Towards Sustainable Serverless Processing of Massive Graphs on the Computing Continuum

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

Dragi Kimovski
dragi.kimovski@aau.at
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria

Sashko Ristov
sashko.ristov@uibk.ac.at
University of Innsbruck
Innsbruck, Austria

Alexandru Iosup
a.iosup@vu.nl
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands

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

ABSTRACT

With the ever-increasing volume of data and the demand to analyze and comprehend it, graph processing has become an essential approach for solving complex problems in various domains, like social networks, bioinformatics, and finance. Despite the potential benefits of current graph processing platforms, they often encounter difficulties supporting diverse workloads, models, and languages. Moreover, existing platforms suffer from limited portability and interoperability, resulting in redundant efforts and inefficient resource and energy utilization due to vendor and even platform lock-in. To bridge the aforementioned gaps, the *Graph-Massivizer* project, funded by the Horizon Europe research and innovation program, conducts research and develops a high-performance, scalable, and sustainable platform for information processing and reasoning based on the *massive graph* (MG) representation of extreme data. In this paper, we briefly introduce the Graph-Massivizer platform. We explore how the emerging *serverless computing* paradigm can be leveraged to devise a scalable graph analytics tool over a codesigned *computing continuum* infrastructure. Finally, we sketch seven crucial research questions in our design and outline three ongoing and future research directions for addressing them.

CCS CONCEPTS

• **Theory of computation → Graph algorithms analysis**; • **Computer systems organization → Cloud computing**.

KEYWORDS

Computing Continuum; Serverless Computing; Graph Processing; Massive Graph; Sustainability.

ACM Reference Format:
Reza Farahani, Dragi Kimovski, Sashko Ristov, Alexandru Iosup, and Radu Prodan. 2023. Towards Sustainable Serverless Processing of Massive Graphs on the Computing Continuum . In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3578245.3585331

ICPE '23 Companion, April 15–19, 2023, Coimbra, Portugal

ACM ISBN 979-8-4007-0072-9/23/04.
https://doi.org/10.1145/3578245.3585331

1 INTRODUCTION

Graphs are data structures consisting of vertices (*i.e.*, nodes) and edges (*i.e.*, relationships) between these vertices, which are commonly used to represent a wide variety of real-world data in various fields, *e.g.*, computer science, mathematics, physics, bioinformatics, and engineering. *Graph processing* refers to using algorithms to analyze and manipulate data represented as graphs and perform a series of tasks, *e.g.*, finding the shortest path between two vertices, grouping vertices into clusters, identifying patterns in the data, and making predictions based on the graphs' structure [18]. The parallel and distributed nature of graph processing enables the processing of *massive graphs* (MG), making it possible to handle extreme data and solve complex problems at scale.

Graph processing platforms often employ high-performance computing (HPC) or cloud computing resources (*i.e.*, storage, computation) to offer their users a range of graph processing and analysis services. Although numerous cloud- or HPC-based graph processing platforms exist, they always encounter difficulties in supporting *scalable*, *latency-*, *resource-*, *energy-aware*, and *cost-efficient* services. This is because graph processing platforms must be able to cater to the diverse needs of their users, who may have different requirements for graph processing and analysis depending on the specific use case. Consequently, carrying out graph processing on the computing continuum environment with a diverse range of devices, *e.g.*, HPC, cloud, and local devices, for satisfying various users' requirements is a step forward [2, 18, 20].

As shown in Fig. 1, the computing continuum is not limited to only cloud servers. It encompasses a range of devices in *three* layers, so-called *cloud*, *fog*, and *edge*, where these layers are connected and used together seamlessly to provide an incorporated computing experience. Thousand of powerful remote servers (in terms of computation and storage resources) are in the cloud layer, where data and applications can be directly stored on them rather than on local devices. Indeed, this allows users to access computing resources from anywhere, anytime, employing any device with an Internet connection. The fog layer extends cloud computing services by placing millions of servers with limited resources compared to the cloud layer in close proximity to users. In fact, the computing continuum architecture leverages this layer to address the challenges

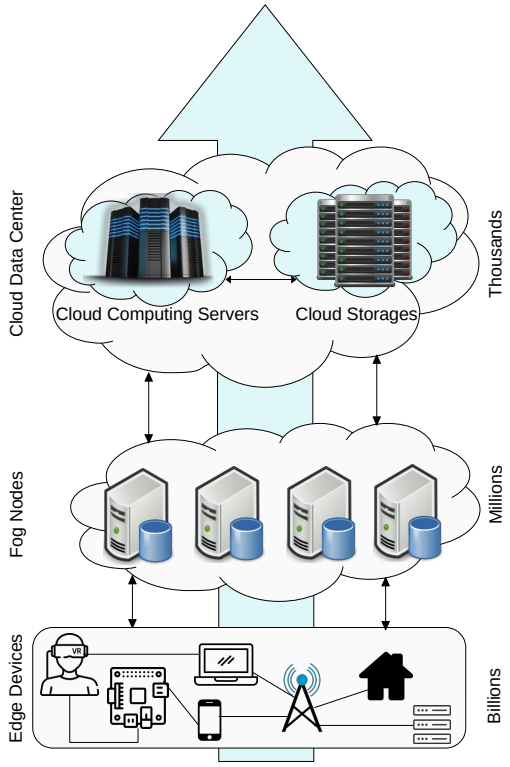

Figure 1: Illustration of the computing continuum

of cloud computing, such as latency and bandwidth constraints, by bringing computing resources closer to the source of the data. In addition, billions of edge devices (*e.g.*, traditional computers, servers, or Internet of things (IoT) devices) can be incorporated to provide real-time data processing and deliver faster and more responsive computing experiences [25]. Although the computing continuum allows graph processing platforms to process various graph categories over heterogenous environments, there are still burdens to accomplishing a *pure pay-per-use* and effortless scalability.

To address the abovementioned issues, modern computing continuum infrastructures are equipped with an emerging paradigm, so-called *serverless computing* [1]. Utilizing new techniques such as *microservices*, *Function-as-a-Service* (FaaS) [9], *event-driven* programming, and *containerization* enables serverless systems to work based on simple and small *functions*. Therefore, the operational concerns are abstracted to developers, and ease-of-use, fine-grained scalability, automation of server management, and pay-per-use billing instead of subscription are provided in such systems. [36]. However, using serverless techniques, which rely on stateless functions, poses numerous challenges when it comes to processing graphs, as inherently extreme data nature contradicts the stateless concept. Therefore, designing *serverless graph processing platforms on the computing continuum* requires careful investigation of the following fundamental research questions (RQs):

RQ1 **State Management:** How can states be effectively managed by a serverless computing continuum system where functions are stateless, and graph data requires to be stored in a (remote) memory?

RQ2 **Scalability:** How can the system's scalability be ensured when processing different kinds of graphs (in terms of type and size) in a serverless computing continuum environment?

RQ3 **Diversity:** How can a serverless computing continuum system be designed to support a broad class of iterative graph algorithms and handle both directed and undirected, weighted and unweighted graphs that can change at runtime?

RQ4 **Efficiency** How can the uncertainty of latencies and resource utilization in a serverless computing continuum system be mitigated to ensure the efficient processing of MG?

RQ5 **Sustainability:** How can a serverless computing continuum system be devised to minimize energy consumption and carbon footprint while maximizing performance and scalability?

RQ6 **Automation:** How can a serverless computing continuum system be fully automated to meet diverse graph processing developers' requirements with minimal resource management knowledge?

RQ7 **Cost-Efficiency:** How can the cost of executing graph processing functions in a serverless computing continuum system be optimized to balance performance and affordability?

The *Graph-Massivizer* project, funded by the Horizon Europe research and innovation program of the European Union, aims to tackle the limitations of existing graph processing platforms and draws inspiration from innovative computing paradigms over the computing continuum. The project aims to develop a high-performance, scalable, gender-neutral, secure, and sustainable MG platform [30]. In this paper, we leverage the serverless paradigm, introduce one of the *Graph-Massivizer* software tools, *i.e.*, `Serverlizer`, and discuss how `Serverlizer` can address the aforementioned RQs.

The remainder of the paper is organized as follows. Section 2 surveys state-of-the-art works. We briefly explain Graph-Massivizer conceptual architecture and introduce its associated tools in Section 3. We elaborate on the details of the proposed serverless solution and our ongoing and future works in Section 4. Finally, Section 5 concludes the paper.

2 RELATED WORK

This section reviews the state-of-the-art graph processing solutions in two categories.

2.1 Current Graph Procesing Platforms

The adaptability of graph data structures makes them suitable formats for representing complex systems and relationships. Thus, graph processing is a valuable tool for many applications and Use cases. Use cases include recommendation systems [10], social network analysis [11], network optimization [18], fraud detection [24], and bioinformatics [28]. Graph processing algorithms have unique natures, *e.g.*, computation-intensive and unpredictable access patterns, which make them well-suited for running on complex environments such as HPC systems and cloud-based environments. Therefore, several large-scale graph processing platforms,

e.g., *Apache Giraph* [3, 26], *Apache Flink* [8], *GraphMat* [33], and *GraphX* [15], are introduced.

Despite the famous graph processing platforms' benefits, most encounter scalability issues when dealing with extremely massive and complex graphs. Since they generally run on a fixed number of compute nodes, they cannot adapt to variations in workload by either scaling in or scaling out resources; consequently, they face performance limitations. In addition, these platforms make complexity for users who require more expertise in specific frameworks, such as *Apache Hadoop* [19]. Moreover, since most of these platforms view large-scale graph processing as only relevant for large organizations, the potential for cost-efficient deployment options is ignored in their designs [19]. As a result, there has been growing needs and interests to design and develop graph processing platforms that enhance the performance and capabilities of current systems while addressing their limitations.

2.2 Serverless Graph Processing Platforms

Inspired by the issues outlined in Section 2.1, academia and industry have started to adopt the promising serverless computing paradigm to design easy-to-use, scalable, resource- and cost-efficient graph processing platforms. Several pioneering works in the literature, like [21, 23, 27], employed serverless computing to present (big) data processing systems. However, none of them provides support for graph processing applications. Szalay *et al.* [34] employ resource partitioning and centralized cluster-level heuristics to schedule latency and throughput-sensitive serverless applications but do not take into account graph processing operations explicitly. Aslanpour *et al.* [5] presents energy-aware scheduling for serverless edge computing. However, their solution is focused on edge devices and does not provide a graph processing platform that operates across the entire computing continuum nodes.

Heidari *et al.* [17] suggest a cost-effective auto-scaling method that adjusts the number and types of virtual machines according to the computation needs of convergent graph applications. However, this approach is incompatible with serverless paradigms since it does not consider the fundamental principles of serverless, *i.e.*, containerization and FaaS. Current serverless platforms, such as *Apache OpenWhisk* [4], rely on deployment frameworks like *Kubernetes* and *Docker Swarm*, which utilize greedy decision-making techniques, such as filtering nodes and are unable to host a given function and establish a rating among the rest. The authors in [35] propose a serverless graph processing system called Graphless, which is implemented with AWS Lambda. Graphless allows graph processing functions to be deployed using push or pull operations on a set of predefined worker resources utilizing static and super-step schedulers. However, sustainability has not been considered in Graphless design. In addition, the authors demonstrate that the efficiency of Graphless can be degraded due to the variability of network characteristics under communication-intensive workloads.

In contrast to the state-of-the-art solutions, the `Serverlizer` tool aims to enable scalable and sustainable serverless execution of MG processing over extreme data, combining sustainability and performance metrics (*e.g.*, greenhouse gases (GHG) emissions, processing, and data throughput). It also enables processing MG over

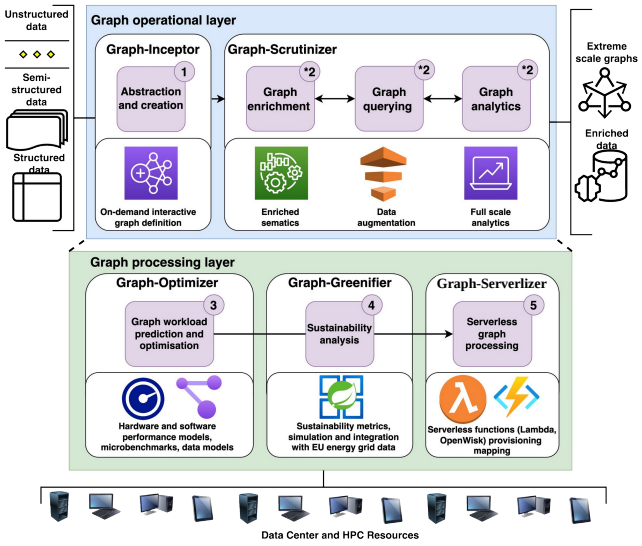

Figure 2: Graph-Massivizer architecture.

a vast set of heterogeneous resources across the computing continuum, comprising HPC, cloud, fog, edge, and specialized hardware.

3 GRAPH-MASSIVIZER PLATFORM

Graph-Massivizer develops, integrates, deploys, and validates a novel integrated toolkit for sustainable development and operation of MG processing on extreme data. The conceptual architecture of the Graph-Massivizer, including three layers, *i.e.*, *(i) graph operational layer*, *(ii) graph processing layer*, and *(ii) hardware infrastructure layer*, is depicted in Fig. 2.

The graph operational layer facilitates generating, transforming, and manipulating extreme data through *basic graph operations* (BGO), comprising graph creation, enrichment, query, and analytics. The following components are placed in this layer:

① **Graph creation:** implemented by the *Graph-Inceptor* tool translates extreme data from various static and event streams or follows heuristics to generate synthetic data, persist it, or publish it within a graph structure.

② **Graph enrichment, Graph query, and Graph analytics:** three BGO implemented by the *Graph-Scrutinizer* tool. They analyse and expand extreme datasets using probabilistic reasoning and ML algorithms for graph pattern discovery, low memory footprint graph generation, and low latency error-bounded query response. The output of this phase is a new graph, a query, or an enriched structured dataset.

Following three phases, the graph processing layer provides sustainable and energy-aware serverless graph analytics on the underlying heterogeneous HPC infrastructure. This layer includes the following entities:

③ **Graph workload prediction and optimisation**: it is represented by the *Graph-Optimiser* tool that analyses and expresses the given graph processing workload into a workflow of BGO. It further combines parametric BGO performance and energy

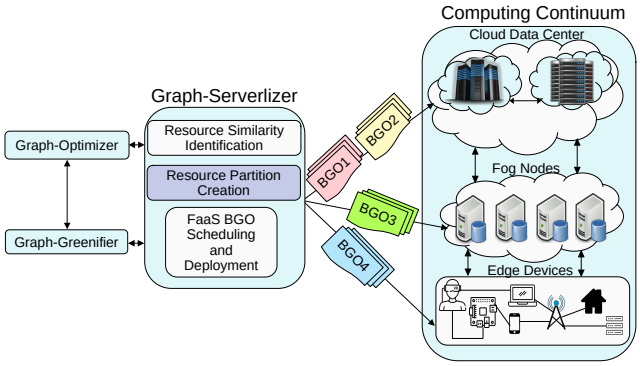

Figure 3: Demonstration of `Graph-Serverlizer` workflow.

models with hardware models to generate accurate performance and energy consumption predictions for the workload running on a given multi-node, heterogeneous infrastructure of CPUs, GPUs, and FPGAs. The predictions indicate the most promising combinations of BGO optimisations and infrastructure, *i.e.*, a codesigned solution for the given workload while guaranteeing its performance and energy consumption bounds.

④ **Sustainability analysis**: it is implemented by the *Graph-Greenifier* tool that collects, studies, and archives performance and sustainability data from operational data centers and national energy suppliers on a large scale. This phase simulates multi-objective infrastructure sustainability profiles for operating graph analytics workloads, trading off performance and energy (*e.g.*, consumption, CO2, methane, GHG emissions) metrics. Its ultimate purpose is to model the impact of specific graph analytics workloads on the environment for evidence-based decision-making.

⑤ **Serverless graph processing:** the `Graph-Serverlizer` tool is responsible to implement it. It uses performance and sustainability models and data from the previous phases to deploy serverless graph analytics on the computing continuum. It relies on novel scheduling heuristics, infrastructure partitioning, and environment-aware processing for scalable orchestration of serverless graph analytics with an accountable performance and energy consumption trade-off.

The *hardware infrastructure* layer considered by Graph-Massivizer consists of geographically distributed data centers across the HPC, cloud, fog, and edge computing continuum. A data center is a collection of nodes connected through high-performance networks. A node represents a heterogeneous set of tightly coupled devices comprising commodity multiprocessors and specialized accelerators such as GPU or FPGA.

4 GRAPH-SERVERLIZER

This section designs essential components of the `Graph-Serverlizer` tool, show its plans for addressing described RQs, and outlines our ongoing and future works.

4.1 Design

The `Graph-Serverlizer` tool encapsulates BGO as serverless functions targeting their scheduling and deployment on the computing continuum according to the performance and sustainability metrics and labels communicated by Graph-Optimizer and Graph-Greenifier (See Fig. 3). The serverless technology allows deploying BGO with minimal operational delay and reduced financial cost based on a pay-as-use business model. Furthermore, it lessens the burden on developers by providing transparent runtime management. As shown in Fig. 3, `Graph-Serverlizer` includes three components, *i.e.*, *(1) Resource Similarity Identification*, *(2) Resource Partition Creation*, *(3) FaaS BGO Scheduling and Deployment* , which provide the following services:

- **Similarity resource partitioner:** `Graph-Serverlizer` employs a three-step approach that partitions the underlying infrastructure nodes by considering the resources, I/O, and network BGO requirements. It first applies resource extraction to identify HPC, Cloud, and Edge codesigned infrastructure nodes' characteristics based on resource types, such as processing cores, memory, storage, and energy consumption. Next, the multilayer infrastructure facilitation clusters the infrastructure nodes comprising specialised hardware (GPUs, FPGAs) based on their topological (betweenness centrality) and similarity relationships between the related resources. Finally, the layer partitioning clusters each resource layer of the multilayer infrastructure in disjoint resource partitions of nodes with similar resource types, including operational delay, sustainability, and energy consumption metric.
- **Sustainable BGO function operation:** `Graph-Serverlizer` targets three BGO function operation phases, considering heterogeneous hardware resources and sustainability profiles. Firstly, feature partitioning identifies appropriate nodes with similar features, cost, and sustainability characteristics to the resource requirements of the BGO encompassed as functions. Afterward, function scheduling allocates appropriate virtual instances within the nodes of the same feature partition and highly connected network layer partition based on the performance and sustainability metrics provided by Graph-Optimizer. `Graph-Serverlizer` envisions a scheduling algorithm inspired by matching theory opposing two conflicting players (*i.e.*, BGO and hardware nodes), bounded by the sustainability metrics provided by Graph-Greenifier.
- **BGO serverlization:** implements function wrapping techniques for BGO and the required execution libraries for a set of serverless computing platforms, such as Apache OpenWisk [4].
- **Cybersecure deployment:** addresses runtime aspects of the FaaS functions and provides elastic and scalable deployment of the BGO while minimizing operational costs. It further features real-time sustainability analysis and automated decision-making with a reduced negative environmental impact. `Graph-Serverlizer` employs state-of-the-art security and privacy mechanisms [14] installed at the MG providers to protect against malicious attacks.

4.2 Ongoing and Future Directions

We continue our research for the current and future works as follows:

- **Serverless resource (in)similarity (RQs 2,4,7):** The computing continuum and, in particular, federated FaaS offer extremely

heterogeneous resources to deploy serverless functions. In general, the maximum amount of memory used by each function is configured by the user during the deployment. However, the CPU resources are managed by each cloud provider differently. While cloud providers offer *fine-grained* memory setups per MB, they provide *coarse-grained* CPU resources, often configured by the providers without user control [6]. Unfortunately, even the underlying CPUs are heterogeneous for all evaluated cloud providers, *e.g.*, AWS, IBM, Google, and Azure, as reported by Kelly *et al.* [22]. Ristov *et al.* [32] introduced function twins, which represent the resource similarity of the same function being deployed across different regions of the same cloud provider with the same amount of memory usage. Indeed, twins that access cloud storage within the identical region as the function provide the same performance. The authors introduced an accurate model to simulate the performance of serverless workflow applications across different locations in federated FaaS. Therefore, one of our directions will be equipping the `Serverlizer` with such a model.

- **Serverless benchmark platforms (RQs 3,4,7):** As Copik *et al.* [12] used three BGOs (*i.e.*, pagerank, mst, and bfs) and presented a serverless benchmark suite for FaaS Computing (named SeBS), we plan to utilize BGOs to benchmark serverless platforms. More precisely, categorizing various BGOs into distinct models and mimicking serverless function execution, including CPU and memory usage simulation, could be employed for this aim [16].
- **BGO serverlization with FaaSification (RQs 1,5,6):** There are several works in the literature to produce serverless functions. For instance, some solutions, *e.g.*, FaaSifier M2FaaS [29], transform certain sections of current monolithic applications into serverless functions. Other methods like [13] involve developing the code of a local technique with the corresponding handler method of the target cloud provider and then deployment as a serverless function on numerous cloud providers. However, the abovementioned approaches are mainly focused on creating a deployment package with a limited setup for deployment, which is usually restricted to a single memory setup, region, and provider. For more advanced deployment, `Serverlizer` can be equipped to enables user to generally utilize Infrastructure-as-a-Code tools [7, 31] to deploy functions across the computing continuum from a single deployment package.

5 CONCLUSION

This paper aimed to explore how the emerging serverless computing paradigm can be leveraged to design scalable serverless graph analytics over a codesigned continuum infrastructure. We raised seven primary research questions to address the existing graph processing platform challenges using serverless computing. We surveyed state-of-the-art graph processing solutions and introduced our serverless tool, *i.e.*, `Graph-Serverlizer`, within the Graph-Massivizer project toolkit. Moreover, three ongoing and future directions were presented to outline our overall vision of our research plan.

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] Cristina Abad, Ian T Foster, Nikolas Herbst, and Alexandru Iosup. 2021. Serverless computing (Dagstuhl seminar 21201). In *Dagstuhl Reports*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.
[2] Hamilton Wilfried Yves Adoni, Tarik Nahhal, Moez Krichen, Brahim Aghezzaf, and Abdeltif Elbyed. 2020. A survey of current challenges in partitioning and processing of graph-structured data in parallel and distributed systems. *Distributed and Parallel Databases* (2020).
[3] Apache. 2020. Apache Giraph. Retrieved 2023-02-12 from https://giraph.apache.org/
[4] Apache. 2020. Apache OpenWhisk. Retrieved 2023-02-12 from https://openwhisk.apache.org/
[5] Mohammad Sadegh Aslanpour, Adel N Toosi, Muhammad Aamir Cheema, and Raj Gaire. 2022. Energy-aware resource scheduling for serverless edge computing. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE.
[6] Setia Blog. 2020. Apache Giraph. Retrieved 2023-02-12 from https://www.sentiatechblog.com/aws-re-invent-2020-day-3-optimizing-lambda-cost-with-multi-threading?utm_source=reddit&utm_medium=social&utm_campaign=day3_lambda
[7] Yevgeniy Brikman. 2019. *Terraform: Up & Running: Writing Infrastructure as Code*. O'Reilly Media.
[8] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache flink: Stream and batch processing in a single engine. *The Bulletin of the Technical Committee on Data Engineering* (2015).
[9] Paul Castro, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. 2019. The rise of serverless computing. *Commun. ACM* (2019).
[10] Janneth Chicaiza and Priscila Valdiviezo-Diaz. 2021. A comprehensive survey of knowledge graph-based recommender systems: Technologies, development, and contributions. *Information* (2021).
[11] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. 2015. One trillion edges: Graph processing at facebook-scale. *Proceedings of the VLDB Endowment* (2015).
[12] Marcin Copik, Grzegorz Kwasniewski, Maciej Besta, Michal Podstawski, and Torsten Hoefler. 2021. SeBS: A Serverless Benchmark Suite for Function-as-a-Service Computing. Association for Computing Machinery.
[13] Cordingly, Robert and Shu, Wen and Lloyd, Wes J. 2020. Predicting performance and cost of serverless computing functions with SAAF. In *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE.
[14] Muhammed Golec, Ridvan Ozturac, Zahra Pooranian, Sukhpal Singh Gill, and Rajkumar Buyya. 2021. IFaaSBus: A security-and privacy-based lightweight framework for serverless computing using IoT and machine learning. *IEEE Transactions on Industrial Informatics* (2021).
[15] Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. 2014. GraphX: Graph Processing in a Distributed Dataflow Framework. In *USENIX OSDI*.
[16] Ryan Hancock, Sreeharsha Udayashankar, Ali José Mashtizadeh, and Samer Al-Kiswany. 2022. OrcBench: A Representative Serverless Benchmark. In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*.
[17] Safiollah Heidari and Rajkumar Buyya. 2019. A cost-efficient auto-scaling algorithm for large-scale graph processing in cloud environments with heterogeneous resources. *IEEE Transactions on Software Engineering* (2019).
[18] Safiollah Heidari, Yogesh Simmhan, Rodrigo N Calheiros, and Rajkumar Buyya. 2018. Scalable graph processing frameworks: A taxonomy and open challenges. *ACM Computing Surveys (CSUR)* (2018).
[19] Alexandru Iosup, Alexandru Uta, Laurens Versluis, Georgios Andreadis, Erwin Van Eyk, Tim Hegeman, Sacheendra Talluri, Vincent Van Beek, and Lucian Toader. 2018. Massivizing computer systems: a vision to understand, design, and engineer computer ecosystems through and beyond modern distributed systems. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE.
[20] Matthijs Jansen, Auday Al-Dulaimy, Alessandro V Papadopoulos, Animesh Trivedi, and Alexandru Iosup. 2022. The SPEC-RG Reference Architecture for the Edge Continuum. *arXiv preprint arXiv:2207.04159* (2022).
[21] Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht. 2017. Occupy the cloud: Distributed computing for the 99%. In *Proceedings of the 2017 symposium on cloud computing*.

[22] Daniel Kelly, Frank Glavin, and Enda Barrett. 2020. Serverless Computing: Behind the Scenes of Major Platforms. In *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*.

[23] Youngbin Kim and Jimmy Lin. 2018. Serverless data analytics with flint. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE.

[24] Eren Kurshan and Hongda Shen. 2020. Graph computing for financial crime and fraud detection: Trends, challenges and outlook. *International Journal of Semantic Computing* (2020).

[25] Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2020. Application management in fog computing environments: A taxonomy, review and future directions. *ACM Computing Surveys (CSUR)* (2020).

[26] Claudio Martella, Roman Shaposhnik, Dionysios Logothetis, and Steve Harenberg. 2015. *Practical graph analytics with apache giraph*. Springer.

[27] Stefan Nastic, Thomas Rausch, Ognjen Scekic, Schahram Dustdar, Marjan Gusev, Bojana Koteska, Magdalena Kostoska, Boro Jakimovski, Sasko Ristov, and Radu Prodan. 2017. A serverless real-time data analytics platform for edge computing. *IEEE Internet Computing* (2017).

[28] Pavlopoulos, Georgios A and Secrier, Maria and Moschopoulos, Charalampos N and Soldatos, Theodoros G and Kossida, Sophia and Aerts, Jan and Schneider, Reinhard and Bagos, Pantelis G. 2011. Using graph theory to analyze biological networks. *BioData mining* (2011).

[29] Stefan Pedratscher, Sasko Ristov, and Thomas Fahringer. 2022. M2FaaS: Transparent and fault tolerant FaaSification of Node. js monolith code blocks. *Future Generation Computer Systems* 135 (2022), 57–71.

[30] Radu Prodan, Dragi Kimovski, Andrea Bartolini, Michael Cochez, Alexandru Iosup, Evgeny Kharlamov, Jože Rožanec, Laurenţiu Vasiliu, and Ana Lucia Vărbănescu. 2022. Towards Extreme and Sustainable Graph Processing for Urgent Societal Challenges in Europe. In *2022 IEEE Cloud Summit*. IEEE.

[31] Sashko Ristov, Simon Brandacher, Michael Felderer, and Ruth Breu. 2022. GoDeploy: Portable Deployment of Serverless Functions in Federated FaaS. In *2022 IEEE Cloud Summit*. https://doi.org/10.1109/CloudSummit54781.2022.00012

[32] Sashko Ristov, Mika Hautz, Christian Hollaus, and Radu Prodan. 2022. SimLess: Simulate Serverless Workflows and Their Twins and Siblings in Federated FaaS. Association for Computing Machinery.

[33] Narayanan Sundaram, Nadathur Rajagopalan Satish, Md Mostofa Ali Patwary, Subramanya R Dulloor, Satya Gautam Vadlamudi, Dipankar Das, and Pradeep Dubey. 2015. Graphmat: High performance graph analytics made productive. *arXiv preprint arXiv:1503.07241* (2015).

[34] Márk Szalay, Péter Mátray, and László Toka. 2021. Real-time task scheduling in a FaaS cloud. In *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*. IEEE.

[35] Lucian Toader, Alexandru Uta, Ahmed Musaafir, and Alexandru Iosup. 2019. Graphless: Toward serverless graph processing. In *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE.

[36] Erwin Van Eyk, Lucian Toader, Sacheendra Talluri, Laurens Versluis, Alexandru Uţă, and Alexandru Iosup. 2018. Serverless is more: From paas to present cloud computing. *IEEE Internet Computing* (2018).