in variables in memory or data written to local storage. You do have such storage, but there's no guarantee that it'll be preserved across invocations, and you shouldn't anticipate state from one function to be available in another. As a result, FaaS functions are sometimes referred to as stateless; nevertheless, any FaaS function state that needs to be durable must be externalized outside the FaaS function instance. For FaaS functions that are naturally stateless—that is, those that provide a purely functional transition from their input to their output—this is irrelevant. [22]Also, in FaaS functions, the amount of time each invocation can execute is normally limited. This means that some long-duration jobs may not be suited for FaaS functions without re-architecture—you may need to build several coordinated FaaS functions, whereas in a traditional system, one long-duration work could coordinate and execute both coordination and execution. Thus, a functionality should be stateless because the FaaS platform cannot guarantee re-execution on the same instance. This signifies that a feature should not maintain state from previous invocations that is required for subsequent invocations (e.g., data saved in memory or on the function instance's disk). As a result, a serverless function follows the stateless component paradigm, and any state that is required should be externalized.

This is unimportant for FaaS functions that are naturally stateless—that is, those that provide a purely functional transition of their input to their output. However, for others, this might have a significant impact on application architecture. [19] A functionality is a discrete piece of business logic that can be built as a serverless function and optionally deployed. An application is made up of different features. A serverless application is one that is mostly made up of serverless functions. [19]

4 CURRENT FUNCTION AS A SERVICE PLATFORMS

4.1 AWS Lambda

AWS Lambda was the first cloud functions product when it released in 2015, and it offered something unique and compelling: the ability to execute practically any code that runs on a server. It offered pay-as-you-go support for a variety of programming languages and arbitrary libraries, all while remaining secure and scalable.
Lambda runs your code on high-availability compute infrastructure and manages your compute resources for you. Maintenance of the server and operating system, capacity provisioning and automatic scaling, code and security patch release, and code monitoring and logging are all part of this. All you have to do now is provide the code.
It did, however, place several limits on the programming model, which confine it to specific uses even today. There is a maximum execution time, no persistent state, and limited networking to name a few. Several serverless environments can now run any code, each tailored to a specific use case. Google Cloud Dataflow and AWS Glue, for example, allow programmers to run arbitrary code as part of a data processing pipeline.
In November 2014, Amazon launched AWS Lambda, the first company to bring serverless computing to a wider audience. 2018a, Amazon Web Services, Inc. AWS Lambda offers the most flexibility of any of the providers considered. Memory usage can be specified in 64-MB increments starting at 128 MB and going up to 3008 MB (a little less than 3 GB). Furthermore, many more configuration options, such as concurrency and cloud function chaining, are available. While the number of CPU cycles allocated per memory configuration is unknown, it

is stated that CPU power is “proportional” to the amount of memory selected. AWS Lambda’s benefit comes from its integration with other AWS products, which can be considered vendor lock-in if used.[23]

4.2 Google Cloud Functions

Google Cloud Functions is a serverless platform for developing and connecting cloud services. Cloud Functions allows you to create simple, single-purpose functions that are linked to events generated by your cloud infrastructure and services. When an event that is being watched is fired, your function is called. After its launch in February 2016, Google Cloud Functions remained in beta until mid-June 2018. The service is now available for production use. (Google LLC, 2018a) Although there are enough options to run and monitor an application, the configuration options are limited and sparse when compared to other cloud service providers. [3]

4.3 IBM Cloud Functions

In a nutshell, IBM Cloud Functions is a FaaS platform for running functions in reaction to events, and it is built on Apache OpenWhisk. [24] The IBM Cloud Functions service limits function execution to 600 seconds, 512MB of RAM per function execution, and a maximum of 1,000 concurrent invocations at the time of writing, though the number of concurrent functions can be extended if necessary. IBM Cloud Functions is the only solution that does not require direct vendor lock-in. As previously said, the RAM looks to be limited to 512 MB, while more may be possible depending on the configuration. Using Docker’s memory management, however, more RAM is accessible for cloud functions without the need to terminate them right once.[25]

IBM’s 2019 notion of predicting the future via Monte Carlo simulations was done by making use of their IBM Cloud Functions. Monte Carlo simulations are mathematical approaches that have been used to forecast the future outcomes of some hard-to-predict events for more than a century. When not in use, IBM Cloud Functions is a serverless functions-as-a-service platform that executes code in response to incoming events. They were able to complete the Monte Carlo simulation in around 90 seconds with 1000 concurrent invocations, as opposed to 247 minutes with nearly 100% CPU utilization executing the identical flow on a laptop with four CPU cores.[26]

4.4 Microsoft Azure Functions

Microsoft Azure Functions is Microsoft's FaaS service for the Azure Cloud. In contrast to other cloud service providers,
Azure Functions is an event-driven, compute-on-demand solution that adds the ability to integrate code triggered by events in Azure or third-party services, as well as on-premises systems, to the existing Azure application platform. Developers may use Azure Functions to take action by connecting to data sources or messaging solutions, making it simple to process and react to events. Developers can use Azure Functions to create HTTP-based API endpoints that can be accessed by a variety of apps, mobile devices, and IoT devices. You only pay for the resources you use because Azure Functions is scale-based and on-demand. Microsoft Azure Functions does not require users to specify the amount of RAM required during function deployment, making it the only true serverless solution. Instead, only the used peak memory consumption per invocation is measured and billed during the entire execution period. Furthermore, with a BTU of 1 ms, Microsoft Azure Functions has the lowest BTU among cloud service providers. [3]

4.5 OpenFaaS

In the academic and corporate worlds, OpenFaaS is a prominent open-source serverless platform. OpenFaaS has a basic scheduling policy that distributes functions among cluster computing resources. As a result, it is unsuitable for handling latency-sensitive applications in a geo-distributed context, where network latencies are minor and significantly impact application response time.
Because of its lightweight and extensible architecture, OpenFaaS is now one of the most widely used open-source serverless platforms. OpenFaaS, which is based on Kubernetes, makes it easier to manage complicated multi-function applications. Function instances run within containers to achieve rapid startup and shutdown times. One of the primary components of OpenFaaS is the API gateway, which provides an interface for creating, deleting, modifying, monitoring, and scaling functions. It also accepts both external and internal requests and routes them to the relevant function for processing.

The Kubernetes scheduler is used by OpenFaaS to distribute functions across computing nodes. All of these key serverless platforms treat functions as black-boxes, assuming that computer resources are distributed locally and connected via high-speed communication lines.[27]

Table 2. [15]

Vendor	**Application Domain**
AWS Lambda	File and stream processing – Web applications – IoT and Mobile backends
Google Cloud	Integration with third-party services and APIs - Virtual assistants and conversational experiences - Sentiment analysis
IBM Cloud	Actions exposed through API gateway - Packages for common tasks
Microsoft Azure	Computations - Analytics - Storage and Networking.
OpenFaaS	open-source platform with a range of applications

5 Conclusion

FaaS is still in its beginning phases in the cloud business, but it has made a strong performance. Observing the use of FaaS, there were various disadvantages and obstacles. The answers to the previously identified research questions that were derived from the literature analysis are summarized in this part, and a short discussion with the mention of the limitations of this research are mentioned.

What are considered the challenges for FaaS? Predictable performance: it has shown that FaaS has unpredictable performance [17], preventing them from being used in applications that require rigorous guarantees. The rationale for this is simple: serverless providers use statistical multiplexing to give the appearance of boundless resources while denying users control over resource oversubscription. There is always the possibility of queueing delays due to bad timing. Reassigning resources from one customer to another incurs a delay cost, which is referred to as a "cold start" in the cloud function context. Cold start latency comprises various components, one of which is the time it takes to initialize the function's software environment. There's definitely still a lot of potential for improvement in these areas, but there's evidence that performance optimization and security isolation are fundamentally incompatible. This means that some long-duration activities are not suited for FaaS functions without re-architecture—you may need to build several coordinated FaaS functions, whereas in a traditional system, one long-duration work could coordinate and execute both coordination and execution. [28]

In which application domains does FaaS have the biggest potential? FaaS is a new approach to developing and distributing server-side software applications that uses individual functions as the deployment unit. In the public cloud, there are a multitude of companies that offer FaaS. AWS Lambda, Google Cloud Functions, and Microsoft Azure being the most well-known suppliers in the business application industry, as described in section four. AWS Lambda is the most well-known of the four applications, and it is primarily employed in the sector of business. Given how new serverless computing is, research in FaaS benchmarking is still limited.

Currently, the idea of FaaS has the biggest appeal for low usage (memory) applications, In particular, the code may be scaled to zero when no servers are actually running when the user's function code is not used, and the user incurs no cost." This is in contrast to VM-based alternatives, which charge the user even during idle periods. [3]

Alternative platforms, such as OpenFaas, have the flexibility of being open-source, which is a significant benefit over other suppliers. Because of its lightweight and extensible architecture, OpenFaaS is now one of the most widely used open-source serverless platforms wordwide.[27]

Finally, by improving software quality while cutting costs, serverless computing has the potential to bring significant benefits to software developers, architects, and application owners. As a result, FaaS is best suited to low-usage apps that don't require scaling.

Does FaaS have a long-lasting future in the cloud landscape? Serverless functions are usually created for a specific FaaS platform and may not be deployable on other platforms unless significant changes are made. Furthermore, FaaS platforms offer a number of further services such as platform-specific storage adapters, scaling, monitoring and logging tools, authentication components, and so on. Moving to another FaaS platform becomes increasingly challenging as

serverless operations become more reliant on these platform-provided services. This could be a significant barrier to FaaS reaching its full potential.
However, given that companies such as OpenFaaS and IBM provide a solution to this issue, it appears that the problem will be handled for customers who require platform flexibility. AWS, for example, appears to be growing steadily in the market, indicating that the need for FaaS will continue to expand in the future. [4]
Overall, serverless computing is still in its infancy, and there are many open questions about how to define and implement its abstractions. Because serverless is now only suitable for a few applications, cloud providers will develop more application-specific and general-purpose serverless products to support a wider range of use cases. As a result, in order to make a good prediction, we must wait and watch what the developments are in the future. Nonetheless, based on what we know about FaaS and Serverless, we have high hopes for positive recognition for the two in the future.

6 Discussion

One shortcoming of this literature review is the scarcity of scholarly papers on FaaS. Topics such as deploying serverless computing in science and academic applications of the serverless model. There were only a few academic papers exploring or discussing the topic. This is primarily owing to the ongoing research and use of serverless computing and FaaS, although there was still some useful literature available. Most studies, however, suggested their own version of a Function as a Service product, adapted to their individual case studies and frameworks. This field is still relatively new, with numerous contributions expected in the future years.

References

1. Ling Qian, Zhiguo Luo, Yujian Du, and Leitao Guo. Cloud computing: An overview. In *IEEE international conference on cloud computing*, pages 626–631. Springer, 2009.
2. Joel Scheuner and Philipp Leitner. Function-as-a-service performance evaluation: A multivocal literature review. *Journal of Systems and Software*, 170:110708, 2020.
3. Timon Back. *Hybrid serverless and virtual machine deployment model for cost minimization of cloud applications*. PhD thesis, 2018.
4. Johann Schleier-Smith, Vikram Sreekanti, Anurag Khandelwal, Joao Carreira, Neeraja J Yadwadkar, Raluca Ada Popa, Joseph E Gonzalez, Ion Stoica, and David A Patterson. What serverless computing is and should become: The next phase of cloud computing. *Communications of the ACM*, 64(5):76–84, 2021.
5. Vojdan Kjorveziroski, Sonja Filiposka, and Vladimir Trajkovik. Iot serverless computing at the edge: A systematic mapping review. *Computers*, 10(10):130, 2021.
6. Scott Hendrickson, Stephen Sturdevant, Tyler Harter, Venkateshwaran Venkataramani, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. Serverless computation with {OpenLambda}. In *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.
7. Mohammad Shahrad, Jonathan Balkind, and David Wentzlaff. Architectural implications of function-as-a-service computing. In *Proceedings of the 52nd annual IEEE/ACM international symposium on microarchitecture*, pages 1063–1075, 2019.
8. Aws lambda. `http://precog.iiitd.edu.in/people/anupama`, 2022.
9. Google cloud function. `https://cloud.google.com/functions/docs/concepts/overview`, 2022.
10. Ibm cloud function. `https://cloud.ibm.com/docs/openwhisk?topic=openwhisk-getting-started`, 2022.
11. Microsoft azure. `https://azure.microsoft.com/nl-nl/`, 2022.
12. Apache openwhisk. `https://openwhisk.apache.org/`, 2022.
13. Michael Roberts and John Chapin. *What is Serverless?* O'Reilly Media, Incorporated, 2017.
14. Mengting Yan, Paul Castro, Perry Cheng, and Vatche Ishakian. Building a chatbot with serverless computing. In *Proceedings of the 1st International Workshop on Mashups of Things and APIs*, pages 1–4, 2016.
15. Alessandro Bocci, Stefano Forti, Gian-Luigi Ferrari, and Antonio Brogi. Secure faas orchestration in the fog: how far are we? *Computing*, 103(5):1025–1056, 2021.
16. Robin Lichtenthäler, Stefan Winzinger, Johannes Manner, and Guido Wirtz. When to use faas?-influencing technical factors for and against using serverless functions. In *ZEUS*, pages 39–47, 2020.

17. RTJ Bolscher. Leveraging serverless cloud computing architectures: developing a serverless architecture design framework based on best practices utilizing the potential benefits of serverless computing. Master's thesis, University of Twente, 2019.
18. Blesson Varghese and Rajkumar Buyya. Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems*, 79:849–861, 2018.
19. Matt Crane and Jimmy Lin. An exploration of serverless architectures for information retrieval. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, pages 241–244, 2017.
20. Gustavo André Setti Cassel, Vinicius Facco Rodrigues, Rodrigo da Rosa Righi, Marta Rosecler Bez, Andressa Cruz Nepomuceno, and Cristiano André da Costa. Serverless computing for internet of things: A systematic literature review. *Future Generation Computer Systems*, 128:299–316, 2022.
21. R Arokia Paul Rajan. A review on serverless architectures-function as a service (faas) in cloud computing. *Telkomnika*, 18(1):530–537, 2020.
22. Simon Shillaker and Peter Pietzuch. Faasm: Lightweight isolation for efficient stateful serverless computing. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 419–433, 2020.
23. Maciej Malawski, Adam Gajek, Adam Zima, Bartosz Balis, and Kamil Figiela. Serverless execution of scientific workflows: Experiments with hyperflow, aws lambda and google cloud functions. *Future Generation Computer Systems*, 110:502–514, 2020.
24. Josep Sampé, Gil Vernik, Marc Sánchez-Artigas, and Pedro García-López. Serverless data analytics in the ibm cloud. In *Proceedings of the 19th International Middleware Conference Industry*, pages 1–8, 2018.
25. Pedro García López, Marc Sánchez-Artigas, Gerard París, Daniel Barcelona Pons, Álvaro Ruiz Ollobarren, and David Arroyo Pinto. Comparison of faas orchestration systems. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 148–153. IEEE, 2018.
26. Josep Sampe, Marc Sanchez-Artigas, Gil Vernik, Ido Yehekzel, and Pedro Garcia-Lopez. Outsourcing data processing jobs with lithops. *IEEE Transactions on Cloud Computing*, 2021.
27. Fabiana Rossi, Simone Falvo, and Valeria Cardellini. Gofs: Geo-distributed scheduling in openfaas. In *2021 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2021.
28. Erwin van Eyk. *The design, productization, and evaluation of a serverless workflow-management system*. PhD thesis, 2019.