# IoT Serverless Computing at the Edge

Subjects: Computer Science, Information Systems

Contributor: Vojdan Kjorveziroski

Serverless computing is a new concept allowing developers to focus on the core functionality of their code, while abstracting away the underlying infrastructure. Even though there are existing commercial serverless cloud providers and open-source solutions, dealing with the explosive growth of new Internet of Things (IoT) devices requires more efficient bandwidth utilization, reduced latency, and data preprocessing closer to the source, thus reducing the overall data volume and meeting privacy regulations. One way of achieving this is to move serverless computing to the network edge.

serverless computing edge computing function as a service Internet of Things

## 1. Introduction

The ever increasing progress in hardware development and computer networking paved the way for the introduction of cloud computing, which in turn has led to a new revolution, allowing computing capacity to be perceived as just another utility, used on-demand, with virtually limitless capacity . Both academia and industry have invested in the creation of different cloud computing infrastructure, depending on their needs, currently available resources, and cost, resulting in the deployment of various private, public, community, and hybrid clouds [1]. However, to allow regular users to benefit from such vast computing capacity, additional abstractions are introduced, in the form of infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) offerings. IaaS provides the lowest level of abstraction, allowing users to rent computing, networking, and storage capacity on-demand, usually in the form of virtual machines (VMs), and utilize them as they see fit, building their own infrastructure on top. PaaS goes a step further, and is primarily aimed at software developers, abstracting away the necessary VM management, and instead providing the building blocks and interfaces for directly hosting developed applications, along with any other prerequisites, such as databases and message queues. Finally, SaaS, aimed at end-users, provides the highest level of abstraction, where the service being offered is a finished software product, ready to be used, without any additional requirements in terms of maintenance, updates, or management.

These three offerings are by no means the only products available as a service today. The idea of abstracting complicated tasks away from the users is natural and proved very popular, resulting in * (anything) as a service [2], which serves as a general term for products that free the end-user from performing a demanding task, and instead offloading it to a professional service provider, thus freeing up customers' time and reducing the time to market.

Even though most service providers support granular billing policies for all of the above service offerings, and customers can be billed on a minute-by-minute or even per-second intervals, costs are incurred for simply leaving the infrastructure running, no matter the amount of visitors that it serves. Serverless computing is a recent paradigm shift that aims to overcome these issues, while making the application development process even simpler for developers. The major question faced by developers is no longer “where to deploy”, but instead “how to create” the application, focusing foremost on its features. Serverless is comprised of Function as a Service (FaaS) and Backend as a Service (BaaS), and despite its name, it still relies on underlying servers for hosting the workload and data processing. However, compared to the other *as a service* approaches, it provides an even greater abstraction layer to the developers, who no longer have to think in terms of the infrastructure, resource requirements, or even scaling, and can instead focus on writing granular functions with a well defined role, and integrating their functionality to achieve more complex systems or applications. Thus, the main point of function as a service, and serverless computing in general, is to allow the developer to write functional code with a well defined task, using the desired programming language, which can then be uploaded and hosted as an atomic unit on a provider’s infrastructure. This FaaS approach combined with common backend functionality such as databases, or message queues which are offered as a service as part of BaaS offerings, accessible through provider-defined APIs, unburdens users from any server management. Furthermore, by billing per function invocation, and allowing function instances to be scaled down to zero replicas when not being utilized, customers are billed only for the time that the function is active, while having access to seamless scalability, monitoring, and security features. The serverless approach is beneficial to service providers as well, since it advances the ever-present ideal of executing more workload on the same amount of resources. By transferring the responsibility for resource dimensioning away from the customers, service providers are better able to manage their computing capacity, utilized resources, as well as power usage.

The first public FaaS offering dates to 2015, when Amazon AWS introduced its Lambda computing service [3], aimed primarily at web developers. Others quickly followed [4][5][6] by introducing competing services inspired by the initial success and the potential benefits that the serverless approach might unlock. Open-source FaaS solutions are widely popular as well [7][8][9], and there are even cases where commercial service providers have either based their FaaS offerings on one of the existing open-source solutions, as is the case with IBM and OpenWhisk [10], or have open-sourced either in part, or completely the underlying components of their FaaS architecture [11], contributing to the open-source community, and thus directly investing in the serverless ecosystem.

Web development is not the only area where serverless computing unlocks interesting new opportunities. Another research area which has seen an enormous growth in recent years is the Internet of Things (IoT). Serverless for IoT would be particularly beneficial as a result of the inherently real-time and event-based workload of these systems [12][13]. However, IoT faces a different set of challenges in comparison to the typical client-server applications that were the primary targets for the initial serverless push. Even though the cloud has been utilized to a great extent in IoT scenarios, offering endless computing capacity, and data storage, means of actually transferring the data in an acceptable time-frame, without much delay, have always proved a challenge. While the cloud is an excellent choice for applications that run at human perception speed and response times of hundreds of milliseconds or even seconds are acceptable, optimizations have to be made to meet requirements for real-time

IoT applications, running at machine perception speed [14]. This latency and network capacity problem will become even more pronounced with the advent of billions of new IoT devices that will find their way in our lives. Moving the computing capacity towards the devices that actually generate the data is one of the solutions attracting great research interest. Edge computing reduces the network latency by allowing time-sensitive computations to be executed on compute infrastructure close to the data sources and can be seen as the missing piece to bring the simplicity of serverless computing to the event driven IoT world. Utilizing serverless edge computing transforms the previously utilized ship-data-to-code paradigm, which incurred high network latency and transmission costs, to a ship-code-to-data paradigm [15]. Furthermore, by initially preprocessing the data at the edge, not only can network bandwidth be saved and faster response time obtained, but compliance with data protection laws can be ensured as well. In this manner, customer data can be anonymized closer to the data source, in the same jurisdiction before being shipped to the cloud for long term storage and aggregation.

Many infrastructure providers have adapted their service offerings to include serverless products aimed at the network edge, such as AWS Greengrass [16], and Azure IoT Hub [17], bringing the associated benefits such as fast development, easy deployment, and seamless scalability to this part of the network. A number of open-source initiatives are also present, either adapting the existing open-source serverless platforms for the network edge, or starting from a clean slate, without any pre-existing technical debt, and developing entirely new solutions. While there is a perpetual discussion of centralized versus decentralized architectures, and the cycle seems to reverse itself during the years, serverless at the edge is still a novel research area with many outstanding issues left to be resolved.

## 2. Related Work

Serverless computing is an active research topic which has attracted a noticeable research interest in recent years with a large number of both primary and secondary literature. The majority of this work is focused on serverless computing in the cloud, categorizing it as an emerging technology with potentially great impact to various fields and use-cases in the future.

Varghese et al. [18] argue that with further advancements to the serverless paradigm, it can become a viable alternative for many more applications, including IoT ones which are primarily event driven. The authors of [12] share this vision for serverless computing, classifying it as the driving force behind sensor networks at the edge in the future, together with the help of blockchain and artificial intelligence (AI). The large applicability of this new paradigm is evident even now, with vastly different use-cases available today, such as the ability to run JavaScript serverless functions on provider edge infrastructure, offering faster response time to web users across the globe [19]. Other areas that might benefit from serverless are further discussed by Shafiei [20] et al. and Hassan et al. [21], including real-time collaboration and analytics, video processing, scientific computing, serving of machine learning models, file processing, smart grid, information retrieval, and chatbots.

By leveraging the effortless scalability that it offers, serverless computing can also be used for on-demand data processing and execution of resource intensive tasks which can be sped up by parallelly executing the same

function on various compute nodes, where each instance would work on a smaller partition of the original data. Buyya et al. [22] drive this concept even further, describing serverless pipelines comprised of multiple functions chained together with the aim of modeling complex data analysis workflows. Real world examples are already available in this case as well [23][24]. The data processing does not need to take place exclusively on serverless platform in the cloud, and instead can be migrated to the edge as well, optimizing bandwidth usage should the computing resources meet the required performance [25].

All these different workloads that have unpredictable load levels and need to cope efficiently with large increases in the number of requests emphasize the need for advanced resource allocation and scheduling algorithms that can better meet the FaaS quality of service (QoS) expectations during peaks [22]. A review of existing scheduling optimizations is offered in [26]. Even though it primarily focusses on the cloud, it is also relevant in network edge environments.

When it comes to the network edge, the authors of [27] argue that there are significant benefits to moving serverless computing to this part of the network, and that it should not be limited to the cloud environment only. The establishment of an edge–cloud continuum which would allow dynamic workload migration and be transparent to the end users would bring the best of both worlds, data preprocessing at the edge when reduced latency is needed, and the vast compute capacity of the cloud for further analysis and long term storage. Unfortunately, before establishing a true edge–cloud continuum, further research is needed into efficiency optimizations in terms of runtime environments, their performance at the edge, and the feasibility of on-the-fly data migration. Hellerstein et al. [15] outline all of the efficiency problems affecting first generation serverless implementations, such as the limited execution time of functions imposed by serverless platforms, slow first invocation of the functions, low performance of input/output (I/O) operations, and limited support for specialized hardware, such as graphics cards. Discussion about potential solutions to the initial start up delay is offered by Kratzke et al. in [28], while reviewing cloud application architectures. Apart from comparing the advantages and disadvantages of serverless, the utilization of unikernels is proposed as a more lightweight runtime environment for serverless function execution. However, in order to effectively test any performance improvements, adequate and standardized benchmarks are needed which would be capable of cross platform execution. The authors of [29] provide a review of existing efforts made to benchmark FaaS platforms.

Real-world serverless platforms that are ready to be used also play an important role in the serverless adoption across its different realms of usage, and they are responsible for implementing all the other advancements in terms of security, scheduling, and efficiency in a comprehensive, ready to use package. Bocci et al. provide [30] a systematic review of serverless computing platforms, focusing on supported languages, models, and methodologies to define FaaS orchestrations. Special attention is also given to security issues, but single node serverless platforms are purposefully excluded. In our opinion, even though not natively scalable, single node platforms are still a valuable resource and can act as a guidance in relevant platform development trends. In a future work they can be expanded to encompass multiple nodes or can serve as an inspiration to other platforms by repurposing individual components. Additional analysis, but in a wider context, reviewing general features of

existing popular serverless edge platforms is also available in [19], which can aid the decision making process when choosing a new serverless solution for the network edge.

Even though there are research papers that deal with serverless security and evaluate isolation levels of the various platforms available today [30], the analysis of Stack Overflow [31] questions related to FaaS products suggests that developers rarely concern themselves with such topics, focusing more on the implementation and functional aspects of their applications instead. Still, many serverless platforms mandate strong runtime isolation between different serverless functions, in part mitigating such security concerns, albeit leading to reduced performance, additional function non-portability, and vendor lock-in [21].

In conclusion, multiple reviews have identified serverless computing as an emerging technology with prospects of being utilized in a variety of different contexts, including IoT. However, to the best of our knowledge, no comprehensive review exists focusing primarily on serverless edge computing from an IoT perspective. In our opinion, IoT is not just another use-case for this new paradigm, instead it is the killer application with a great potential, should the identified open issues be solved.

## 3. Current Research Trends

*Application Implementation* is the topic with most published papers in the reviewed period, with 19 entries from the 64 analyzed entries in total. Even though serverless computing was initially targeted primarily at web developers to simplify the development process, recently novel use-cases have emerged demanding lower latency and the deployment of edge infrastructure. Serverless computing is especially suitable for event-driven scenarios [13] involving IoT devices. One such area is cyber-physical systems, where a successful implementation of a power grid monitoring solution capable of dynamically responding to unpredicted events and balancing supply according to current demand has been described [32]. Smart city applications like monitoring garbage disposal [33], energy usage optimization [34][35], or improving public transportation systems [36] have also been discussed. However, serverless computing at the edge can also be utilized without a dedicated infrastructure, by harvesting the computing power of nearby devices instead. Using portable JavaScript runtimes, the authors of [37] have created a system which can offload processing to devices in the close vicinity for an AR/VR application [38]. Reports on converting existing serverful applications to a serverless architecture have also been published [39], with the intention of driving a higher adoption and outlining the benefits. Nonetheless, a recent survey on Stack Overflow, analyzing questions related to the topic of serverless computing [31], shows that the majority of encountered problems by developers are related particularly to application implementation. To solve this and to drive a higher level of adoption, formal guidelines should be published educating developers about the limitations of the network edge.

*Efficiency* improvements have been made to serverless edge platforms, trying to overcome the fact that existing serverless platforms developed initially for environments with plentiful resources are not a good fit for the resource constrained edge. The focus of this research area is finding alternative runtime environments that do not rely on containerization, thus avoiding the slow start-up incurred during the first invocation of a given function. A promising

option is WebAssembly [40] with its portability and fast function start-up time [41], albeit further work is needed on improving the execution speed of the deployed functions. Alternatives include the introduction of unikernels, a surprisingly under researched topic today, and the development of micro virtual machines [28], with some implementations already being open-sourced [42].

*Scheduling* algorithms optimally determining where and when a given function needs to be executed [43] are another way in which the cold-start problem [44] typical for container based serverless systems can be overcome, apart from introducing new runtime environments. Further optimizations in terms of reduced latency [45], bandwidth [46], and cost [47] have also been described, depending on the use-case and priorities of the administrators. Recently, efforts have been made to develop alternative scheduling systems to popular serverless platforms, utilizing machine learning algorithms [48][49] with the aim of analyzing historical function metric data and adapting the scheduling decisions accordingly. However, scheduling decisions are not limited only to the initial placement of the functions, but can also be extended to live function migration, alleviating unexpected memory pressure, or dynamically pausing and then resuming function execution on the same node while waiting for a synchronous operation to complete [50].

*Benchmarks* can be used to measure and compare the performance of different efficiency optimizations, scheduling algorithms, and complete serverless platforms in terms of other alternatives. Multiple benchmarking suites have been proposed [51][52] to this effect, utilizing a number of different tests, ranging from purpose built microbenchmarks targeted at measuring raw compute, network, or I/O performance, to all encompassing serverless applications. Unfortunately, lacking a unified abstraction layer that would be supported across all serverless platforms, these benchmarking suites are limited in the number of environments that they support. The addition of a new supported platform is often a tedious process as a result of the different provider application programming interfaces (APIs) available or runtime restrictions. Researchers have attempted to solve this issue by open-sourcing their code and relying on the community to introduce support for popular solutions. This leads to problems where the majority of authors do publish performance results about their implementation, but they are hard to verify, replicate, and compare to other platforms that have not been included in their analysis.

*Platform Implementations* have decided to adopt the API interfaces of popular cloud-based serverless products [53] with the aim of solving the issue of vendor lock-in and cross-platform incompatibility, thus making all existing functions automatically compatible with the newly presented solution. The development of new serverless edge platforms using existing commercial solutions is not uncommon, and is mostly focused on features that are lacking by default. The authors of [54] extend the AWS Greengrass software to be able to automatically fetch AWS Lambda functions for local execution when there is such demand. This behavior is possible since both AWS Lambda and Greengrass support the same function languages and constructs. Others have instead focused on improving existing open-source serverless platforms and optimizing them for the network edge [38][55]. AI, as one popular use-case of serverless functions, has also incentivized the development of specialized platforms satisfying its requirements [56][57]. However, by offering easy-to-use interfaces, and integration with the cloud, it is possible to leverage the proximity of the edge not only for reduced latency, but also for increased privacy, to preprocess data that would ultimately be analyzed and aggregated in the cloud. This is especially useful for research studies that

gather various sensor data containing personally identifiable information, which needs to be anonymized first [25]. A persistent issue faced by all serverless edge platforms is how to connect with the end-users and end-devices who would invoke the available functions. With the continuous improvement in mobile network infrastructure and introduction of new generations of connectivity, the idea of collocating compute infrastructure with providers' base stations becomes a reality. The concept of mobile edge computing (MEC) [58], coupled with serverless can play an important role both for service providers and end-users alike [46]. By deploying serverless platforms capable of offering FaaS to prospective customers [43], operators can rent their in-place edge infrastructure, while enabling additional IoT use-cases without the need for standalone deployment of new compute or networking equipment.

*Continuum* describes a hierarchical execution environment comprised of edge, fog, and cloud resources, working in tandem with dynamic workload migration between them. Many serverless edge platforms are not limited to only running at the edge, instead their aim is to develop versatile products that can be run anywhere, at either the edge, fog, or cloud, offering the same function syntax across the whole network [59][60]. When coupled with intelligent scheduling algorithms that can automatically determine the optimal execution location, as opposed to relying on the administrator to make the right decision [61], a true edge-fog-cloud continuum [27] can be established. Attempts have been made to offer such continuums even for commercial products with both cloud and edge counterparts, but not providing a native integration between them [62].

*Security, Integrity, Policy* is one of the least researched serverless edge topics, even though it is of paramount importance, especially in multi-tenant environments where multiple customers share the same infrastructure for function execution. Careful attention is warranted to the level of isolation that the chosen runtime offers, as well as the behavior for serving new requests. Aiming to reduce the cold-start latency, many platforms forgo per-invocation isolation, instead reusing the same environment without clearing it and spawning a new one, leaving leftover files or processes [63]. Another problem with serverless execution in scenarios where multiple functions are chained together in a pipeline is the prospect of intermediate data corruption which would require the repeated execution of the whole pipeline to alleviate the problem. Lin et al. [64] describe an append-only system storing function inputs and results, allowing granular reexecution of downstream functions, without affecting the upstream ones in the pipeline, thus minimizing the effects of any data corruption as well as reducing the time needed for repair, with low performance overhead.

# 4. Discussion

It is evident that there is a large interest in employing serverless computing at the edge of the network, with various research topics tackled. **Figure 1** shows the primary category distribution of the selected papers, with the inclusion of review papers as well. The x-axis represents the percentage of all papers published in the given year. The y-axis represents the percentage of all papers which have a connection to the given category. Please note that the numbers on the y-axis do not add up to 100 per cent because one paper can be relevant to multiple categories. The color coding of the bubbles relates to the open-access policy of the papers, with green denoting that all associated papers within the given category are open-access and yellow representing mixed policy—both open-access and closed-access are present.

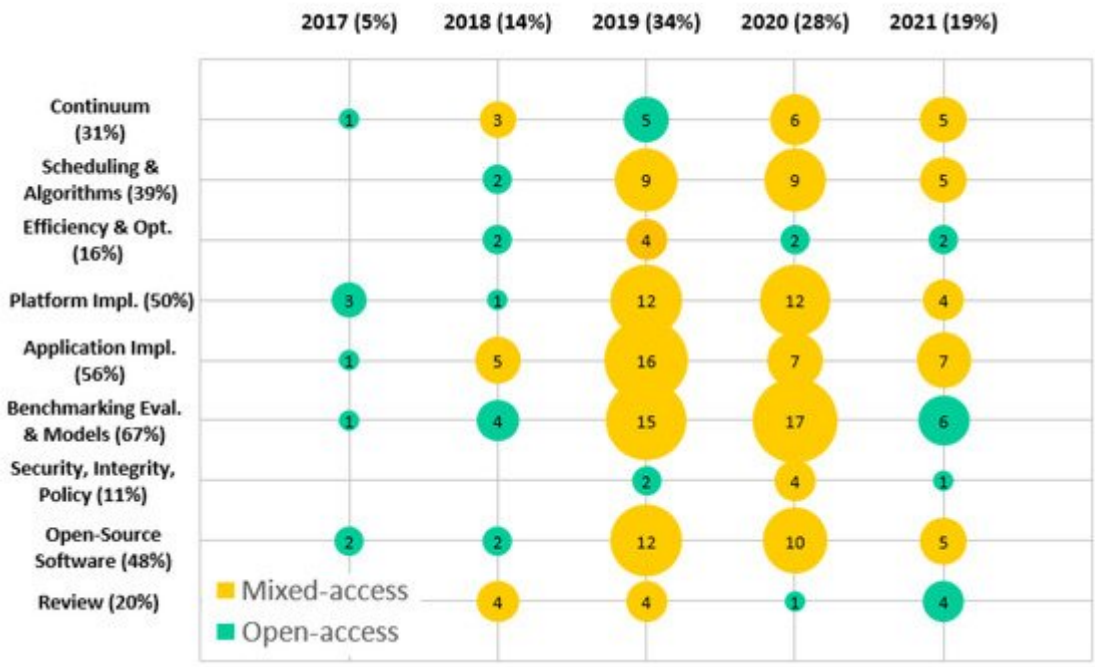


**Figure 1.** Primary category distribution per year and open-access classification.

The majority of analyzed papers (67 per cent) have been classified as offering some level of benchmarking and comparison with existing solutions, which is understandable taking into account the high representation of both platform implementation (50 per cent) and application implementation (56 per cent), two categories where performance discussion and comparison is commonplace. Open-source is also a highly popular category, accounting for 48 per cent of all entries, with many papers either basing their work on existing open-source code or publishing their implementation in turn. On the other hand, very few papers deal with the security aspects of using FaaS platforms, the integrity of the analyzed and produced data, or with policy in general, such as avoiding vendor lock-in problems.

A topic that is under active research especially is the past few years is the establishment of a true edge-fog-continuum. However, additional advancements are needed in the area of intelligent scheduling algorithms and efficiency optimizations before such erasure of network boundaries can become commonplace. We present a list of open issues which we deem need to be solved in order for serverless computing to achieve an even wider adoption at the edge of the network:

- Development of efficient scheduling algorithms that are capable of handling high volumes of function instantiations and deletions in short amounts of times, across different infrastructures, providing an edge–cloud continuum.

- Safe migration of running serverless functions across different environments, allowing for better resiliency and cost effectiveness.

- Performance improvement of existing serverless function runtimes to make them suitable for resource constrained devices located at the edge, and migration away from containerization technologies altogether, by adopting more lightweight alternatives, such as WebAssembly, and unikernels. However, further research is needed in terms of execution speed performance, and development of easy-to-use solutions, which would in turn lead to an increase in popularity.

- Eliminating the cold start problem associated with the dynamic nature of serverless functions and the scale-to-zero feature.

- Eliminating vendor lock-in, as a prerequisite for a wider adoption, as well as constructing more elaborate hierarchical infrastructures, which would include both commercial and private elements. This is also the main issue preventing the establishment of cross-platform function marketplaces where users can freely exchange existing serverless functions.

- Improvements to serverless function security and isolation, especially in multi-tenant environments. Even though security is of great concern for resource constrained IoT devices, innovative ways in which greater function isolation can be established, without resulting in increased execution or start-up time are needed. Exhaustion of resources as a result of ever more present denial of service attacks is also an open issue, especially for serverless functions utilizing a commercial platform, where billing is done depending on the number of invocations and the total runtime. An increase in denial of service attacks aiming to take a given service offline by incurring large monetary cost to its owners is not excluded.

- Improvements to function chaining, and shift to asynchronous execution where possible. One of the main benefits of serverless, the scale-down-to-zero feature, cannot be realized when a chain of subsequent functions is executed in a serial manner, all waiting for an intermediate result before they can be terminated. Not only does this lead to less efficient resource utiliziation, but also to increased cost, as a result of each function being billed independently, even when it is stuck waiting on another one.

- Lack of comprehensive guidelines for development of new serverless IoT applications, or migration of existing ones, taking into account the specifics of this new paradigm.

- Support for hardware-acceleration and utilization of specific hardware, essential for artificial intelligence and video processing workloads.

## References

1. Buyya, R.; Yeo, C.S.; Venugopal, S.; Broberg, J.; Brandic, I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 2009, 25, 599–616.

2. Duan, Y.; Fu, G.; Zhou, N.; Sun, X.; Narendra, N.C.; Hu, B. Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends. In Proceedings of the 2015 IEEE 8th International Conference on Cloud Computing, New York, NY, USA, 27 June–2 July 2015; pp. 621–628.
3. AWS Lambda—Serverless Compute—Amazon Web Services. Available online: https://aws.amazon.com/lambda/ (accessed on 27 May 2021).
4. Azure Functions Serverless Compute|Microsoft Azure. Available online: https://azure.microsoft.com/en-us/services/functions/ (accessed on 27 May 2021).
5. IBM Cloud Functions-Overview. Available online: https://www.ibm.com/cloud/functions (accessed on 27 May 2021).
6. Cloud Functions. Available online: https://cloud.google.com/functions (accessed on 27 May 2021).
7. Apache OpenWhisk Is a Serverless, Open Source Cloud Platform. Available online: https://openwhisk.apache.org/ (accessed on 27 May 2021).
8. Available online: https://www.openfaas.com/ (accessed on 27 May 2021).
9. Kubeless. Available online: https://kubeless.io/ (accessed on 27 May 2021).
10. Getting Started with IBM Cloud Functions. Available online: https://cloud.ibm.com/docs/openwhisk?topic=openwhisk-getting-started (accessed on 27 May 2021).
11. Azure/Iotedge. Available online: https://github.com/Azure/iotedge (accessed on 27 May 2021).
12. Gill, S.S.; Tuli, S.; Xu, M.; Singh, I.; Singh, K.V.; Lindsay, D.; Tuli, S.; Smirnova, D.; Singh, M.; Jain, U.; et al. Transformative effects of IoT, Blockchain and Artificial Intelligence on cloud computing: Evolution, vision, trends and open challenges. Internet Things 2019, 8, 100118.
13. Aslanpour, M.S.; Toosi, A.N.; Cicconetti, C.; Javadi, B.; Sbarski, P.; Taibi, D.; Assuncao, M.; Gill, S.S.; Gaire, R.; Dustdar, S. Serverless Edge Computing: Vision and Challenges. In 2021 Australasian Computer Science Week Multiconference; ACM: Dunedin, New Zealand, 2021; pp. 1–10.
14. Gadepalli, P.K.; Peach, G.; Cherkasova, L.; Aitken, R.; Parmer, G. Challenges and Opportunities for Efficient Serverless Computing at the Edge. In Proceedings of the 2019 38th Symposium on Reliable Distributed Systems (SRDS), Lyon, France, 1–4 October 2019; pp. 261–2615.
15. Hellerstein, J.M.; Faleiro, J.; Gonzalez, J.E.; Schleier-Smith, J.; Sreekanti, V.; Tumanov, A.; Wu, C. Serverless Computing: One Step Forward, Two Steps Back. arXiv 2018, arXiv:1812.03651.
16. AWS IoT Greengrass—Amazon Web Services. Available online: https://aws.amazon.com/greengrass/ (accessed on 27 May 2021).

17. IoT Hub|Microsoft Azure. Available online: https://azure.microsoft.com/en-us/services/iot-hub/ (accessed on 27 May 2021).
18. Varghese, B.; Buyya, R. Next generation cloud computing: New trends and research directions. Future Gener. Comput. Syst. 2018, 79, 849–861.
19. El Ioini, N.; Hästbacka, D.; Pahl, C.; Taibi, D. Platforms for Serverless at the Edge: A Review. In Advances in Service-Oriented and Cloud Computing; Zirpins, C., Paraskakis, I., Andrikopoulos, V., Kratzke, N., Pahl, C., El Ioini, N., Andreou, A.S., Feuerlicht, G., Lamersdorf, W., Ortiz, G., et al., Eds.; Springer International Publishing: Cham, Switzerland, 2020; Volume 1360, pp. 29–40.
20. Shafiei, H.; Khonsari, A.; Mousavi, P. Serverless Computing: A Survey of Opportunities, Challenges and Applications. arXiv 2019, arXiv:1911.01296.
21. Hassan, H.B.; Barakat, S.A.; Sarhan, Q.I. Survey on serverless computing. J. Cloud Comput. 2021, 10, 39.
22. Buyya, R.; Srirama, S.N.; Casale, G.; Calheiros, R.; Simmhan, Y.; Varghese, B.; Gelenbe, E.; Javadi, B.; Vaquero, L.M.; Netto, M.A.S.; et al. A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade. ACM Comput. Surv. 2019, 51, 1–38.
23. Kubeflow. Available online: https://www.kubeflow.org/ (accessed on 28 September 2021).
24. Argo Workflows—The Workflow Engine for Kubernetes. Available online: https://argoproj.github.io/argo-workflows/ (accessed on 28 September 2021).
25. Risco, S.; Moltó, G.; Naranjo, D.M.; Blanquer, I. Serverless Workflows for Containerised Applications in the Cloud Continuum. J. Grid Comput. 2021, 19, 30.
26. Adhikari, M.; Amgoth, T.; Srirama, S.N. A Survey on Scheduling Strategies for Workflows in Cloud Environment and Emerging Trends. ACM Comput. Surv. 2019, 52, 1–36.
27. Bittencourt, L.; Immich, R.; Sakellariou, R.; Fonseca, N.; Madeira, E.; Curado, M.; Villas, L.; DaSilva, L.; Lee, C.; Rana, O. The Internet of Things, Fog and Cloud continuum: Integration and challenges. Internet Things 2018, 3–4, 134–155.
28. Kratzke, N. A Brief History of Cloud Application Architectures. Appl. Sci. 2018, 8, 1368.
29. Scheuner, J.; Leitner, P. Function-as-a-Service performance evaluation: A multivocal literature review. J. Syst. Softw. 2020, 170, 110708.
30. Bocci, A.; Forti, S.; Ferrari, G.L.; Brogi, A. Secure FaaS orchestration in the fog: How far are we? Computing 2021, 103, 1025–1056.
31. Wen, J.; Chen, Z.; Liu, Y.; Lou, Y.; Ma, Y.; Huang, G.; Jin, X.; Liu, X. An empirical study on challenges of application development in serverless computing. In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the

Foundations of Software Engineering; Association for Computing Machinery: New York, NY, USA, 2021; pp. 416–428.

32. Zhang, S.; Luo, X.; Litvinov, E. Serverless computing for cloud-based power grid emergency generation dispatch. Int. J. Electr. Power Energy Syst. 2021, 124, 106366.
33. Al-Masri, E.; Diabate, I.; Jain, R.; Lam, M.H.; Reddy Nathala, S. Recycle.io: An IoT-Enabled Framework for Urban Waste Management. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 5285–5287.
34. Albayati, A.; Abdullah, N.F.; Abu-Samah, A.; Mutlag, A.H.; Nordin, R. A Serverless Advanced Metering Infrastructure Based on Fog-Edge Computing for a Smart Grid: A Comparison Study for Energy Sector in Iraq. Energies 2020, 13, 5460.
35. Huber, F.; Mock, M. Toci: Computational Intelligence in an Energy Management System. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 1–4 December 2020; pp. 1287–1296.
36. Herrera-Quintero, L.F.; Vega-Alfonso, J.C.; Banse, K.B.A.; Carrillo Zambrano, E. Smart ITS Sensor for the Transportation Planning Based on IoT Approaches Using Serverless and Microservices Architecture. IEEE Intell. Transp. Syst. Mag. 2018, 10, 17–27.
37. Salehe, M.; Hu, Z.; Mortazavi, S.H.; Mohomed, I.; Capes, T. VideoPipe: Building Video Stream Processing Pipelines at the Edge. In Proceedings of the 20th International Middleware Conference Industrial Track; ACM: Davis, CA, USA, 2019; pp. 43–49.
38. Baresi, L.; Filgueira Mendonça, D.; Garriga, M. Empowering Low-Latency Applications Through a Serverless Edge Computing Architecture. In Service-Oriented and Cloud Computing; De Paoli, F., Schulte, S., Broch Johnsen, E., Eds.; Springer International Publishing: Cham, Switzerland, 2017; Volume 10465, pp. 196–210.
39. Großmann, M.; Ioannidis, C.; Le, D.T. Applicability of Serverless Computing in Fog Computing Environments for IoT Scenarios. In Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion; Association for Computing Machinery: New York, NY, USA, 2019; pp. 29–34.
40. Hall, A.; Ramachandran, U. An execution model for serverless functions at the edge. In Proceedings of the International Conference on Internet of Things Design and Implementation; ACM: Montreal, QC, Canada, 2019; pp. 225–236.
41. Gadepalli, P.K.; McBride, S.; Peach, G.; Cherkasova, L.; Parmer, G. Sledge: A Serverless-first, Light-weight Wasm Runtime for the Edge. In Proceedings of the 21st International Middleware Conference; Association for Computing Machinery: New York, NY, USA, 2020; pp. 265–279.
42. Firecracker—Secure and Fast microVMs for Serverless Computing. Available online: https://firecracker-microvm.github.io/ (accessed on 27 May 2021).

43. Cicconetti, C.; Conti, M.; Passarella, A. Low-latency Distributed Computation Offloading for Pervasive Environments. In Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom), Kyoto, Japan, 11–15 March 2019; pp. 1–10.
44. Wang, I.; Liri, E.; Ramakrishnan, K.K. Supporting IoT Applications with Serverless Edge Clouds. In Proceedings of the 2020 IEEE 9th International Conference on Cloud Networking (CloudNet), Piscataway, NJ, USA, 9–11 November 2020; pp. 1–4.
45. Cicconetti, C.; Conti, M.; Passarella, A. A Decentralized Framework for Serverless Edge Computing in the Internet of Things. IEEE Trans. Netw. Serv. Manag. 2020.
46. Yang, S.; Xu, K.; Cui, L.; Ming, Z.; Chen, Z.; Ming, Z. EBI-PAI: Towards An Efficient Edge-Based IoT Platform for Artificial Intelligence. IEEE Internet Things J. 2020.
47. Elgamal, T. Costless: Optimizing Cost of Serverless Computing through Function Fusion and Placement. In Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, USA, 25–27 October 2018; pp. 300–312.
48. Wang, B.; Ali-Eldin, A.; Shenoy, P. LaSS: Running Latency Sensitive Serverless Computations at the Edge. In Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing; Association for Computing Machinery: New York, NY, USA, 2020; pp. 239–251.
49. Agarwal, S.; Rodriguez, M.A.; Buyya, R. A Reinforcement Learning Approach to Reduce Serverless Function Cold Start Frequency. In Proceedings of the 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Melbourne, Australia, 10–13 May 2021; pp. 797–803.
50. Karhula, P.; Janak, J.; Schulzrinne, H. Checkpointing and Migration of IoT Edge Functions. In Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking; ACM Press: Dresden, Germany, 2019; pp. 60–65.
51. Kim, J.; Lee, K. FunctionBench: A Suite of Workloads for Serverless Cloud Function Service. In Proceedings of the 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), Milan, Italy, 8–13 July 2019; pp. 502–504.
52. Das, A.; Patterson, S.; Wittie, M. EdgeBench: Benchmarking Edge Computing Platforms. In Proceedings of the 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), Zurich, Switzerland, 17–20 December 2018; pp. 175–180.
53. Wolski, R.; Krintz, C.; Bakir, F.; George, G.; Lin, W.T. CSPOT: Portable, multi-scale functions-as-a-service for IoT. In Proceedings of the 4th ACM/IEEE Symposium on Edge Computing; ACM: Arlington, VA, USA, 2019; pp. 236–249.
54. Quang, T.; Peng, Y. Device-driven On-demand Deployment of Serverless Computing Functions. In Proceedings of the 2020 IEEE International Conference on Pervasive Computing and

Communications Workshops (PerCom Workshops), Austin, TX, USA, 23–27 March 2020; pp. 1–6.

55. Ling, W.; Ma, L.; Tian, C.; Hu, Z. Pigeon: A Dynamic and Efficient Serverless and FaaS Framework for Private Cloud. In Proceedings of the 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 5–7 December 2019; pp. 1416–1421.
56. Rausch, T.; Hummer, W.; Muthusamy, V.; Rashed, A.; Dustdar, S. Towards a Serverless Platform for Edge AI. In Proceedings of the 2nd USENIX Workshop On Hot Topics In Edge Computing (HotEdge 19), Renton, WA, USA, 9 July 2019; Available online: https://www.usenix.org/conference/hotedge19/presentation/rausch (accessed on 27 May 2021).
57. Zhang, M.; Krintz, C.; Wolski, R. STOIC: Serverless Teleoperable Hybrid Cloud for Machine Learning Applications on Edge Device. In Proceedings of the 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Austin, TX, USA, 23–27 March 2020; pp. 1–6.
58. Cicconetti, C.; Conti, M.; Passarella, A.; Sabella, D. Toward Distributed Computing Environments with Serverless Solutions in Edge Systems. IEEE Commun. Mag. 2020, 58, 40–46.
59. Baresi, L.; Mendonça, D.F.; Garriga, M.; Guinea, S.; Quattrocchi, G. A Unified Model for the Mobile-Edge-Cloud Continuum. ACM Trans. Internet Technol. 2019, 19, 1–21.
60. Pinto, D.; Dias, J.P.; Sereno Ferreira, H. Dynamic Allocation of Serverless Functions in IoT Environments. In Proceedings of the 2018 IEEE 16th International Conference on Embedded and Ubiquitous Computing (EUC), Bucharest, Romania, 29–31 October 2018; pp. 1–8.
61. Luckow, A.; Rattan, K.; Jha, S. Pilot-Edge: Distributed Resource Management Along the Edge-to-Cloud Continuum. In Proceedings of the 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Portland, OR, USA, 17–21 June 2021; pp. 874–878.
62. Zhang, M.; Wang, F.; Zhu, Y.; Liu, J.; Wang, Z. Towards cloud-edge collaborative online video analytics with fine-grained serverless pipelines. In Proceedings of the 12th ACM Multimedia Systems Conference; Association for Computing Machinery: New York, NY, USA, 2021; pp. 80–93.
63. Datta, P.; Kumar, P.; Morris, T.; Grace, M.; Rahmati, A.; Bates, A. Valve: Securing Function Workflows on Serverless Computing Platforms. In Proceedings of The Web Conference 2020; ACM: Taipei, Taiwan, 2020; pp. 939–950.
64. Lin, W.T.; Bakir, F.; Krintz, C.; Wolski, R.; Mock, M. Data Repair for Distributed, Event-based IoT Applications. In Proceedings of the 13th ACM International Conference on Distributed and Event-Based Systems; ACM: Darmstadt, Germany, 2019; pp. 139–150.