Graph Greenifier: Towards Sustainable and Energy-Aware Massive Graph Processing in the Computing Continuum

Alexandru Iosup and VU Team
A.Iosup@vu.nl
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands

Radu Prodan and KLU Team
radu.prodan@aau.at
AAU Klagenfurt
Klagenfurt, Austria

Ana Lucia Varbanescu
a.l.varbanescu@utwente.nl
U. Twente
Enschede, The Netherlands

ABSTRACT

Our society is increasingly digital, and its processes are increasingly digitalized. As an emerging technology for the digital society, graphs provide a universal abstraction to represent concepts and objects, and the relationships between them. However, processing graphs at a massive scale raises numerous sustainability challenges; becoming energy-aware could help graph-processing infrastructure alleviate its climate impact. Graph Greenifier aims to address this challenge in the conceptual framework offered by the Graph Massivizer architecture. We present an early vision of how Graph Greenifier could provide sustainability analysis and decision-making capabilities for extreme graph-processing workloads. Graph Greenifier leverages an advanced digital twin for data center operations, based on the OpenDC open-source simulator, a novel toolchain for workload-driven simulation of graph processing at scale, and a sustainability predictor. The input to the digital twin combines monitoring of the information and communication technology (ICT) infrastructure used for graph processing with data collected from the power grid. Graph Greenifier thus informs providers and consumers on operational sustainability aspects, requiring mutual information sharing, reducing energy consumption for graph analytics, and increasing the use of electricity from renewable sources.

CCS CONCEPTS

• Computer systems organization → Architectures; **• Information systems → Data management systems**; **• Software and its engineering → Software organization and properties**.

KEYWORDS

Graph Greenifier, Graph Massivizer, graph processing, sustainability, energy-awareness, scalability, digital twin, computing continuum

ACM Reference Format:
Alexandru Iosup and VU Team, Radu Prodan and KLU Team, and Ana Lucia Varbanescu. 2023. Graph Greenifier: Towards Sustainable and Energy-Aware Massive Graph Processing in 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/10.1145/3578245.3585329

ICPE '23 Companion, April 15–19, 2023, Coimbra, Portugal
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0072-9/23/04.
https://doi.org/10.1145/3578245.3585329

1 INTRODUCTION

Increasingly, our society and economy are becoming digital. At the core of this transformation are massive datasets, encoding concepts and processes, and their evolving state. Graphs are a common, universal data abstraction, with wide applicability [31]. For example, in the Graph Massivizer[1], applications of graph processing span green and sustainable finance, global foresight for environment protection, green artificial intelligence for a sustainable automotive industry, and green exascale computing through advanced digital twinning. Graphs related to these and other applications lead to challenges commonly associated with big data, but exacerbated due to the irregular nature of graph processing. Thus, processing graphs at scale raises important sustainability and particularly energy-related challenges. With Graph Greenifier, we envision how to address such challenges through a combined approach that is simulation-based and data-driven.

Graph processing consists of very diverse approaches, covering a variety of algorithms and datasets. Correspondingly, a variety of graph-processing systems already exist [12, 19], for example, the open-source, Hadoop-based Apache Giraph [2, 26] and the Spark-based GraphX [16]. However, none supports sustainable, serverless graph processing across the computing continuum, as Graph Massivizer aims to investigate. Notably, although a variety of graph benchmarks exist [5], e.g., LDBC Graphalytics [20, 21] that renews periodically to stay current, no benchmark that considers *basic graph operations* (*BGO*) such as graph enrichment, query, and analytics, or *sustainable* and/or *serverless graph operations*, currently exists.

The operational challenges related to massive graphs, including volume, velocity, and veracity, may seem similar to those experienced in big data processing. However, the irregular nature of graph applications, where between any pair of nodes there could be few degrees of separation, also poses the challenge of *vicissitude*, where any of these challenges may arise at any time. Although vicissitude is not a new concept [15], for graph processing it has not been explored. One of the important consequences of vicissitude is that understanding and predicting energy use, and further analyzing the sustainability of graph processing, remains an important open challenge.

Addressing the open challenge, we make the following contributions:

C1 Together with partners in the Graph Massivizer project, we propose an operational model for serverless, massive graph processing in the computing continuum (§2). The overall architecture includes five main components, *Graph-Inceptor* for on-demand,

[1]EU H2020 project Graph-Massivizer, https://graph-massivizer.eu/

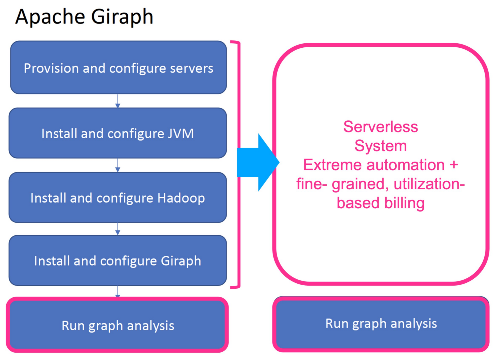
Apache Giraph
Provision and configure servers
Install and configure JVM
Install and configure Hadoop
Install and configure Giraph
Run graph analysis
Serverless System
Extreme automation + fine- grained, utilization-based billing
Run graph analysis

Figure 1: Comparison between operations for graph processing: (left) self-managed and (right) serverless.

interactive graph definition; *Graph-Scrutinizer* for essential BGO; *Graph-Optimizer* for graph workload modeling and optimization; *Graph Greenifier* for sustainability analysis and energy awareness capabilities (this work); and *Graph-Serverless* for serverless BGO processing.

C2 We design Graph Greenifier, the sustainability analysis and energy-awareness component of Graph Massivizer (§3). Graph Greenifier aims to help collect, study, use, and archive performance and sustainability data from operational data centers and national energy suppliers on a large scale, through a combination of five main components: a work-driven simulation toolchain, a sustainability predictor, various monitoring capabilities, a sustainability benchmark, and a power-grid data interface.

C3 We consider an approach to realize the design (§4). Toward this end, we consider five research questions: How to provide continuous, fast-grained monitoring data across various services? How to operate in the computing continuum? How to enable digital twinning of massive graph-processing operations? How to combine ICT and energy infrastructure metrics for benchmarking and labeling purposes? and How to combine BGO-oriented and operations-related predictions?

2 OPERATIONAL MODEL: SERVERLESS, MASSIVE GRAPH PROCESSING

For contribution **C1**, we analyze the main components of massive graph processing in the computing continuum.

2.1 Serverless Operations for Graph Processing

Figure 1 compares self-managed and serverless operations for graph processing. In the scenario for self-managed operations, the graph-processing stack is derived from a real-world Apache Giraph deployment. Operations include provisioning and configuring servers, installing and configuring the virtual machines (VMs) and containers, installing and configuring Hadoop, installing and configuring Giraph (the graph-processing engine), before running the graph analysis.

In the serverless scenario, based on the Graphless toolkit for serverless graph processing [37], the stack is managed automatically, so the application developer and the user are released from operational concerns. Further concerns, beyond detailed operational decisions of auto-scaling, etc., are fine-grained reporting of resources actually used and utilization-based billing [1].

As Figure 1 depicts, the two scenarios differ significantly in the effort the developer (or user) have to put into operational concerns, instead of focusing on the domain-specific logic of the applications.

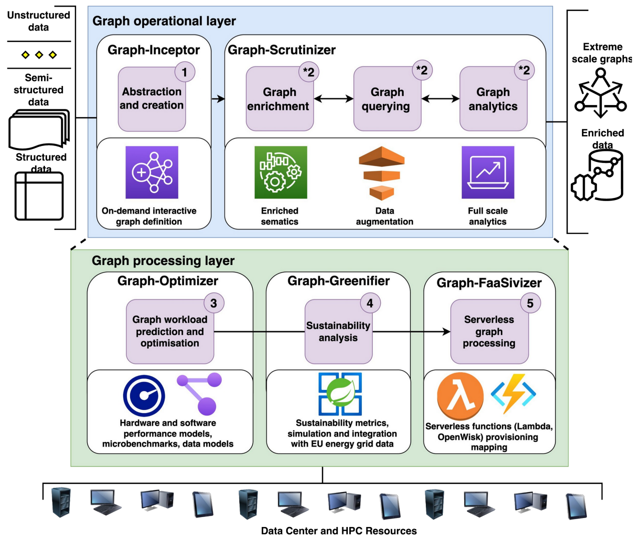

Figure 2: The Graph-Massivizer architecture.

2.2 Graph Massivizer Framework for Massive Graph Processing

The Graph Massivizer architecture consists of the five major components depicted in Figure 2; we refer to the project proposal for a full description. By combining these five components, Graph Massivizer ingests data in structured, semi-structured, and unstructured form, and, step by step, produces from it extreme scale graphs, enriched data, and actionable insight by processing the graphs and enriched data.

The *Graph operational layer* facilitates generating, transforming, and manipulating extreme data through *basic graph operations* (BGO), comprising graph creation, enrichment, query, and analytics. To achieve this, this layer combines two major components:

❶ *Graph-Inceptor*, which achieves on-demand, interactive graph definition by translating data from various static and event streams, or by following heuristics to generate synthetic data, persist it, and publish it within a graph structure. Because the input data can be large-scale, Graph-Inceptor can produce extreme-scale graphs.

❷ *Graph-Scrutinizer* realizes three essential BGOs: graph enrichment, query, and analytics. With these, users can analyse and expand extreme datasets using probabilistic reasoning and ML algorithms for graph pattern discovery, while the system achieves graph generation with a low-memory footprint, and error-bounded responses to queries with low latency. The output is a new graph, a query, or an enriched structured dataset, possibly of a larger scale than the input.

The *Graph processing layer* provides sustainable and energy aware serverless graph analytics on the underlying heterogeneou HPC infrastructure, following three phases:

❸ *Graph-Optimizer* focuses on graph workload modeling and op timization. This component further analyses and expresses a given graph-processing workload into a workflow of BGO. I further combines parametric BGO performance and energy mod els 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 optimizations and infrastructure, i.e., a codesigned solution for the given workload while guaranteeing its performance and energy consumption bounds.

❹ *Graph Greenifier* offers capabilities for sustainability analysis and energy awareness, focusing on sustainability metrics, simu lation and digital twinning, and integration with the EU energy grid data (see §3).

❺ *Graph-Serverless* enables serverless BGO processing. To this end, this component uses performance and sustainability models and data provided by the other components. A main feature of this component is the ability to deploy (serverless) graph analytics on the computing continuum, using 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.

3 GRAPH GREENIFIER: SUSTAINABLE AND ENERGY-AWARE MASSIVE GRAPH PROCESSING

For contribution **C2**, we envision the high-level design of Graph Greenifier. Graph Greenifier aims to facilitate collecting, studying, and archiving performance and sustainability data from operational data centers and national energy suppliers on a large scale. To this end, Graph Greenifier 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 typical use case is to model the impact of specific graph analytics workloads running on arbitrary infrastructure on the environment, for evidence-based decision making.

Graph Greenifier has five design components:

Work-driven simulation toolchain (Figure 3, ❶): Graph Greenifier utilizes and extends the OpenDC [27] simulator, encompassing public information from national energy suppliers to model the impact of graph processing on the climate and, therefore, society. OpenDC is free open-source software. The simulator uses the selected sustainability indicators, such as carbon footprint, CO2, and methane emissions as calculated by the sustainability predictor, to estimate the impact of different scenarios.

Sustainability predictor (❷): Graph Greenifier extends and upscales Graph-Optimizer's predictions to rank graph processing scenarios based on performance, energy efficiency and sustainability at scale. Data center operators use this ranking to choose the most sustainable operational procedures at runtime, with the help of Graph-Serverlizer (the FaaS BGO scheduling and deployment component, Ⓑ), which steers the workload to more appropriate

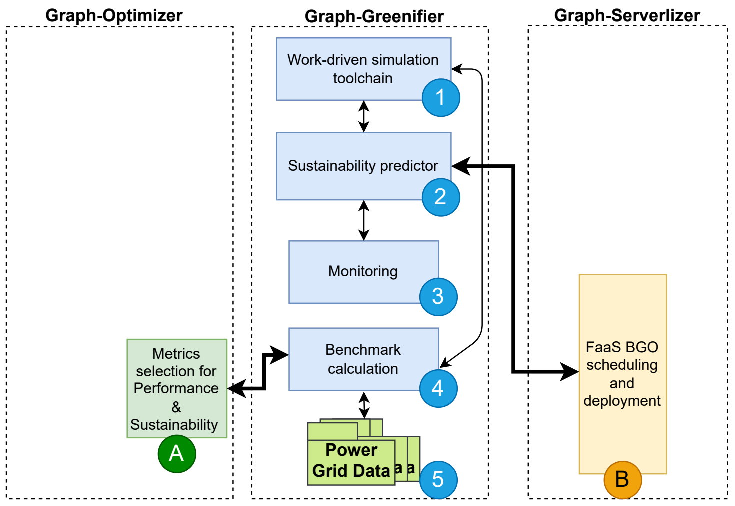

Figure 3: The Graph Greenifier architecture.

infrastructure resources. The process is transparent and evidence-based.

Monitoring (❸): Graph Massivizer identifies relevant sustainability metrics based on extensive state-of-the-art analysis and stakeholder interviews and establishes effective means to monitor and calculate them depending on installed measurement capabilities. For example, it uses software energy counters where hardware counters and power meters are unavailable, and deploys the infrastructure to collect and archive these metric calculations.

Sustainability benchmark (❹): Graph Greenifier proposes a sustainability benchmark providing runtime energy labels, including information about energy sources derived from data centers and energy operation models. Graph Greenifier creates a closed monitoring loop, encompassing simulation and estimations of the emitted GHG pollutants for performing the BGO processing. The GHG estimates engage the data center operators and other stakeholders in meaningful dialogues for reaching informed graph processing decisions with reduced impact on the environment. The Graph-Optimizer component provides detailed metrics for this (the metrics component, Ⓐ).

Power grid data interface (❺): Graph Greenifier automates data gathering based on the offer and price of electrical energy on the open market and its energy source and greenness. It uses monitoring data from the local data center operators such as the SURF operational dataset [38], availability insights from cloud operators, and third-party aggregators of user reports such as Outage Report [36], and energy- and sustainability-related data published by national infrastructures such as the EU-wide ENTSO-E[2] transparency platform or the Dutch national data source[3] coupled with credible energy-consumption analyses such as those provided by national statistics bureaus.

[2]https://www.entsoe.eu/
[3]https://energieopwek.nl/

4 TOWARDS REALIZING GRAPH GREENIFIER

For contribution **C3**, we envision a set of challenges, formulated as research questions, and steps to addressing them aiming for design, implementation, and, finally, realization in practice.

4.1 How to provide continuous, fast-grained monitoring data across various services?

Serverless computing promises unprecedented fine-granularity in reporting (and charging) use of resources and services [1]. Providing serverless operations combines the capabilities of multiple operational layers, such as resource, (single) function, and workflow management [14], and various state-related operations [1]. Even if the resource management layer is uniform, e.g., around the de facto standard offered by Kubernetes interfaces, collecting and reporting data across multiple (distributed) services with synchronous and particularly asynchronous access, remains non-trivial [33].

Through its monitoring component, Graph Greenifier aims to provide a trade-off between the accuracy and precision of monitoring information, and the performance and scalability overheads incurred by it, both short- and long-term. Often, the lack of accuracy inherent to all but the most prohibitive profiling processes, which affects for example all tracing efforts, prevents extracting accurate performance breakdowns (e.g., latency cannot be ascribed other than end-to-end) [33].

A significant implementation and realization challenge also appears, where monitoring must be concerned with the diversity of services and devices involved in the continuum (see §4.2).

4.2 How to operate in the computing continuum?

A foundational capability for Graph Massivizer is to use resources across the computing continuum [23]. Cloud [25], edge [32, 34], IoT and fog [3, 34], and other computing models offer various services and resources with different characteristics, e.g., performance, availability, energy reserves, and sometimes also latency to the user, as Figure 4 depicts. These models come each with specific software, and sometimes even software-stacks. Currently, there is no approach to seamlessly run stateful applications across all these stacks.

Graph Greenifier aims to enable Graph-Serverlizer to take operational decisions that not only allow selecting services and resources, but doing so with energy-awareness, toward more sustainable operations. To this end, Graph Greenifier provides predictions of how arbitrary decisions would impact operations and, alongside, energy- and sustainability-related metrics.

To enact such decisions, Graph Massivizer has dedicated engineering effort that is enabling the use of services and resources across these stacks. The Continuum framework [24] can automatically deploy and run across (emulated) infrastructures and networks, locally and in the cloud, using Kubernetes interfaces. This allows, for example, combining cloud resources operated with Kubernetes and Knative, with edge resources operated with KubeEdge; and higher-level services offered under an OpenFaaS interface. Other tools, like Fogify [35] and MockFog [18], offer different emulation capabilities.

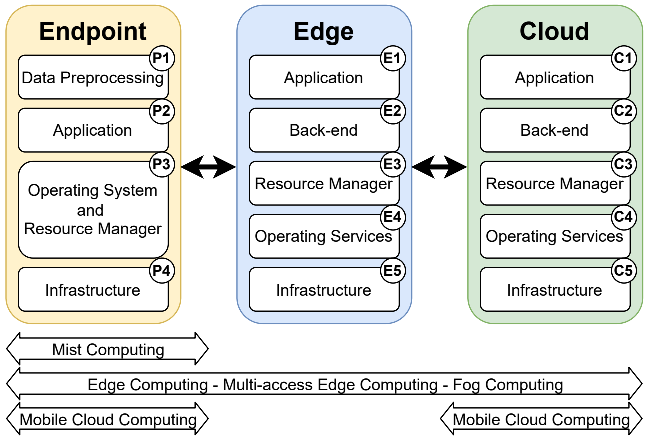

Figure 4: The SPEC-RG reference architecture for the compute continuum [23].

4.3 How to enable digital twinning of massive graph-processing operations?

Every field operating complex infrastructure or processes eventually resorts to a dense structure of simulation, emulation, and prototype-based experimentation. For parts of the computing continuum, many simulation tools already exist [4, 7], e.g, from the very recent, OpenDC 2.0 [27], WRENCH [9], and iFogSim [17], to the early approaches, such as GridSim and CloudSim [8], SimGrid [10], DGSim [22], and GroudSim [28]. However, no integrated capabilities exist, only OpenDC offers support for both short- and long-term decisions, and the energy and sustainability modeling capabilities of these simulators remain limited.

Graph Greenifier aims to address these gaps, and in particular: combine the capabilities of OpenDC with support for extensive energy and sustainability modeling, capabilities related to serverless operation across the computing continuum (see §4.2), and calibration with detailed monitoring information (see §4.1).

A non-trivial addition to the simulation process is the concept of *provenance*, which generally is "machine-readable summary of the collection and computational history of a datase" [13]. Recording provenance can be done at many levels in the system, and end-to-end provenance can include: (i) where provenance, tracking where (each item of) output comes from and how it links to specific (items of) the input, (ii) how provenance, tracking how the output was produced from the input, (iii) lineage-based provenance, linking to output multiple sets of input items, etc. Such metadata could be used to trace how graph operations of different granularities, short-term and particularly long-term, contribute to energy use and pollution.

4.4 How to combine ICT and energy infrastructure metrics for benchmarking and labeling purposes?

Two vast and complex classes of infrastructure contribute to graph processing operations, information and communication technology (ICT) and energy infrastructures. Each can provide diverse monitoring capabilities, with existing and emerging metrics (for ICT, see §4.1), but no uniform approach to collect and combine such data.

Graph Greenifier aims to combine the information provided by powerful yet flexible monitoring systems such as Examon [6], with energy-related data from sources such as the EU-sponsored ENTSO-E. Combining such information, combined with the advanced simulation capabilities (see §4.3), could deliver both benchmarking capabilities, i.e., through a small set of representative metrics that can be reported across every graph-processing infrastructure, and labeling features, i.e., through a high-level label that applies to each "graph-processing appliance".

It is important that monitoring data can be evaluated and tested for fitness to its purpose. This could be achieved by combining the more expressive capabilities of Graph Greenifier, e.g., detailed simulation-based analysis of how much energy is consumed for specific workloads and resource topologies, with the reported data in both ICT and energy infrastructures. A back-and-forth testing process could help identify possible inconsistencies and start an improvement cycle.

4.5 How to combine BGO-oriented and operations-related predictions?

The Graph-Optimizer component of Graph Massivizer focuses on BGO-level predictions. These involve detailed decisions in modeling, particularly those related to the heterogeneous hardware. However, the operational infrastructure further includes a diverse software ecosystem. Predicting how the operational techniques at each layer will behave when put together into the same ecosystem, subject to the dynamics provided by input data and more generally by the input workload, remains an open challenge.

Graph Greenifier aims to address this challenge to make useful predictions about sustainability, where predictions can be further explained with operational detail. For example, caching input data that gets reused multiple times is much used in industrial serverless systems such as Snowflake [39], Databricks [11] and others [29, 30, 40], but explaining its use and further linking these caches to energy use and climate impact, has not yet been tried in the context of serverless computing, graph processing, or computing continuum.

5 CONCLUSION

In this work, we focus on the emerging societal need to process graphs at a massive scale. We posit energy awareness, and more generally sustainability, are essential goals on the roadmap for graph-processing technology.

Starting from the Graph Massivizer framework, we propose a high-level design for Graph Greenifier, the sustainability analysis component of the project, and consider important aspects to consider, to realize this design. We focus on computing continuum operation, monitoring with fine granularity, digital twinning, combining ICT- and energy-related information for benchmarking and labeling purposes, and combining BGO-oriented and operations-related predictions.

In future work, we aim to explore systematically the space for detailed design enabled by the high-level design introduced in this work. Through experimental approaches, we aim to validate that high-level and detailed designs can be realized in practice. To encourage community adoption, we aim to create a public archive with operational results and conduct use-case studies with four classes of applications.

AUTHORS

The VU Team further includes Sacheendra Talluri, Gilles Magalhaes, Kailhan Hokstam, Hugo Zwaan, and Vincent van Beek.

The KLU Team further includes Reza Farahani, Dragi Kimovski, and Sashko Ristov.

ACKNOWLEDGMENTS

This project is co-funded by the projects NWO Top2 OffSense, EU H2020 GraphMassivizer, and EU MCSA-RISE CLOUDSTARS.

REFERENCES

[1] Cristina L. Abad, Ian T. Foster, Nikolas Herbst, and Alexandru Iosup. 2021. Serverless Computing (Dagstuhl Seminar 21201). *Dagstuhl Reports* 11, 4 (2021), 34–93.

[2] Apache. 2020. Apache Giraph. Retrieved 2023-02-12 from https://giraph.apache.org/

[3] AWS. 2021. AWS Greengrass. https://aws.amazon.com/greengrass/. Accessed: 2021-05-12.

[4] Ilyas Bambrik. 2020. A Survey on Cloud Computing Simulation and Modeling. *SN Computer Science* 1, 5 (2020), 249.

[5] Angela Bonifati, George H. L. Fletcher, Jan Hidders, and Alexandru Iosup. 2018. A Survey of Benchmarks for Graph-Processing Systems. In *Graph Data Management, Fundamental Issues and Recent Developments*, George H. L. Fletcher, Jan Hidders, and Josep Lluís Larriba-Pey (Eds.). Springer, 163–186.

[6] Andrea Borghesi, Alessio Burrello, and Andrea Bartolini. 2023. ExaMon-X: A Predictive Maintenance Framework for Automatic Monitoring in Industrial IoT Systems. *IEEE Internet Things J.* 10, 4 (2023), 2995–3005. https://doi.org/10.1109/JIOT.2021.3125885

[7] James Byrne, Sergej Svorobej, Konstantinos M. Giannoutakis, Dimitrios Tzovaras, Peter J. Byrne, Per-Olov Östberg, Anna Gourinovitch, and Theo Lynn. 2017. A Review of Cloud Computing Simulation Platforms and Related Environments. In *CLOSER 2017 - Proceedings of the 7th International Conference on Cloud Computing and Services Science, Porto, Portugal, April 24-26, 2017*, Donald Ferguson, Víctor Méndez Muñoz, Jorge S. Cardoso, Markus Helfert, and Claus Pahl (Eds.). SciTePress, 651–663.

[8] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 41, 1 (2011), 23–50.

[9] Henri Casanova, Rafael Ferreira da Silva, Ryan Tanaka, Suraj Pandey, Gautam Jethwani, William Koch, Spencer Albrecht, James Oeth, and Frédéric Suter. 2020. Developing accurate and scalable simulators of production workflow management systems with WRENCH. *Future Generation Computer Systems* 112 (2020), 162–175.

[10] Henri Casanova, Arnaud Giersch, Arnaud Legrand, Martin Quinson, and Frédéric Suter. 2014. Versatile, scalable, and accurate simulation of distributed applications and platforms. *J. Parallel and Distrib. Comput.* 74, 10 (2014), 2899–2917.

[11] Databricks. 2022. Optimize performance with caching on Databricks. https://docs.databricks.com/optimizations/disk-cache.html. Accessed: 10-10-2022.

[12] Niels Doekemeijer and Ana Lucia Varbanescu. 2014. A survey of parallel graph processing frameworks. Technical report from Delft University of Technology.

[13] Aaron M. Ellison, Emery R. Boose, Barbara Staudt Lerner, Elizabeth Fong, and Margo I. Seltzer. 2020. The End-to-End Provenance Project. *Patterns* 1, 2 (2020), 100016.

[14] Erwin Van Eyk, Alexandru Iosup, Johannes Grohmann, Simon Eismann, André Bauer, Laurens Versluis, Lucian Toader, Norbert Schmitt, Nikolas Herbst, and Cristina L. Abad. 2019. The SPEC-RG Reference Architecture for FaaS: From

Microservices and Containers to Serverless Platforms. *IEEE Internet Comput.* 23, 6 (2019), 7–18.

[15] Bogdan Ghit, Mihai Capota, Tim Hegeman, Jan Hidders, Dick H. J. Epema, and Alexandru Iosup. 2014. V for Vicissitude: The Challenge of Scaling Complex Big Data Workflows. In *IEEE/ACM CCGRID*. 927–932.

[16] 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*. 599–613.

[17] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. 2017. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience* 47, 9 (2017), 1275–1296.

[18] Jonathan Hasenburg, Martin Grambow, and David Bermbach. 2020. MockFog 2.0: Automated Execution of Fog Application Experiments in the Cloud. *CoRR* abs/2009.10579 (2020). arXiv:2009.10579

[19] Safiollah Heidari, Yogesh Simmhan, Rodrigo N. Calheiros, and Rajkumar Buyya. 2018. Scalable Graph Processing Frameworks: A Taxonomy and Open Challenges. *ACM Comput. Surv.* 51, 3 (2018), 60:1–60:53.

[20] Alexandru Iosup, Tim Hegeman, Wing Lung Ngai, Stijn Heldens, Arnau Prat-Pérez, Thomas Manhardt, Hassan Chafi, Mihai Capota, Narayanan Sundaram, Michael J. Anderson, Ilie Gabriel Tanase, Yinglong Xia, Lifeng Nai, and Peter A. Boncz. 2016. LDBC Graphalytics: A Benchmark for Large-Scale Graph Analysis on Parallel and Distributed Platforms. *Proc. VLDB Endow.* 9, 13 (2016), 1317–1328.

[21] Alexandru Iosup, Ahmed Musaafir, Alexandru Uta, Arnau Prat-Pérez, Gábor Szárnyas, Hassan Chafi, Ilie Gabriel Tanase, Lifeng Nai, Michael J. Anderson, Mihai Capota, Narayanan Sundaram, Peter A. Boncz, Siegfried Depner, Stijn Heldens, Thomas Manhardt, Tim Hegeman, Wing Lung Ngai, and Yinglong Xia. 2020. The LDBC Graphalytics Benchmark. *CoRR* abs/2011.15028 (2020). arXiv:2011.15028

[22] Alexandru Iosup, Omer Ozan Sonmez, and Dick H. J. Epema. 2008. DGSim: Comparing Grid Resource Management Architectures through Trace-Based Simulation. In *Euro-Par 2008 - Parallel Processing, 14th International Euro-Par Conference, Las Palmas de Gran Canaria, Spain, August 26-29, 2008, Proceedings (Lecture Notes in Computer Science, Vol. 5168)*, Emilio Luque, Tomàs Margalef, and Domingo Benitez (Eds.). Springer, 13–25.

[23] Matthijs Jansen, Auday Al-Dulaimy, Alessandro V. Papadopoulos, Animesh Trivedi, and Alexandru Iosup. 2022. The SPEC-RG Reference Architecture for the Edge Continuum. In *IEEE/ACM CCGRID*. arXiv, https://arxiv.org/abs/2207.04159.

[24] Matthijs Jansen, Linus Wagner, Animesh Trivedi, and Alexandru Iosup. 2023. Continuum: Automate Infrastructure Deployment and Benchmarking in the Compute Continuum. In *FastContinuum at ICPE*.

[25] Liu et al. 2011. NIST cloud computing reference architecture. *NIST special publication* 500, 2011 (2011).

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

[27] Fabian Mastenbroek, Georgios Andreadis, Soufiane Jounaid, Wenchen Lai, Jacob Burley, Jaro Bosch, Erwin Van Eyk, Laurens Versluis, Vincent van Beek, and Alexandru Iosup. 2021. OpenDC 2.0: Convenient Modeling and Simulation of Emerging Technologies in Cloud Datacenters. In *IEEE/ACM CCGRID*. 455–464.

[28] Simon Ostermann, Kassian Plankensteiner, Radu Prodan, and Thomas Fahringer. 2010. GroudSim: An Event-Based Simulation Framework for Computational Grids and Clouds. In *Euro-Par WS (Lecture Notes in Computer Science, Vol. 6586)*. Springer, 305–313.

[29] Qubole. 2023. Qubole Rubix. https://github.com/qubole/rubix. Accessed: 01-02-2023.

[30] Quickwit.io. 2023. Architecture of Quickwit Full-text Search. https://quickwit.io/docs/concepts/architecture/. Accessed: 01-02-2023.

[31] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid G. Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. 2021. The future is big graphs: a community view on graph processing systems. *Commun. ACM* 64, 9 (2021), 62–71.

[32] Mahadev Satyanarayanan. 2017. The Emergence of Edge Computing. *Computer* 50, 1 (2017), 30–39.

[33] Joel Scheuner, Simon Eismann, Sacheendra Talluri, Erwin Van Eyk, Cristina L. Abad, Philipp Leitner, and Alexandru Iosup. 2022. Let's Trace It: Fine-Grained Serverless Benchmarking using Synchronous and Asynchronous Orchestrated Applications. *CoRR* abs/2205.07696 (2022). arXiv:2205.07696

[34] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* 3, 5 (2016), 637–646.

[35] Moysis Symeonides, Zacharias Georgiou, Demetris Trihinas, George Pallis, and Marios D. Dikaiakos. 2020. Fogify: A Fog Computing Emulation Framework. In *5th IEEE/ACM Symposium on Edge Computing, SEC 2020, San Jose, CA, USA, November 12-14, 2020*. IEEE, 42–54.

[36] Sacheendra Talluri, Leon Overweel, Laurens Versluis, Animesh Trivedi, and Alexandru Iosup. 2021. Empirical Characterization of User Reports about Cloud Failures. In *IEEE ACSOS*. 158–163.

[37] Lucian Toader, Alexandru Uta, Ahmed Musaafir, and Alexandru Iosup. 2019. Graphless: Toward Serverless Graph Processing. In *ISPDC*. 66–73.

[38] Alexandru Uta, Kristian Laursen, Alexandru Iosup, Paul Melis, Damian Podareanu, and Valeriu Codreanu. 2020. Beneath the SURFace: An MRI-like View into the Life of a 21st-Century Datacenter. *login Usenix Mag.* 45, 3 (2020).

[39] Midhul Vuppalapati, Justin Miron, Rachit Agarwal, Dan Truong, Ashish Motivala, and Thierry Cruanes. 2020. Building An Elastic Query Engine on Disaggregated Storage. In *USENIX NSDI*, Ranjita Bhagwan and George Porter (Eds.). USENIX Association, 449–462.

[40] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *SIGMOD*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 2614–2627.