Future Generation Computer Systems 164 (2025) 107573

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Serverless Computing for Next-generation Application Development

ARTICLE INFO

Keywords:
Serverless
Function-as-a-Service
FaaS
Edge Computing
Resource Management
Scheduling

ABSTRACT

Serverless computing is a cloud computing model that abstracts server management, allowing developers to focus solely on writing code without concerns about the underlying infrastructure. This paradigm shift is transforming application development by reducing time to market, lowering costs, and enhancing scalability. In serverless computing, functions are event-driven and automatically scale in response to events such as data changes or user requests. Despite its advantages, serverless computing presents several research challenges, including managing state for ephemeral functions, mitigating cold start delays, optimizing function composition, debugging, efficient auto-scaling, resource management, and ensuring security and compliance. This special issue focused on addressing these challenges by promoting research on innovative solutions and exploring the potential of serverless computing in new application domains.

1. Introduction

Serverless computing is an emerging paradigm for developing, deploying, and operating cloud applications. It represents a progression towards greater abstraction in virtualization technology, evolving from bare metal to virtual machines, containers, and Functions-as-a-Service (FaaS) and beyond. Serverless computing simplifies the user experience by outsourcing infrastructure and operational tasks to the service provider, offering high agility and reduced costs in application development without requiring (much) operational expertise from users. Functions in serverless computing are typically event-driven, triggered by various events such as changes in a data source, user requests, messages added to a queue, or records inserted into a database. With serverless, users only pay for the resources they consume, eliminating the need for provisioning and paying for resources in advance.

Despite its commercial success and benefits, serverless computing presents several open research challenges, including managing state for ephemeral functions, addressing cold start issues, optimizing function composition, debugging and monitoring, efficient auto-scaling and resource management, and ensuring security and compliance. While serverless has been successfully applied to various domains such as web and mobile backends, it is also expected to play a significant role in scaling High- Performance Computing (HPC) and High-Performance Data Analytics (HPDA), developing smart applications relying on IoT technologies, facilitating the adoption of AI/ML/DL techniques, and enabling Big Data processing, among others. The development of these applications introduces new challenges specific to these domains, necessitating further advancements in serverless technology and dedicated research attention.

This special issue aims to promote high-quality research on recent serverless computing advances to support existing and emerging applications and to inspire related research efforts. In the remaining part of this editorial article, we provide the details of editorial process, the accepted papers, and future research challenges.

2. Editorial Process

All manuscripts submitted to this special issue (SI) were subjected to a thorough single-blind review process, ensuring a high standard of scholarly assessment. Each submission was carefully reviewed and evaluated by a committee of at least two internationally recognized experts in the field, who possess deep expertise and a comprehensive understanding of the subject matter. The review process was guided by several key criteria, including the manuscript's alignment with the topics covered in this SI, the originality and novelty of the research presented, and the scientific rigor and methodological soundness demonstrated in the study. Additionally, reviewers assessed the relevance and practicality of any proposed use cases, the potential impact of the findings on the broader scientific community, and the overall readability and clarity of the manuscript. These stringent criteria ensured that only the highest-quality contributions were selected for publication, reflecting the latest advancements and innovative approaches in the field.

The collective involvement of authors and reviewers from around the world has been instrumental in shaping the success of our SI. Fig. 1 illustrates the representation of contributions to our SI, showing both 67 reviewers and 39 submitted papers from various countries. The figure reflects the diverse international engagement and underscores the collective efforts that have contributed to the success of this SI. The list of reviewers can be found in the Acknowledgments section, and an overview of submissions and accepted articles is provided in the following section.

https://doi.org/10.1016/j.future.2024.107573

Available online 21 October 2024

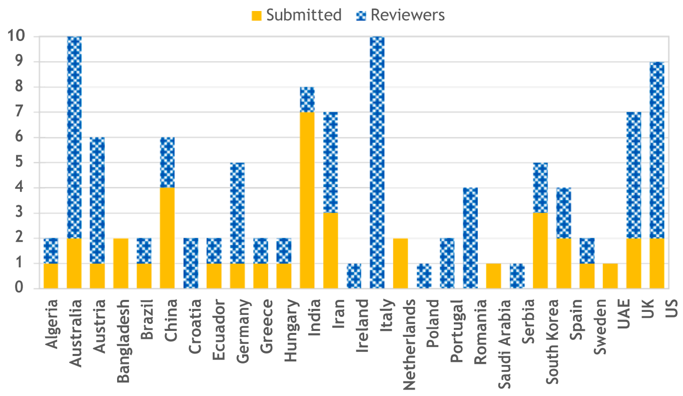

Fig. 1. Geographic distribution of 67 reviewers and 39 submitted papers for the Special Issue (SI).

3. Overview of the Articles

A total of 39 papers were submitted to this SI. Of these, 6 were desk-rejected by the editors, and 16 were rejected following the review process. Ultimately, 17 articles were accepted, resulting in an **acceptance rate of 43.6%**. The accepted articles explore a wide range of applications, including Big Data processing, workflow automation, machine learning, AI, and IoT, across diverse domains such as cloud and edge computing environments. They propose various frameworks and methodologies to enhance performance and efficiency, spanning different architectures from microservices-based applications to cutting-edge technologies like Quantum Computing. Overall we categorized articles into five subtopics:

1. Performance Optimization and Cold Start Management
2. Serverless Architecture and Infrastructure
3. Edge Computing and IoT
4. Machine Learning and AI Integration
5. Workflow and Application Composition

In the following subsections, we provide a brief overview of the accepted articles featured in the SI.

3.1. Performance Optimization and Cold Start Management

Fireman et al [4] in “**Prebaking Runtime Environments to Improve the FaaS Cold Start Latency**” address the challenge of cold start latency in Function-as-a-Service (FaaS) platforms, where delays occur due to the time needed to provision resources and start runtime environments. The authors propose a technique called Prebaking, which uses the CRIU (Checkpoint/Restore In Userspace) tool to create snapshots of warmed function instances. These snapshots can later be restored to significantly reduce the startup time for functions. Their experiments, conducted on OpenFaaS with runtimes like Node.js, CPython, and JVM, show that Prebaking improves function start-up times by up to 25 times, even for complex functions. The study also compares Prebaking with the SEUSS technique and concludes that Prebaking offers better performance, making it a practical solution for reducing cold start latency in serverless platforms.

Similarly in “**Holistic Cold-start Management in Serverless Computing Cloud with Deep Learning for Time Series**”, Nguyen [9] addresses the cold-start problem in serverless computing particularly in time-sensitive applications. Current cold-start management solutions work in isolated silos, lacking communication across system levels. The author propose a 2-prong cold-start management policy using a Temporal Convolutional Network (TCN) to predict function arrivals, enabling continuous feedback between management policies. Evaluation results show reliable performance, offering a more holistic and efficient approach to cold-start management that supports AI-driven, self-adaptive computing systems.

Tran and Kim [14] in “**Optimized Resource Usage with Hybrid Auto-scaling System for Knative Serverless Edge Computing**” address limitations in Knative’s serverless platform auto-scaling, where current horizontal scaling approaches do not account for the need to dynamically adjust resources per instance. Existing hybrid auto-scaling solutions perform poorly with multiple concurrent services and require significant modifications to Knative. Instead, the authors developed Kubernetes operators and custom resources to assist Knative’s auto-scaler with optimal hybrid auto-scaling configurations based on traffic predictions. Their solution adjusts resource levels, target concurrency, and instance numbers dynamically, improving resource efficiency by up to 20% compared to default Knative auto-scaling methods, without modifying Knative’s components.

Finally, in “**Embedding Automated Function Performance Benchmarking, Profiling and Resource Usage Categorization in Function as a Service DevOps Pipelines**”, Katevas et al. [6] develop a comprehensive pipeline that captures both function-level metrics (e.g., wait time, execution time) and resource usage (e.g., CPU, memory). It clusters functions based on their resource consumption patterns, enabling more precise performance analysis. By applying this pipeline to four functions, the authors demonstrate that a two-stage load generation process significantly improves profiling accuracy. Moreover, the method reduces concurrency overheads in function co-placement, enhancing performance and increasing confidence in the FaaS cost model. Future work will expand the profiling dataset and explore how function inputs impact resource usage classification.

3.2. Serverless Architecture and Infrastructure

Werner and Tai [16] in “**A Reference Architecture for Serverless Big Data Processing**” address the challenges of processing large-scale data with FaaS platforms, which are not originally designed for big data tasks. To overcome limitations and improve performance, the authors propose CREW, a serverless data processing framework that leverages application-platform co-design. CREW reduces entry barriers, costs, and outperforms traditional big data frameworks, making serverless data processing a viable alternative for complex tasks, while anticipating the development of specialized serverless platforms for diverse applications.

In another work using FaaS for Big Data processing, titled “**Serverless-like platform for container-based YARN clusters**”, O’scar Castellanos-Rodr’ıguez et al. [17] introduce a serverless platform based on Hadoop YARN for running Big Data workloads with dynamic, real-time resource scaling. Unlike traditional FaaS models, which are limited to handling simple, stateless functions, this platform supports more complex applications through the deployment of serverless YARN clusters. Experimental results demonstrate significant improvements in performance and resource efficiency, with up to a 41% reduction in runtime and a 50% improvement in CPU usage compared to standard YARN. Future developments aim to extend the platform to other container engines and Big Data frameworks.

A novel article “**QFaaS: A Serverless Function-as-a-Service Framework for Quantum Computing**” by Nguyen et al [10] introduces a Quantum Function-as-a-Service framework, called QFaaS, designed to simplify quantum software development in the Noisy Intermediate-Scale Quantum (NISQ) era. By leveraging serverless computing, DevOps practices, and hybrid quantum-classical computation, QFaaS facilitates quantum application development in cloud environments. It integrates multiple quantum development kits (Qiskit, Q#, Cirq, and Braket), quantum simulators, and cloud providers like IBM Quantum and Amazon Braket. The framework mitigates cold start issues and automates backend selection, enabling developers to create quantum functions across different platforms. Experimental results showcase QFaaS’s potential, though some limitations of quantum serverless computing require further research.

In “**Pattern-based Serverless Data Processing Pipelines for Function-as-a-Service Orchestration Systems**”, Mathew et al. [8] address the challenge of vendor lock-in in serverless computing, particularly for FaaS orchestration, which relies on vendor-specific languages to build complex data processing pipelines. To mitigate this, the authors propose a generic, provider-agnostic model for serverless pipelines using well-established enterprise integration and workflow patterns. These patterns are mapped to vendor-specific orchestration languages, enabling the transformation of provider-agnostic models into executable workflows across different platforms. An industrial case study involving Garmin demonstrates the approach’s effectiveness, with positive results in execution efficiency. The authors suggest extending the method to additional FaaS orchestrators and propose building a community to further refine and expand the pattern-based approach.

3.3. *Edge Computing and IoT*

Tusa et al. [15], in “**Microservices and Serverless Functions – Lifecycle, Performance, and Resource Utilisation of Edge-based Real-time IoT Analytics**”, conduct a performance evaluation of Microservices and FaaS functions for real-time IoT analytics in edge computing environments. Microservices generally offer better latency and resource efficiency, especially with large data streams, but require more developer effort for deployment. Serverless functions, such as those in OpenFaaS and similar tools, simplify development and provide auto-scaling capabilities, though they may suffer from higher latency in parallel processing scenarios. The authors conclude that the choice between these architectures depends on the specific IoT use case and the constraints of edge resources.

Mahdizadeh and Abrishami [7], in “**An assignment mechanism for workflow scheduling in Function as a Service edge environment**”, address the challenge of deploying FaaS applications in edge computing, where resource constraints and geographic limitations hinder efficient task execution. The authors propose a dynamic resource allocation strategy that models FaaS workflows as directed acyclic graphs and introduces two methods: Highest Bid First Mechanism (HBFM) and Warm Function First Mechanism (WFFM). These methods aim to optimize resource use, prioritize critical tasks, and reduce workflow completion time. Experimental results demonstrate that these mechanisms outperform traditional methods in both efficiency and fairness in resource allocation.

In another work on microservices applications, Hussain and Amini-Salehi [5] focused on improving the resiliency of Industry 4.0 applications deployed on resource-limited fog computing systems at remote industrial sites, in their paper titled “**Resource Allocation of Industry 4.0 Micro-service Applications Across Serverless Fog Federation**”. The authors propose a serverless platform for fog federation to enhance elasticity and manage real-time, fault-intolerant workloads. Their approach includes partitioning microservice-based workflows using a graphbased model and developing a Bayesian resource allocation strategy to maximize on-time task completion. Experimental results show that the proposed methods increase deadline meet rates by 15-18% and improve system scalability compared to existing techniques, making it more effective in oversubscribed and disaster-prone environments.

Khansari and Sharifian [3] in a closely related area proposed a novel IoT microservice composition approach hosted on the fog layer in their article titled “**A Scalable Modified Deep Reinforcement Learning Algorithm for Serverless IoT Microservice Composition Infrastructure in the Fog Layer**”. Their approach uses a modified Deep Reinforcement Learning (DRL) algorithm that dynamically adapts to changes in the IoT environment while considering cloudlet computing capacities and link bandwidths. The serverless architecture combines the advantages of serverless and fog computing, enabling efficient resource provisioning, scalability, low latency, and high security. The proposed DRL-based Microservice Chaining at Fog Layer (DRLMCF) algorithm optimizes microservice composition by minimizing resource utilization and reducing delays. The DRLMCF algorithm operates autonomously using a Partially Observable Markov Decision Process (POMDP) and can scale without relying on a central controller, making it ideal for distributed IoT environments. Evaluations demonstrate significant improvements, including up to a 57.3% reduction in end-to-end delay and an 84% increase in successfully chained microservices compared to existing methods.

3.4. *Machine Learning and AI Integration*

“**Efficient and scalable covariate drift detection in machine learning systems with serverless computing**” by Sisniega et al. [2] introduces a serverless-based architecture for efficient batch covariate drift detection in machine learning (ML) systems, focusing on edge-to-cloud environments. By separating ML inference from drift detection tasks, the proposed solution improves scalability, cost-effectiveness, and system reliability. Future work aims to extend this approach to handle streaming data and incorporate broader ML model monitoring capabilities, including performance tracking and explainability features.

Huang et al. [12] in “**GeoPM-DMEIRL: A Deep Inverse Reinforcement Learning Security Trajectory Generation Framework with Serverless Computing**” present a framework for generating high-utility vehicle trajectories while protecting privacy. The method leverages AWS Lambda for training reinforcement learning models and applies a Geo-aware local differential privacy mechanism (GeoPM) to perturb trajectories while adhering to real-world traffic rules. Using A2C reinforcement learning and Deep Maximum Entropy Inverse Reinforcement Learning, the system optimizes trajectory generation. Experimental results show that their approach improves data utility by 54.6% and enhances privacy protection by 30.7%. Serverless computing significantly reduces the model training time compared to local training methods.

3.5. *Workflow and Application Composition*

In “**Serverless Application Composition Leveraging Function Fusion: Theory and Algorithms**”, Czentye and Sonkoly [1] introduce novel algorithms employing function fusion techniques to address the complexities of state management and multicore resources in serverless applications. The research tackles the NP-complete problem of latency-constrained tree partitioning, proposing a bicriteria approximation scheme and a greedy heuristic for deriving cost-efficient configurations quickly. Extensive simulations demonstrate that these methods outperform existing solutions in runtime performance, and further cost reductions of 3-6% are possible with allowable latency violations.

Ristov et al. [11], in their article titled “**CODE: Code once, deploy everywhere serverless functions in federated FaaS**” introduce the CODE framework, designed to simplify the deployment of serverless functions in federated FaaS environments. CODE significantly reduces developer effort and the lines of code (LoC) needed, by up to 81.8% compared to traditional FaaS provider SDKs and up to 9.23× less than Infrastructure-as-Code (IaC) frameworks. It also allows dynamic selection of storage providers and supports faster deployment, with AWS deployment being up to 6× quicker than on GCP. The framework promotes a “code once, deploy everywhere” approach, facilitating multi-region and multi-provider FaaS deployments.

Finally, the article “**ExDe: Design space exploration of scheduler architectures and mechanisms for serverless data-processing**” by Talluri et al. [13] presents a framework to explore the complex design space of scheduling architectures and mechanisms in serverless data-processing systems. The framework provides an easy-to-use abstraction for testing and evaluating different scheduler designs. ExDe enables system designers to examine various architectures and mechanisms, such as fat nodes and work-stealing, to achieve the same objective of low task slowdown. The framework is open-source and allows for extensive performance evaluations in a cost-effective and timely manner, promoting the exploration of various scheduling designs.

4. Research Challenges and Future Directions

While the articles in this SI offer valuable insights, key challenges in serverless computing remain, particularly in resource management, stateful execution, and QoS guarantees. These gaps continue to limit serverless computing's full potential for the next-generation of cloud-native applications. In this section, we highlight some of the unresolved issues and future research directions.

- **Resource Management and Scheduling**: Serverless systems face challenges in optimizing resource allocation and task scheduling. Future research should focus on developing more efficient, adaptive schedulers that can handle diverse workloads, minimize latency, and optimize cost-performance trade-offs, particularly in complex, distributed environments like edge and compute continuum.
- **QoS and Latency Guarantees**: Serverless architectures often lack QoS guarantees, especially in latency-sensitive applications. Research should address how to offer better QoS assurances and reduce unpredictable latency in serverless platforms, potentially through hybrid approaches combining serverless with other computing paradigms.
- **State Management**: Managing state across stateless serverless functions is difficult. Research should focus on designing frameworks that enable more efficient state handling, allowing for better coordination and performance in stateful applications without violating the principles of serverless.
- **Cost Optimization**: While serverless offers cost-efficiency, pricing models can become complex with unpredictable workloads. Research should explore new cost models and optimization techniques to make serverless platforms more transparent and cost-effective for users.
- **Application Portability and Interoperability**: With various cloud providers offering serverless platforms, ensuring seamless migration and interoperability between different environments remains a challenge. Future research should explore standards and frameworks that allow for greater portability across heterogeneous serverless environments.
- **Edge, Compute Continuum and IoT Integration**: As serverless computing extends to edge and IoT devices, challenges related to limited resources, network variability, and energy efficiency arise. Research should focus on adapting serverless models to edge computing and compute continuum environments, ensuring efficient function execution and resource management in constrained and geographically dispersed settings.
- **Debugging and Monitoring**: Debugging and monitoring serverless applications remain difficult due to their distributed and stateless nature. Future research should aim at developing advanced tools that improve visibility, debugging, and performance tracking for serverless functions in real-time.

Acknowledgments

We would like to express our sincere gratitude to the Editor-in-Chief, Prof. *Michela Taufer*, and the entire team at Future Generation Computer Systems (FGCS) for their invaluable support in bringing this SI to fruition.

The success of SI would not have been possible without the timely and valuable contributions of our reviewers. We sincerely appreciate the significant time and effort they dedicated to evaluating manuscripts and offering insightful feedback that greatly enhanced the quality of the submissions. It is our pleasure to extend our thanks to the all reviewers, listed below in alphabetical order.

- Alireza Furutanpey (*TU Vienna, Austria*)
- Anna Kobusin'ska (*Poznan' University of Technology, Poland*)
- Anupama Mampage (*University of Melbourne, Australia*)
- Arun Ravindran (*University of North Carolina at Charlotte, US*)
- Ashiq Anjum (*University of Leicester, UK*)
- Ciprian Dobre (*University Politehnica of Bucharest, Romania*)
- Claudio Cicconetti (*Institute of Informatics and Telematics – National Research Council, Italy*)
- Cristina Abad (*Escuela Superior Politécnica del Litoral, Ecuador*)
- Darko Andrŏcec (*University of Zagreb, Croatia*)
- David Bermbach (*TU Berlin, Germany*)
- Dena Petvu (*West University of Timisoara, Romania*)
- Donkyu Baek (*Chungbuk National University, South Korea*)
- Florin Pop (*University Politehnica of Bucharest, Romania*)
- George Kousiouris (*Harokopio University of Athens, Greece*)
- Gwanggil Jeon (*Incheon National University, South Korea*)
- Hadi Tabatabaee Malazi (*University College Dublin, Ireland*)
- Huaqing Li (*Southwest University, China*)
- Istv́an Pelle (*Budapest University of Technology and Economics, Hungary*)
- Jim Basilakis (*Western Sydney University, Australia*)
- Jõao Henriques (*University of Coimbra,Portugal*)
- Jośe Sim̃ao (*Polytechnic Institute of Lisbon, Portugal*)
- Joseph Doyle (*Queen Mary University of London, UK*)
- Laiping Zhao (*Tianjin University, China*)
- Loris Belcastro (*University of Calabria, Italy*)
- Luciano Baresi (*Politecnico di Milano, Italy*)
- Mustafa Rafique (*Rochester Institute of Technology, US*)
- Mahdi Abbasi (*Bu-Ali Sina University, Iran*)
- Marco A. S. Netto (*Microsoft, US*)
- Marco Lapegna (*University of Naples Federico II, Italy*)
- Maria Rodriguez (*University of Melbourne, Australia*)
- Martin Grambow (*TU Berlin, Germany*)
- Massimo Villari (*University of Messina, Italy*)
- Mauro Femminella (*University of Perugia, Italy*)
- Milŏs Simíc (*University of Novi Sad, Serbia*)
- Mohammad Reza Heidarpour (*Isfahan University of Technology, Iran*)
- Mohammad S. Aslanpour (*Monash University, Australia*)
- Mohsen Amini Salehi (*University of North Texas, US*)
- Muhammad Sadaqat Janjua (*Purdue University, US*)
- Nadia Zeghib (*Universit́e Constantine 2, Algeria*)
- Narayanamoorthi Thilagavathi (*Anna University, India*)
- Nikola Tankovíc (*Juraj Dobrila University of Pula, Croatia*)
- Óscar Castellanos-Rodŕiguez (*Universidade da Coruña, Spain*)
- Pablo Serrano (*Universidad Carlos III de Madrid, Spain*)
- Patrizio Dazzi (*University of Pisa, Italy*)
- Philipp Raith (*TU Wien, Austria*)
- Praveen Kumar Donta (*Stockholm University, Sweden*)
- Redowan Mahmud (*Curtin University, Australia*)
- Rodrigo Calheiros (*University of Western Sydney, Australia*)
- Saeed Sharifian (*Amirkabir University of Technology, Iran*)
- Saeid Abrishami (*Ferdowsi University of Mashhad, Iran*)
- Sashko Ristov (*University of Innsbruck, Austria*)
- Sebastian Werner (*TU Berlin, Germany*)
- Shitharth Selvarajan (*Leeds Beckett University, UK*)
- Shreshth Tuli (*Imperial College London, UK*)
- Siddharth Agarwal (*University of Melbourne, Australia*)
- Sınică Alboaie (*Universitatea Alexandru Ioan Cuza Iasi, Romania*)
- Stefan Nastic (*TU Wien, Austria*)
- Sukhpal Gill (*Queen Mary University of London, UK*)
- Tessema Mindaye Mengistu (*Virginia Tech, US*)
- Thiago Emmanuel Pereira (*Universidade Federal de Campina Grande, Brazil*)
- Thomas Pusztai (*TU Wien, Austria*)
- Tianzhang He (*University of Melbourne, Australia*)

- Tobias Pfandzelter (*Technische Universität Berlin, Germany*)
- Tyler Skluzacek (*Laboratory in Oak Ridge, US*)
- Valeria Cardellini (*University of Rome Tor Vergata, Italy*)
- Valerio Bellandi (*Universit`a degli Studi di Milano, Italy*)
- Virginia Pilloni (*University of Cagliari, Italy*)

Closing Remarks

This special issue highlighted the key advances in serverless computing, including performance optimization and cold start management, serverless architecture and infrastructure, edge computing and IoT, machine learning and AI integration, and workflow and application composition. The accepted articles reflect the growing breadth of serverless computing and the ongoing efforts to overcome technical challenges in this space.

We hope this collection inspires further exploration and innovation in serverless computing. We thank all authors, reviewers, and contributors for their efforts in making this issue a success.

References

[1] János Czentye, Balázs Sonkoly, Serverless application composition leveraging function fusion: theory and algorithms, Future Gener. Comput. Syst. 153 (2024) 403–418.
[2] Jaime Céspedes Sisniega, Vicente Rodríguez, Germán Moltó, Álvaro López García, Efficient and scalable covariate drift detection in machine learning systems with serverless computing, Future Gener. Comput. Syst. 161 (2024) 174–188.
[3] Mina Emami Khansari, Saeed Sharifian, A scalable modified deep reinforcement learning algorithm for serverless iot microservice composition infrastructure in fog layer, Future Gener. Comput. Syst. 153 (2024) 206–221.
[4] Daniel Fireman, Paulo Silva, Thiago Emmanuel Pereira, Luis Mafra, Dalton Valadares, Prebaking runtime environments to improve the faas cold start latency, Future Gener. Comput. Syst. 155 (2024) 287–299.
[5] Razin Farhan Hussain, Mohsen Amini Salehi, Resource allocation of industry 4.0 micro-service applications across serverless fog federation, Future Gener. Comput. Syst. 154 (2024) 479–490.
[6] Vasileios Katevas, Georgios Fatouros, Dimosthenis Kyriazis, George Kousiouris, Embedding automated function performance benchmarking, profiling and resource usage categorization in function as a service devops pipelines, Future Gener. Comput. Syst. 160 (2024) 223–237.
[7] Samaneh Hajy Mahdizadeh, Saeid Abrishami, An assignment mechanism for workflow scheduling in function as a service edge environment, Future Gener. Comput. Syst. 157 (2024) 543–557.
[8] Anil Mathew, Vasilios Andrikopoulos, Frank J. Blaauw, Dimka Karastoyanova, Pattern-based serverless data processing pipelines for function-as-a-service orchestration systems, Future Gener. Comput. Syst. 154 (2024) 87–100.
[9] Tam n. Nguyen, Holistic cold-start management in serverless computing cloud with deep learning for time series, Future Gener. Comput. Syst. 153 (2024) 312–325.
[10] Hoa T. Nguyen, Muhammad Usman, Rajkumar Buyya, Qfaas: a serverless function-as-a-service framework for quantum computing, Future Gener. Comput. Syst. 154 (2024) 281–300.
[11] Sashko Ristov, Simon Brandacher, Mika Hautz, Michael Felderer, Ruth Breu, Code: code once, deploy everywhere serverless functions in federated faas, Future Gener. Comput. Syst. 160 (2024) 442–456.
[12] Yi rui Huang, Jing Zhang, Hong ming Hou, Xiu cai Ye, Yi Chen, Geopm-dmeirl: a deep inverse reinforcement learning security trajectory generation framework with serverless computing, Future Gener. Comput. Syst. 154 (2024) 123–139.
[13] Sacheendra Talluri, Nikolas Herbst, Cristina Abad, Tiziano De Matteis, Alexandru Iosup, Exde: design space exploration of scheduler architectures and mechanisms for serverless dataprocessing, Future Gener. Comput. Syst. 153 (2024) 84–96.
[14] Minh-Ngoc Tran, YoungHan Kim, Optimized resource usage with hybrid auto-scaling system for knative serverless edge computing, Future Gener. Comput. Syst. 152 (2024) 304–316.
[15] Francesco Tusa, Stuart Clayman, Alina Buzachis, Maria Fazio, Microservices and serverless functions—lifecycle, performance, and resource utilisation of edge based real-time iot analytics, Future Gener. Comput. Syst. 155 (2024) 204–218.
[16] Sebastian Werner, Stefan Tai, A reference architecture for serverless big data processing, Future Gener. Comput. Syst. 155 (2024) 179–192.
[17] Óscar Castellanos-Rodríguez, Roberto R. Expósito, Jonatan Enes, Guillermo L. Taboada, Juan Touriño, Serverless-like platform for container-based yarn clusters, Future Gener. Comput. Syst. 155 (2024) 256–271.

Adel N. Toosi[a,*], Bahman Javadi[b], Alexandru Iosup[c], Evgenia Smirni[d], Schahram Dustdar[e]
[a] *School of Computing and Information Systems, The University of Melbourne, Melbourne, VIC, Australia*
[b] *School of Computer, Data and Mathematical Sciences, Western Sydney University, Penrith, NSW, Australia*
[c] *Vrije Universiteit Amsterdam, Amsterdam, North Holland, Netherlands*
[d] *Department of Computer Science, William and Mary, Williamsburg, VA, USA*
[e] *Distributed Systems Group, TU Wien, Vienna, Austria*

[*] Corresponding author.
E-mail address: adel.toosi@unimelb.edu.au (A.N. Toosi).