Serverless Workflow Management on the Computing Continuum: A Mini-Survey

Reza Farahani
reza.farahani@aau.at
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria

Frank Loh
frank.loh@uni-wuerzburg.de
University of Würzburg
Würzburg, Germany

Dumitru Roman
dumitru.roman@sintef.no
Sintef
Oslo, Norway

Radu Prodan
radu.prodan@aau.at
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria

ABSTRACT

The growing desire among application providers for a cost model based on *pay-per-use*, combined with the need for a seamlessly integrated platform to manage the complex workflows of their applications, has spurred the emergence of a promising computing paradigm known as *serverless* computing. Although serverless computing was initially considered for cloud environments, it has recently been extended to other layers of the *computing continuum*, i.e., edge and fog. This extension emphasizes that the proximity of computational resources to data sources can further reduce costs and improve performance and energy efficiency. However, orchestrating the computing continuum in complex application workflows, including a set of serverless functions, introduces new challenges. This paper investigates the opportunities and challenges introduced by serverless computing for *workflow management systems* (WMS) on the computing continuum. In addition, the paper provides a taxonomy of state-of-the-art WMSs and reviews their capabilities.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**.

KEYWORDS

Workflow; Workflow Management Systems (WMS); Serverless Computing; Function-as-a-Service (FaaS); Edge-Cloud Continuum; Function Scheduling; Service Orchestration; Sustainability.

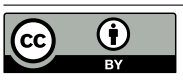
ACM Reference Format:
Reza Farahani, Frank Loh, Dumitru Roman, and Radu Prodan. 2024. Serverless Workflow Management on the Computing Continuum: A Mini-Survey. In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion), May 7–11, 2024, London, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3629527.3652901

1 INTRODUCTION

The proliferation of online applications, advancements in networking and computing technologies, and the continuously growing number of users who opt for diverse online services have collectively propelled application *workflow management systems* (WMS) to the forefront of discussions among various stakeholders [54]. In this context, an application workflow refers to cooperative tasks, activities, or processes that execute a specific business or computational logic. These tasks require network, computational, and storage resources beyond the capabilities of a single on-premises cluster. The emergence of cloud computing has revolutionized the domain of WMSs by offering scalable resources and diverse services. Most public cloud providers offer WMS, such as Amazon Simple Workflow Service and Google Cloud Composer, enabling application providers to build, deploy, schedule, and orchestrate their workflow tasks comprehensively. Although the adoption of cloud services represented a significant advancement, challenges persist in realizing a pure pay-per-use model and achieving seamless scalability. This is because cloud providers typically charge application owners based on allocated resources rather than actual consumption. Furthermore, application providers are burdened with ongoing responsibilities to configure and scale infrastructure instances, requiring comprehensive application monitoring and expertise in both infrastructure and services management [49].

To address the described challenges, both application and cloud providers made substantial architectural modifications. On the application side, the architecture transitioned from monolithic to service-oriented, then to microservices [14], and *Function-as-a-Service* (FaaS) [8], allowing the execution of small pieces of code as functions [46]. Taking into account the distinctive characteristics of emerging applications, particularly those with FaaS-based architectures, cloud providers have taken advantage of their previous experience, e.g., virtualization and containerization paradigms, to establish a pure pay-per-use paradigm known as *serverless* [53] as an alternative to the Infrastructure-as-a-Service (IaaS) model. Hence, cloud providers are increasingly adopting serverless principles across a spectrum of their existing services, encompassing serverless containers (e.g., AWS Fargate or Google Cloud Run), serverless databases (e.g., AWS DynamoDB), and even serverless graph processing (e.g., AWS Neptune). Furthermore, most cloud providers offer serverless WMSs, such as AWS Step Functions, Google Workflows, and IBM Composer.

In pursuit of cost efficiency, reduced latency, and energy conservation, both industry and academia have recently increased their



This work is licensed under a Creative Commons Attribution International 4.0 License.

ICPE '24 Companion, May 7–11, 2024, London, UK
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0445-1/24/05.
https://doi.org/10.1145/3629527.3652901

efforts to utilize fog or edge resources in close proximity to application users, thus, establishing serverless WMSs throughout the entire *computing continuum* [20]. To this end, open-source serverless platforms, e.g., Apache OpenWhisk [1] or OpenFaaS [38], have been developed to operate on on-promise or leased edge or fog instances. However, this presents a challenge, given that the serverless paradigm was originally designed for cloud environments, not accounting for such computational constraint instances. Hence, among numerous critical considerations, one key research question in designing serverless WMSs on the computing continuum is: "How can *service level objectives* (SLOs), e.g., latency and energy, and economic cost, be optimized by determining the strategic allocation of workflow functions, deciding which to execute on edge or fog instances and which to offload to the public cloud?" This mini-survey explores fundamental concepts, reviews the latest WMSs, and discusses the opportunities and challenges arising from integrating serverless paradigms into WMSs.

2 BACKGROUND

2.1 Computing Continuum

The computing continuum is a structural design consisting of various computing and storage resources with varying network bandwidth, interconnected in three layers: Cloud, Fog, and Edge [18, 23]. The *cloud layer* is supported by public providers, such as AWS, IBM, or Google, to provide large-scale infrastructure and a broad range of services. The *fog layer* presents computing capabilities in close proximity to data sources and user devices on a smaller scale. This involves utilizing less powerful devices with lower access latencies compared to cloud servers, typically located in network base stations (e.g., gNodeB in 5G). The *edge layer* consists of local servers and devices with limited resources, including sensors and actuators, equipped with computational capabilities. These are strategically placed at user locations to further minimize service latency. Although each layer can collaborate and exchange information in the execution of applications, it also possesses the ability to operate independently. Adopting this multi-layer architecture empowers application providers to utilize appropriate resources and services, consequently enhancing SLOs, energy, and economic cost compared to dependence on a singular layer of resources.

2.2 Serverless Computing

Serverless is an emerging computing paradigm designed for the deployment of FaaS applications and other services statelessly [3]. Serverless applications typically comprise a collection of stateless and atomic functions, commonly deployed within containers or encapsulated as Zip files. Upon invocation of the function by the application, a *cold start* occurs on the infrastructure side, requiring the deployment of the container from online repositories to the specified resource. In contrast, a *warm start* occurs when the requested function is pre-deployed on the computational resource, resulting in rapid initialization and execution. In such paradigms, functions interact and exchange data using platform services like databases, if necessary [28]. Therefore, it empowers application developers to build scalable and event-driven applications while incurring charges based on the execution time of functions (i.e., pay-per-use), in addition to any supplementary services utilized by these functions. Moreover, it eliminates the complexities associated with the provisioning and maintenance of resources, challenges commonly encountered in traditional cloud-based systems designed for monolithic applications [49]. For a comprehensive overview of the serverless lifecycle, we refer to literature [37].

2.3 Workflow Management Systems

Application workflows typically consist of multiple stages, each comprising a set of independent tasks. These workflows are commonly represented as *directed acyclic graphs* (DAGs), where nodes represent tasks and edges denote data dependencies. Task communication is generally based on shared file systems and task execution occurs only when all dependencies are satisfied [55]. A *workflow management system* (WMS) operates as a dedicated tool to execute and orchestrate such workflows in heterogeneous computing and storage resources, including local and cloud instances. Many WMSs, such as Pegasus [13], Airflow [22], Argo, AWS Step Functions, Google Workflows, or IBM Composer, plus open-source ones like LithOps [48], and PyWren [26], have been developed to facilitate the seamless execution and management of application workflows across diverse computing architectures with serverful or serverless models. Such WMSs typically receive an application DAG as input, actively monitor the progress of running tasks and available resources, and generate execution plans by mapping tasks to the existing computing resources to enforce rigorous adherence to data dependencies while simultaneously striving to minimize the overall execution time.

3 SERVERLESS WORKFLOW MANAGEMENT

The rising customer demands from major public cloud providers, such as AWS, Google, IBM, and Azure, for serverless platform integration, coupled with the increasing complexity of serverless workflows, have greatly boosted the popularity of serverless WMSs [12]. Statistics reveal a substantial six-fold increase in the adoption of serverless workflows in Azure between 2019 and 2022 [32]. Managing complex workflows, often involving multiple functions and adhering to specific SLO levels, presents a challenge that cannot be adequately addressed by a single cloud-based architecture with concurrent function execution limitations (e.g., 1,000 functions for AWS Lambda). Therefore, the prevalent adoption of open source serverless platforms such as OpenFaaS [38] or OpenWhisk [1] has become a common practice to equip the other two layers of the computing continuum with serverless capabilities. However, designing WMSs to meet requested SLO levels across various resources in the computing continuum while accounting for available resources and concurrency limitations presents a considerable and intricate challenge. In the following discussion, we categorize and review state-of-the-art WMSs into three distinct types.

3.1 Cloud-based WMSs

Currently, all leading cloud providers have established their proprietary serverless WMSs. For instance, AWS Step Functions, an Amazon serverless WMS, utilizes the JSON format to orchestrate workflow functions, employing various constructs for parallelization, data distribution, and conditional branches. Despite providing versatile constructs, both WMSs are limited in scalability as they

operate within the same cloud region of a single provider, preventing the use of other cloud regions in a federated cloud manner or the computing continuum. Google Workflows uses the YAML format to define control and data flows within workflows. Unlike the previously mentioned WMSs, it supports function execution through HTTP requests, allowing the deployment of serverless functions across any cloud region.

In recent years, various WMSs have been developed to execute workflows across single or multiple providers' regions. Spock [21] operates as a scalable and adaptive WMS, employing virtual machines and a serverless platform deployed in public clouds. Its objective is to distribute the execution of machine learning inference jobs to minimize SLO violations. Sequoya [51] is another that provides developers with multiple scheduling policies tailored to various Quality of Service (QoS) parameters. Once a function completes, it triggers successor functions either on the local server running OpenWhisk or on cloud servers. The Multi-Provider Serverless Computing (MPSC) framework [2] is one of the multi-cloud WMSs, aiming to optimize task allocation between local servers and cloud platforms, specifically AWS Lambda and IBM Cloud Functions. Hyperflow [36] is another WMS that enables the execution of workflow functions exclusively within a designated region of AWS Lambda or Google Cloud Functions. However, these WMSs mostly ignore the function concurrency limitations of cloud providers.

The literature has also introduced domain-specific serverless WMSs, designed to accelerate the development of serverless applications within specific domains. Examples include scientific workflows, which encompass complex and long-term data-intensive tasks [6, 25]. The mentioned WMSs use HyperFlow to construct WMSs and execute scientific workflows on AWS Lambda and Google Cloud Functions. Furthermore, numerous recent works aim to enhance the execution speed of workflows in public cloud environments [10, 30, 31, 33]. For this aim, Mahgoub et al. [31] introduced three levels of optimizations integrated on AWS Lambda, allocating the appropriate resources for each function invocation. They also introduced SONIC [30], which determines the optimal approach to pass data between various serverless functions, that is, local storage, direct passing, and remote storage.

3.2 Edge-Cloud Continuum-based WMSs

In the domain of edge-cloud WMSs, numerous works have focused mainly on specific tasks within WMSs, such as function scheduling [41, 50, 56, 58]. For instance, Aslanpour et al. proposes an energy-aware serverless scheduling method tailored for applications in edge computing in [4]. Two priority-based and zone-oriented algorithms improve the operational availability of bottleneck edge devices using "sticky offloading" and "warm scheduling" to optimize QoS metrics. Skippy [42] represents another container-based scheduling method within these types of WMS, strategically balancing trade-offs between data and computation exchange. It takes workload-specific compute requirements into account, including GPU acceleration, to optimize overall utilization. Numerous works, such as OSCAR [43], propose the offloading of functions to clouds when edge resources become overloaded. Skedulix is another system on the edge-cloud continuum [11] that offloads functions from OpenFaaS to AWS Lambda to minimize costs while adhering to deadline constraints. Similarly, Serverledge is a decentralized edge-cloud system [47] that runs serverless functions on edge devices and offloads them to cloud servers or neighboring edge instances in case of overload. However, such systems target small single-workflow scheduling due to the concurrency limitations of edge devices and a single cloud instance. Costless [17] is a framework designed to optimize the execution cost of single serverless workflow applications by dividing their execution between the edge layer and the cloud. However, it does not account for scheduling concurrent workflows. The authors of [29] investigated the placement of workflow functions in edge-cloud systems to only minimize completion time.

3.3 Simulations-based WMSs

Addressing the need for proactive performance evaluation and prediction in serverless WMSs, many dedicated simulators have recently been developed. These simulators not only aid in assessing performance or cost metrics before deployment and execution [24, 34, 45, 52], but strive to provide valuable predictions [5, 16] to serverless providers regarding diverse load and request patterns. The authors in [34] simulate serverless functions with a fixed memory setup in a single cloud region and model the average response time of the functions, cold start, and concurrent instances of the serverless function. SimLess [45] is another serverless simulator that assesses the overhead of individual functions, as well as the entire workflow, and their comparable setups in federated clouds. DFaaSCloud [24] as an extension of CloudSim [7] and OpenDC Serverless [27] are simulators specifically designed to simulate the execution time of workflow functions. Faas-sim [40] is an edge-cloud simulator offering a versatile serverless simulation environment backed by real-world trace data.

4 OPPORTUNITIES

4.1 Cost Model

Unforeseeable variations in application workloads pose a challenge when using fixed container provisioning, resulting in charges during inactive periods. While dynamic auto-scaling is an option for most IaaS providers, it introduces additional costs and the potential for imprecise resource provisioning. On the contrary, serverless computing charges are determined by actual triggered events, including dedicated resources and function invocation frequency. The serverless paradigm ensures more predictable pricing irrespective of workload fluctuations. By leveraging the precise scalability of serverless, avoiding unnecessary resource allocation and idle-time costs, the overall price remains unaffected by workload variability.

4.2 Scalability

Certain applications operate consistently in close proximity to data sources on the edge layer of the computing continuum, while several others require seamless integration throughout the computing continuum. Leveraging the execution of serverless functions on the computing continuum enables WMSs to carry out these functions optimally. For instance, many serverless WMSs operating within the computing continuum adopt a straightforward service execution strategy that involves task execution on edge instances as long as resources are available, with a subsequent transition to

offloading tasks to the fog or cloud when needed. Furthermore, concurrent function executions through parallelization techniques not only enhance the practicality but also boost the scalability of WMSs within the computing continuum.

4.3 Auto-scaling

Traditional IaaS infrastructure relying on virtual machines faced drawbacks such as large memory footprints and challenging scalability, involving duplication of significant data when creating more service replicas on an instance. Therefore, one of the critical challenges for IaaS-based resource-limited edge and fog layers lies in resource management. Embracing lightweight abstractions like containers, serverless solutions offer a smaller footprint and precise autoscaling. This efficiency is particularly notable because of the minimal overhead in creating or terminating replicas compared to full virtual machines. The promise is further enhanced when serverless adopts computation principles based on functions inherited from microservice architecture advancements instead of treating the entire application as a black box.

4.4 Statelessness

The statelessness feature of serverless architecture offers several advantages over a serverful architecture. Since each function or service operates independently without maintaining a persistent state between invocations, fault tolerance and resilience improved, since failures in one function do not impact the overall system. Moreover, it makes serverless platforms such as WMSs appealing for real-time collaboration tools such as instant messaging and chatbots [57]. Therefore, it enables WMSs to have more suitable performance, particularly for applications with inherent lack reliance on and awareness of previous event, compared to the serverful ones.

5 CHALLENGES

5.1 Stream Processing

While the serverless paradigm is widely acknowledged as a successful advancement for cloud computing, the scale-to-zero technique and the subsequent cold start of serverless functions may not be optimal for certain latency-sensitive stream processing applications [19, 37]. The ability to scale to zero has both advantages and drawbacks. Although it minimizes energy consumption and reduces economic costs, initial invocation of the function results in a cold start, introducing additional delays with current technologies. Although this delay poses no issue for batch applications, it can lead to performance degradation in stream time-sensitive processing applications, since they require real-time decision-making, particularly on resource-limited devices. This discrepancy has led to active exploration within the research community, with studies analyzing its impacts [15, 44] and proposing various promising solutions, such as reducing overhead in the development phase [53].

5.2 Data Distribution

Data and storage management are key in serverless WMSs, irrespective of the computational and network differences among computing continuum instances. The stateless nature of the serverless, coupled with a lack of server affinity, gives rise to challenges. While cloud-based WMSs uphold function states by storing them in storage, the proximity of maintaining this state becomes pivotal at the edge, consequently, the edge layer transmitting substantial data per function invocation to remote devices. Current serverless WMSs within the edge-cloud continuum tend to prioritize CPU and memory specifications while neglecting crucial storage provisioning techniques. Imagine a scenario where an application involves a sequence of function invocations. In this case, the transfer cost is incurred only once in traditional approaches for monolithic applications. However, with serverless, this cost may be incurred multiple times between any two consecutive function invocations if executed on different layers. In the cloud, these limitations are seamlessly addressed by robust data centers and a high-speed communication. In contrast, in the edge layer of the computing continuum, data transport becomes problematic [3, 35]. Although numerous research studies have been initiated that focus on data caching and storage placement have been initiated to address this problem [9, 39], we believe that more research is needed to fully address this challenge.

5.3 Complexity

The lack of effective abstractions for managing task-based workloads on serverless platforms, especially for workflows that exhibit intricate structures and dependencies, necessitates manual partitioning and encapsulation of workflow function codes. In addition, despite the overarching promise of serverless to diminish the need for manual resource management, it still requires the configuration and adjustment of resource-related parameters for application workflows, such as concurrency and memory per container. The mentioned requirements add complexity for the application providers. In addition, multiple layers of infrastructure within the computing continuum, coupled with middleware and execution engines, pose challenges in monitoring, understanding, and predicting application performance. Therefore, more research activities are needed to alleviate these complications.

6 CONCLUSION

This mini-survey explores relevant technologies and assesses the capabilities of serverless workflow management systems. We discuss the appropriateness of serverless WMSs on the computing continuum due to their (1) potential for seamless integration from cloud to edge, (2) provision of pure pay-per-use and cost-effective design, (3) mitigation of challenges related to resource provisioning and scaling, (4) facilitation of easy parallelization for stateless functions, and (5) provision of fine-grained scalability for resources. However, the design of such WMSs for today's applications faces significant challenges, including (1) prolonged latencies induced by cold startups, (2) difficulties in handling stream processing workflows, (3) complexities in managing data distribution and storage, (4) the intricate task of designing cost-, SLO-, and energy-aware systems, and (5) the potential to induce resource inefficiencies that necessitate collaborative efforts from both academia and industry.

ACKNOWLEDGMENTS

Graph-Massivizer receives funding from the Horizon Europe research and innovation program of the European Union. Its grant management number is 101093202.

REFERENCES

[1] 2023. OpenWhisk. https://openwhisk.apache.org/ Accessed: 2023-11-20.

[2] Austin Aske and Xinghui Zhao. 2018. Supporting Multi-Provider Serverless Computing on the Edge. In *Workshop Proceedings of the 47th International Conference on Parallel Processing (ICPP Workshops '18)*. ACM.

[3] Mohammad S Aslanpour et al. 2021. Serverless Edge Computing: Vision and Challenges. In *Proceedings of the 2021 Australasian Computer Science Week Multiconference*.

[4] Mohammad Sadegh Aslanpour et al. 2022. Energy-Aware Resource Scheduling for Serverless Edge Computing. In *2022 22nd IEEE Intl. Symp. on Cluster, Cloud and Internet Computing*. IEEE.

[5] Barcelona-Pons et al. 2021. Benchmarking Parallelism in FaaS platforms. *Future Generation Computer Systems* (2021).

[6] Krzysztof Burkat et al. 2021. Serverless Containers-Rising Viable Approach to Scientific Workflows. In *2021 IEEE 17th International Conference on eScience (eScience)*. IEEE.

[7] Rodrigo N Calheiros et al. 2011. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and experience* (2011).

[8] Paul Castro et al. 2019. The Rise of Serverless Computing. *Commun. ACM* (2019).

[9] Chen Chen et al. 2023. S-Cache: Function Caching for Serverless Edge Computing. In *Proceedings of the 6th International Workshop on Edge Systems, Analytics and Networking*.

[10] Anirban Das et al. 2020. Performance Optimization for Edge-Cloud Serverless Platforms via Dynamic Task Placement. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE.

[11] Anirban Das et al. 2020. Skedulix: Hybrid Cloud Scheduling for Cost-Efficient Execution of Serverless Applications. In *2020 IEEE 13th Intl. Conf. on Cloud Computing*. IEEE.

[12] Datadog. 2023. Datadog. The State of Serverless, August 2023. https://www.datadoghq.com/state-of-serverless/ Accessed: 2023-11-20.

[13] Ewa Deelman et al. 2015. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems* (2015).

[14] Paolo Di Francesco et al. 2019. Architecting with Microservices: A Systematic Mapping Study. *Journal of Systems and Software* (2019).

[15] Klimentina Djeparoska and Marjan Gusev. 2023. Limitations of AWS and GCP Serverless Functions. In *2023 31st Telecommunications Forum (TELFOR)*. IEEE.

[16] Simon Eismann et al. 2020. Predicting the Costs of Serverless Workflows. In *Int. Conf. on Performance Engineering*. ACM.

[17] Tarek Elgamal et al. 2018. Costless: Optimizing Cost of Serverless Computing through Function Fusion and Placement. In *2018 IEEE/ACM Symp. on Edge Computing*. IEEE.

[18] Reza Farahani et al. 2023. Towards Sustainable Serverless Processing of Massive Graphs on the Computing Continuum. In *Proc. of the 1st Workshop on Serverless, Extreme-Scale, and Sustainable Graph Processing Systems*.

[19] Marios Fragkoulis et al. 2023. A survey on the evolution of stream processing systems. *The VLDB Journal* (2023).

[20] Sukhpal Singh Gill et al. 2024. Modern computing: vision and challenges. *Telematics and Informatics Reports* (2024).

[21] Jashwant Raj Gunasekaran et al. 2019. Spock: Exploiting Serverless Functions for SLO and Cost Aware Resource Procurement in Public Cloud. In *2019 IEEE 12th Intl. Conf. on Cloud Computing*. IEEE.

[22] Scott Haines. 2022. Workflow Orchestration with Apache Airflow. In *Modern Data Engineering with Apache Spark: A Hands-On Guide for Building Mission-Critical Streaming Applications*. Springer.

[23] Matthijs Jansen et al. 2023. The SPEC-RG Reference Architecture for the Compute Continuum. In *2023 IEEE/ACM 23rd Intl. Symp. on Cluster, Cloud and Internet Computing*. IEEE.

[24] Hongseok Jeon et al. 2019. A CloudSim-Extension for Simulating Distributed Functions-as-a-Service. In *International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*.

[25] Aji John et al. 2019. SWEEP: Accelerating Scientific Research Through Scalable Serverless Workflows. In *IEEE/ACM International Conference UCC Companion*. ACM.

[26] Eric Jonas et al. 2017. Occupy the Cloud: Distributed Computing for the 99%. In *Proceedings of the 2017 symposium on cloud computing*.

[27] S Jounaid. 2020. OpenDC Serverless: Design, Implementation and Evaluation of a FaaS Platform Simulator. Ph.D. thesis, Vrije Universiteit Amsterdam.

[28] Samuel Kounev et al. 2023. Serverless Computing: What It Is, and What It Is Not? *Commun. ACM* (2023).

[29] Liuyan Liu et al. 2019. Dependent Task Placement and Scheduling with Function Configuration in Edge computing. In *Proc. of the Intl. Symp. on Quality of Service*.

[30] Ashraf Mahgoub et al. 2021. {SONIC}: Application-aware Data Passing for Chained Serverless Applications. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. 285–301.

[31] Ashraf Mahgoub et al. 2022. {ORION} and the Three Rights: Sizing, Bundling, and Prewarming for Serverless {DAGs}. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*.

[32] Ashraf Mahgoub et al. 2022. WISEFUSE: Workload Characterization and DAG Transformation for Serverless Workflows. *Proc. of the ACM on Measurement and Analysis of Computing Systems* (2022).

[33] Nima Mahmoudi et al. 2019. Optimizing Serverless Computing: Introducing an Adaptive Function Placement Algorithm. In *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*.

[34] Nima Mahmoudi and Hamzeh Khazaei. 2021. SimFaaS: A Performance Simulator for Serverless Computing Platforms. In *Int. Conf. on Cloud Computing and Services Science*.

[35] Redowan Mahmud et al. 2020. Application Management in Fog Computing Environments: A Taxonomy, Review and Future Directions. *ACM Computing Surveys (CSUR)* (2020).

[36] Maciej Malawski et al. 2020. Serverless Execution of Scientific Workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions. *Future Generation Computer Systems* (2020).

[37] Kien Nguyen et al. 2023. Serverless Computing Lifecycle Model for Edge Cloud Deployments. In *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE.

[38] OpenFaaS. 2023. OpenFaaS. https://www.openfaas.com/ Accessed: 2023-11-20.

[39] Li Pan et al. 2022. Retention-Aware Container Caching for Serverless Edge Computing. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE.

[40] Philipp Raith et al. 2023. faaS-sim: A Trace-Driven Simulation Framework for Serverless Edge Computing Platforms. *Software: Practice and Experience* (2023).

[41] Thomas Rausch et al. 2021. Optimized Container Scheduling for Data-Intensive Serverless Edge Computing. *Future Generation Computer Systems* (2021).

[42] Thomas Rausch et al. 2021. Optimized Container Scheduling for Data-Intensive Serverless Edge Computing. *Future Generation Computer Systems* (2021).

[43] Sebastián Risco et al. 2021. Serverless Workflows for Containerised Applications in the Cloud Continuum. *Journal of Grid Computing* (2021).

[44] Sashko Ristov et al. 2022. Colder than the Warm Start and Warmer than the Cold Start! Experience the Spawn Start in FaaS Providers. In *Proceedings of the 2022 Workshop on Advanced tools, programming languages, and PLatforms for Implementing and Evaluating algorithms for Distributed systems*.

[45] Sashko Ristov et al. 2022. SimLess: simulate serverless workflows and their twins and siblings in federated FaaS. In *Proceedings of the 13th Symposium on Cloud Computing*.

[46] Sashko Ristov et al. 2023. Large-scale Graph Processing and Simulation with Serverless Workflows in Federated FaaS. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering*.

[47] Gabriele Russo Russo et al. 2023. Serverledge: Decentralized Function-as-a-Service for the Edge-Cloud Continuum. In *2023 IEEE Intl. Conf. on Pervasive Computing and Communications*. IEEE.

[48] Josep Sampe et al. 2021. Outsourcing Data Processing Jobs with Lithops. *IEEE Transactions on Cloud Computing* (2021).

[49] Hossein Shafiei et al. 2022. Serverless Computing: A survey of Opportunities, Challenges, and Applications. *Comput. Surveys* (2022).

[50] Yang Tang et al. 2020. Lambdata: Optimizing Serverless Computing by Making Data Intents Explicit. In *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*.

[51] Ali Tariq et al. 2020. Sequoia: Enabling Quality-of-Service in Serverless Computing. In *Proceedings of the 11th ACM symposium on cloud computing*.

[52] Erwin van Eyk. 2019. SimFaaS. https://github.com/erwinvaneyk/simfaas. Accessed: 2024-02-07.

[53] Erwin Van Eyk et al. 2018. Serverless Is More: From PaaS to Present Cloud Computing. *IEEE Internet Computing* (2018).

[54] Laurens Versluis and Alexandru Iosup. 2021. A survey of domains in workflow scheduling in computing infrastructures: Community and keyword analysis, emerging trends, and taxonomies. *Future generation computer systems* (2021).

[55] Fuhui Wu et al. 2015. Workflow scheduling in cloud: a survey. *The Journal of Supercomputing* (2015).

[56] Song Wu et al. 2021. Container Lifecycle-Aware Scheduling for Serverless Computing. *Software: Practice and Experience* (2021).

[57] Mengting Yan et al. 2016. Building a Chatbot with Serverless Computing. In *Proceedings of the 1st International Workshop on Mashups of Things and APIs*.

[58] Hanfei Yu et al. 2021. Harvesting Idle Resources in Serverless Computing via Reinforcement Learning. *arXiv preprint arXiv:2108.12717* (2021).