ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 9, No. 2, 2024

SERVERLESS AI AGENTS IN THE CLOUD

Oleh Chaplia[1], *Halyna Klym*[1], *Edgars Elsts*[2]

[1]*Lviv Polytechnic National University, 12, Bandera Str, Lviv, 79013, Ukraine,*
[2]*Institute of Solid State Physics, University of Latvia, 8, Kengaraga, Riga, LV-1063, Latvia.*
Authors' e-mails: *oleh.y.chaplia@lpnu.ua, halyna.i.klym@lpnu.ua, edgars.elsts@cfi.lu.lv*

https://doi.org/10.23939/acps2024.02.115

Submitted on 15.10.2024

***Abstract*: Integrating AI agents within serverless architectures offers a modern approach to deploying and executing intelligent applications. Leveraging the advantages of serverless computing, AI agents can dynamically respond to varying workloads without the overhead of managing the underlying infrastructure. This article explores the concept of scalable serverless AI agents in the cloud, detailing their architecture, benefits and drawbacks, challenges, and real-world applications. The paper provides advantages and drawbacks of the serverless approach. Then a proof-of-concept has been developed, deployed and tested. The AI agent code was deployed to Azure Functions, Google Cloud Functions, and AWS Lambda and tested. As a result, improvements to availability, resilience, reliability, and scalability qualities have been proposed to mitigate the previously defined drawbacks.**

***Index Terms*: cloud computing, cyber-physical systems, AI agents, microservices, serverless**

I. INTRODUCTION

The rapid advancements in Cyber-Physical Systems (CPS), Internet of Things (IoT), and robotics have increasingly relied on intelligent systems to enable autonomous and efficient operations [1]. Self-adaptive AI, robotic and CPS systems enhances the integration with the environments with higher-level process-oriented systems [2].

AI approaches, machine learning and deep learning, LLMs, and modern AI agents play a crucial role in these domains, leveraging complex reasoning capabilities, semantic processing, and integration with physical environments through sensors and actuators [3].

Deploying these agents in the cloud offers significant advantages, including scalability, accessibility, and cost-effectiveness. Among various cloud deployment models, microservices and serverless architectures have gained prominence due to their ability to dynamically manage resources and simplify operations [4]. However, the unique demands of CPS, IoT, and robotics pose challenges to serverless systems, particularly in terms of availability, resilience, reliability, and scalability, data security, privacy, latency for real-time applications [5].

Modern AI agents are commonly a software programs with the deterministic or non-deterministic flows. Recent advances in LLMs pushed the boundaries for agents into more non-deterministic operation, including Natural Language Processing (NLP) and generative AI approaches [6].

New agent frameworks AutoGEN, LangChain, LangGraph and similar help the agent software development by automating many interactions between the agents, LLMs and tool calling [7]. These software agents can be deployed to the cloud and orchestrated. Best practices like modularity and independence, presented in microservice and serverless architecture, help to manage swarms of these agents.

The Azure Functions, Google Cloud Functions, and AWS Lambda platforms enable the execution of event-driven functions that automatically scale in response to demand, providing flexibility and cost-efficiency [8]. The usage of serverless architectures provides the ability to dynamically scale, provide reliable, high-available solutions and handle various workloads [9]. Additionally, cloud-based AI agents benefit from reduced infrastructure management overhead.

This research aims to evaluate the effectiveness of serverless cloud architectures, using Azure Functions, Google Cloud Functions, and AWS Lambda, for deploying AI agents. Research paper provides the literature review and analysis of their advantages and drawbacks concerning availability, resilience, reliability, and scalability, while addressing challenges in execution and deployment. AI agents powered by LLMs and supplied with inputs, outputs, and knowledge, are assessed in terms of integration into dynamic scalable environments. The paper proposes improvements to mitigate identified drawbacks, and contributes to the optimization of AI agents operation and deployments.

II. LITERATURE REVIEW AND PROBLEM STATEMENT

This section presents a literature review, exploring the challenges, existing solutions, cloud architectures, system designs, and their advantages and limitations. The first research question (RQ1) is addressed in this section.

The development and deployment of AI agents for CPS, IoT, and robotics presents unique challenges, particularly in the areas of availability, resilience, reliability, and scalability. AI agents are characterized by their ability to use knowledge, interact with external tools and systems, and execute tasks autonomously in dynamic and often unpredictable environments [3].

AI agents in cloud environments, running within serverless containers to execute functions, enable advanced capabilities like managing agent swarms and inter-agent function calls, but also introduce development and deployment complexities [10].

Microservices architecture, similarly to serverless, recommends developing and deploying small independent services. These best practices can be applied to code of AI agents, which may be optimized for a serverless invocation and orchestration [11].

Serverless computing has transformed microservice deployment by abstracting infrastructure management and enabling scalability, yet it poses challenges in state management, fault tolerance, and ensuring exactly-once execution semantics amid failures and re-executions [12].

Integrating serverless with IoT highlights Fog and Edge computing's potential to enhance performance through decentralization, but reliance on centralized architectures raises issues like vendor lock-in and data privacy [13].

Traffic simulation using microservices, agents, and RESTful interfaces offers scalability, configurability, semantic interoperability, and support for diverse agent frameworks but faces challenges like network overhead and microservice management complexity [14].

A multi-agent framework leveraging Large Language Models (LLMs) was proposed to enhance productivity and reliability in industrial task automation. While demonstrating collaborative potential, further research is needed to optimize agent interactions and refine operational protocols [15].

Enhanced resource management and task scheduling was achieved by using multi-agent system approach [16]. Better solutions needed to improve the reliability of cloud applications while accommodating the inherent complexities of microservice architectures and MAS.

Based on the conducted literature review, RQ1—which asks about the advantages and drawbacks of using serverless cloud architectures for deploying AI agents in CPS and robotics—can now be addressed.

AI agents deployed in serverless and microservice architectures address the dynamic demands of CPS, IoT, and Robotics. Serverless systems ensure availability through redundancy, resilience via automatic failure recovery, reliability through error handling and robust infrastructure, and scalability by adjusting resources in real-time under cloud-managed SLAs. However, challenges persist, including latency from edge-cloud distances, security and privacy risks from data transmission, state management complexities due to stateless functions, vendor lock-in, and integration difficulties with CPS infrastructure. Addressing these requires architectural strategies such as edge computing, encryption, and multi-cloud solutions for secure, efficient deployments.

III. SCOPE OF WORK AND OBJECTIVES

The main objective of this research is to evaluate the effectiveness of serverless architecture for AI agents in the cloud for CPS, IoT, and robotics.

This paper examines the cloud system architecture, execution and deployment of AI agents within serverless cloud architecture—specifically Azure Functions, Google Cloud Functions, and AWS Lambda—and the agent graph execution. The paper defines advantages and drawbacks of using three serverless platforms for agent execution environment. The scope of this work covers an analysis of key properties, challenges and issues inherent in integrating these agents into environments. The study reviews at least fifteen research papers, which were filtered, to synthesize current methodologies, identify best practices, and highlight gaps in existing knowledge. Then, based on the defined drawbacks the improvements are proposed to the associated execution and deployment in the scope of availability, resilience, reliability, and scalability qualities.

The research seeks to address the following questions. The first research question (RQ1) asks what the advantages and drawbacks are of using serverless cloud architectures for deploying AI agents concerning availability, resilience, reliability, and scalability. The second research question (RQ2) asks how Azure Functions, Google Cloud Functions, and AWS Lambda compare in supporting the deployment and operational efficiency of AI agents in CPS, IoT and robotics applications. The third research question (RQ3) asks to propose an availability, resilience, reliability, and scalability improvements to mitigate the previously defined drawbacks.

Through addressing these objectives, the paper aims to contribute to the improvement of intelligent systems in serverless cloud environments.

IV. SERVERLESS ARCHITECTURE FOR AI AGENTS

This section defines a serverless cloud system architecture and its components for executing AI Agents. The first (RQ1) and second research questions (RQ2) are addressed in this section. This is the reference architecture for the research paper, which is used for modelling, gathering metrics, testing, and validation. This architecture may be modified for other specialized needs according to the system requirements.

Fig. 1 presents a high-level architecture model illustrating the interaction between components in a distributed system integrating microservices, serverless environments, edge processing, data storages, sensors and actuators, external services, message queues, monitoring and logging.

Web apps are providing the user interface for the users. Microservices are deployed to the execution environments via CI/CD pipelines. Microservice scope contains servers, web apps and data storage for controlling, monitoring, operating the AI agents. AI agents use a CI/CD pipeline for deployment to the cloud-native serverless execution environment. Agents are connected to the data storage, messaging queues, external services, monitoring and logging systems.

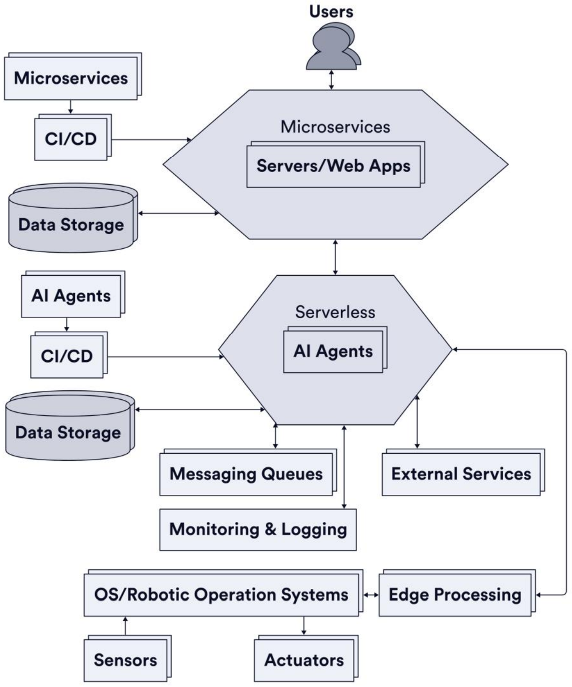

Fig. 1. Cloud system architecture

Fig. 2 illustrates the architecture of an AI agent. The agent code, which is an agent graph or an algorithm, serves as the central orchestrator, connecting key components: memory, reasoning, a Large Language Model (LLM), tools, sensors, actuators, and external services.

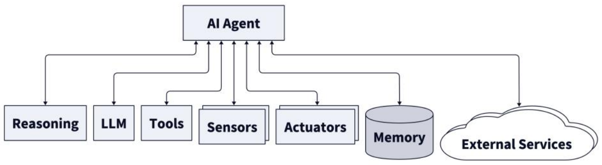

Fig. 2. The architecture of the AI agent

Reasoning module processes inputs to derive actionable insights. Memory provides persistent storage for contextual information, enabling stateful operations. Sometimes it is necessary to have an internal state cache to handle the execution flow. The LLM enhances natural language understanding and decision-making, while tools extend the agent's capabilities for specialized computational tasks. External services provide additional computational resources and functionalities via APIs. This architecture facilitates efficient interaction between sensing, reasoning, and actuation in complex systems.

Agents can also interact with edge devices. Edge processing capabilities allow the system to maintain partial functionality even during network disruptions. Edge devices have bidirectional connections with multiple subsystems, including Robots, CPS, and IoT devices. Sensors capture real-world inputs, which are processed to inform the agent's decisions, and actuators execute these decisions in the physical environment. As a result, the system contains the cloud computing components connected to the physical world and vice versa. This may be useful for building digital twins, simulations, and two-way interactions.

The proposed system leverages modular microservices and serverless execution environments to scale horizontally and vertically to varying workloads with fine-grained resource allocation. Computational resources are provisioned and de-provisioned in near real-time. This architecture allows for efficient scaling not only within cloud environments but also at the edge, where processing workloads can be distributed to edge nodes for local execution, reducing latency and central server load.

High availability is ensured through redundancy mechanisms across the system's microservices, serverless functions, and data storage. Redundancy is provided by the cloud provider using regions, availability zones, and geography according to their SLAs. The integration of messaging queues and pub/sub frameworks provides asynchronous communication, decoupling service dependencies and ensuring message delivery during failures. Sometimes direct requests between the agents may fail, therefore it is valuable to use message queues. The system's resilience and reliability are enhanced through fault-tolerant design, including fail-safe agent code, retry patterns, circuit breakers, and recovery for non-deterministic exceptions from LLM responses or undefined behavior.

The architecture incorporates reliable data storage on the cloud provider's end that support replication and synchronization across multiple nodes for data integrity and availability. OS and Robotic operation systems (ROS) are designed as is, however they use fail-safe mechanisms to handle disruptions in sensor or actuator data. The use of monitoring and logging across the overall system facilitates proactive fault detection and remediation.

Fig 3. highlights the interaction between users, microservices, AI agents, and various connected systems. At the top level, Users interact with microservices, which serve as the intermediary layer, connecting to AI Agents for advanced computational capabilities. These AI agents form the central processing layer, acting as the intelligence hub within the system.

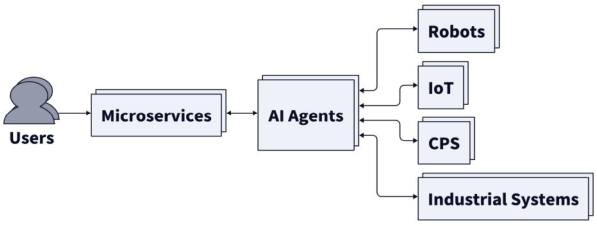

Fig. 3. Use cases for the cloud system

V. PROOF-OF-CONCEPT IMPLEMENTATION

This section presents a summary of the proof-of-concept implementation. According to the previously defined model a simplified version of the model was developed, deployed, and tested. The critical parts of the proposed system architecture model were implemented and deployed to the cloud.

The focus was on the AI agent's implementation. Each AI agent was represented as a piece of the software written in Python. These AI agents utilized LLMs and external tools to perform data analysis, decision-making, and control actions. AutoGEN and LangGraph were used. Leveraging serverless platforms, each AI agent was deployed as an independent function. Manual and automated scaling mechanisms from the cloud provider were tested. The same AI agents were deployed across three major cloud platforms: Azure Functions, Google Cloud Functions, and AWS Lambda.

The deployment phase involved configuring each serverless platform to host the AI agents, setting up event triggers and connections. System's performance was tested against availability, resilience, reliability, and scalability metrics. Automated deployment pipelines were established using CI/CD tools. Additionally, resilience was tested by inducing controlled failures within the system to verify the agents' capability to recover and maintain operational continuity. Reliability assessments focused on uptime statistics and error rates.

The results from the PoC implementation provided insights, strengths and limitations of each serverless platform in the context of supporting AI-driven CPS, IoT, and robotics applications.

VI. PROOF-OF-CONCEPT RESULTS

This section presents the gathered results and metrics of the system and proposed improvements to the system. The developed agents were deployed to the cloud and metrics were gathered. The second (RQ2) and third research question (RQ3) are addressed in this section.

Deployment times varied slightly across platforms, with Azure Functions taking between 30 and 70 seconds, Google Cloud Functions between 30 and 80 seconds, and AWS Lambda demonstrating slightly faster times, ranging from 30 to 60 seconds.

Cold start latency, which measures the delay before function execution after an idle period, ranged from 0.5 to 5 seconds for Azure Functions, 0.5 to 3 seconds for Google Cloud Functions, and 0.5 to 1 second for AWS Lambda, indicating superior cold start performance for AWS Lambda. Hot start latency is minimal at less than one second for each provider.

Maximum execution time limits varied across platforms. The Execution Time Limit varies slightly, with Azure Functions permitting up to 10 minutes, GCF allowing 9 minutes, and AWS Lambda extending to 15 minutes.

The number of fail points in the agent graph is contingent on the complexity of the workflow, including the number of nodes and tools used. This factor is consistent across all three platforms, indicating that workflow design is critical for minimizing failures.

Scalability is inherently managed by all platforms, with Azure Functions automatically scaling up to 200 instances, GCF accommodating up to 1000 instances for v2 functions, and AWS Lambda handling a default regional concurrency limit of 1,000, which can be further scaled upon request.

Resilience is ensured through multi-region and multi-zone deployment capabilities. Azure Functions supports deployment across regions and Availability Zones, Google Cloud Functions employs geographic redundancy, and AWS Lambda operates across multiple Availability Zones within a region to tolerate localized failures.

Reliability is managed by integrated monitoring and logging tools: Azure Functions leverages Azure Monitor. Google Cloud Functions utilizes Cloud Monitoring. AWS Lambda integrates with Amazon Cloud Watch.

All platforms follow a pay-per-use pricing model with variations in execution costs. Finally, Costs are structured on a pay-per-use basis, with Azure Functions and AWS Lambda charging $0.20 per million executions after the free tier, whereas GCF is priced slightly higher at $0.40 per million invocations post free tier.

This comparative analysis underscores the strengths and slight variances among the serverless platforms. Analysis provides insights for selecting the most appropriate service.

VII. IMPROVING THE PROPERTIES OF SERVERLESS CLOUD SYSTEM

This section provides a description of the methods and approaches for improvements based on the simplified cloud system model. The third research question (RQ3) is addressed in this section.

Each agent operates independently and has its own probability of failure. The system is considered fully operational only when all components are functioning correctly. Failure events can occur at multiple levels: cloud provider failures, cloud infrastructure failures, virtualization failures, operating system-level failures, microservice-specific failures, unexpected events, dependency issues, network problems, infrastructure issues, third-party dependency failures, security vulnerabilities, configuration errors, failures due to high load, deployment issues, system errors, or microservice runtime errors. While microservice developers, DevOps engineers, and IT experts can control and manage cloud infrastructure resources, custom virtualization configurations, and the microservices themselves, they cannot maintain the underlying cloud provider's infrastructure.

To improve scalability, serverless functions should be stateless and idempotent by design, allowing for horizontal scaling without complications related to state management. The next step is to define whether the agent graph is loaded dynamically or exist inside the function/lambda code. Having the agent graph and its properties in environment variables or loaded dynamically from data storage provides more flexibility. Even when the agent graph is not complex, e. g. has up to 3 nodes or they are serially executed, the probability of having some error increases. The same is for code complexity.

Therefore, to reduce the agent function complexity we need to use function templates, keep the agent logic in environment variables or dynamically load from the database. The graph should be validated before the execution and deployment. Each function/lambda need to have an internal mechanism of error fail-safe handling or fail-fast when the business logic allows it.

Availability may be enhanced by deploying serverless functions across multiple regions and availability zones provided by the cloud provider. At least 3 zones are recommended. It is important to balance the physical distance between the regions and zones to reduce the network latency. This geographic distribution mitigates the impact of localized outages and ensures continuous operation. Implementing health checks and automated failover mechanisms allows the system to reroute traffic to healthy instances seamlessly in case of failures.

For the current AI agent's precise validation end error handling is necessary to improve the reliability. Retry mechanisms for the tool calling, handling various responses from LLMs, non-deterministic reasoning should be scoped within the valid range of operations. Utilizing monitoring tools like AWS CloudWatch or Azure Monitor provides real-time insights into system performance and errors.

Implementing circuit breaker patterns allows the system to isolate failing components and prevent cascading failures. Conducting chaos engineering experiments helps identify system weaknesses by intentionally introducing faults and observing system responses. This proactive approach leads to a more robust system capable of withstanding the unpredictability of CPS and IoT deployments.

Real-world cloud systems are open to errors due to the inherent instability of execution environment. These systems incorporate hardware devices, edge computing tools, and real-time interactions with physical environments, making them vulnerable to disruptions such as network fluctuations, hardware failures, and environmental variability. However, the responsibility for these issues is on the cloud provider.

Edge computing introduces significant challenges, including intermittent connectivity, power instability, and limited resources in decentralized environments. These issues impact both reliability and overall system performance, underscoring the need for fault-tolerant architectures to ensure adaptability in dynamic conditions. Consequently, these challenges highlight the critical need for the development of robust, fault-tolerant, and reliable architectures that can adapt to dynamic conditions.

VIII. CONCLUSION

This research explores scalable AI agents in the cloud and tests the effectiveness of serverless cloud architectures. Azure Functions, Google Cloud Functions, and AWS Lambda were tested for deploying AI agents. Critical properties such as availability, resilience, reliability, and scalability were investigated. The study highlights the strengths and drawbacks of serverless architectures.

Features like multi-region deployments, automatic failover mechanisms, and inherent scalability make serverless architectures highly adaptable to the dynamic and variable workloads characteristic of CPS, IoT, and robotics applications. However, several limitations, such as latency caused by cold starts, resource constraints on execution time and memory, and integration challenges with real-time systems, which can impact time-sensitive and resource-intensive AI agent workflows.

Through a comparative analysis addressing RQ2, AWS Lambda demonstrates slight advantages in SLA availability (99.95%), execution time limits (15 minutes), and scalability (default concurrency of 1,000 instances), making it a strong candidate for demanding applications. Azure Functions and GCF are competitive, where GCF excelling in maximum instance scalability (up to 1,000 instances for v2 functions). Despite these differences, all three platforms share common strengths, such as reliability through robust error handling and resilience mechanisms like fault isolation and failover strategies. Cost analysis revealed that Azure Functions and AWS Lambda are more economical, charging $0.20 per million executions, compared to GCF's $0.40 per million invocations.

To address the limitations identified in RQ3, several strategies are proposed to enhance serverless deployments. Multi-region and multi-zone deployments, combined with health checks and automated failover, improve availability by mitigating localized outages. Resilience can be strengthened through circuit breaker patterns and chaos engineering to prevent cascading failures and identify vulnerabilities. Reliability is enhanced with robust error handling, idempotent operations, and real-time monitoring for consistent performance and fault recovery. Scalability can be improved by leveraging event-driven architectures, stateless designs, and edge processing to reduce latency and central server load. These enhancements establish serverless architectures as a robust and scalable foundation for AI agents in CPS, IoT, and robotics, supporting intelligent systems in complex, dynamic environments.

References

[1] Serôdio, C., Mestre, P., Cabral, J., Gomes, M., & Branco, F. (2024). Software and Architecture Orchestration for Process Control in Industry 4.0 Enabled by Cyber-Physical Systems Technologies. Applied Sciences, 14(5), Article 5. DOI: 10.3390/app14052160

[2] Guldner, A., et al. (2023). A framework for AI-based self-adaptive cyber-physical process systems. it -

Information Technology, 65(3), 113–128. DOI: 10.1515/itit-2023-0001

[3] Goel, S. (2024). Towards building Autonomous AI Agents and Robots for Open World Environments. New Zealand.

[4] Chaplia, O., Klym, H., & Popov, A. I. (2024). An Approach to Improving Availability of Microservices for Cyber-Physical Systems. ACPS, 9(1), 16–23. DOI: 10.23939/acps2024.01.016

[5] Raith, P., Nastic, S., & Dustdar, S. (2023). Serverless Edge Computing—Where We Are and What Lies Ahead. IEEE Internet Computing, 27(3), 50–64. DOI: 10.1109/MIC.2023.3260939

[6] Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. arXiv. DOI: 10.48550/arXiv.2304.03442

[7] Wu, Q., et al. (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv. DOI: 10.48550/arXiv.2308.08155

[8]Kaur, N., & Mittal, A. (2021). Fog Computing Serverless Architecture for Real-Time Unpredictable Traffic. IOP Conference Series: Materials Science and Engineering, 1022(1), 012026. DOI: 10.1088/1757-899X/1022/1/012026

[9] Aslanpour, M. S., Toosi, A. N., Cheema, M. A., Chhetri, M. B., & Salehi, M. A. (2024). Load balancing for heterogeneous serverless edge computing: A performance-driven and empirical approach. Future Generation Computer Systems, 154, 266–280. DOI: 10.1016/j.future.2024.01.020

[10] Liu, Z., Zhang, Y., Li, P., Liu, Y., & Yang, D. (2023). Dynamic LLM-Agent Network: An LLM-agent Collaboration Framework with Agent Team Optimization. arXiv. DOI: 10.48550/arXiv.2310.02170

[11] Al-Doghman, F., Moustafa, N., Khalil, I., Sohrabi, N., Tari, Z., & Zomaya, A. Y. (2023). AI-Enabled Secure Microservices in Edge Computing: Opportunities and Challenges. IEEE Transactions on Services Computing, 16(2), 1485–1504. DOI: 10.1109/TSC.2022.3155447

[12] Kallas, K., Zhang, H., Alur, R., Angel, S., & Liu, V. (2023). Executing Microservice Applications on Serverless, Correctly. Proceedings of the ACM on Programming Languages, 7. DOI: 10.1145/3571206

[13] Merlino, G., Tricomi, G., D'Agati, L., Benomar, Z., Longo, F., & Puliafito, A. (2024). FaaS for IoT: Evolving Serverless towards Deviceless in I/Oclouds. Future Generation Computer Systems, 154, 189–205. DOI: 10.1016/j.future.2023.12.029

[14] Jagutis, M., Russell, S., & Collier, R. (2023). Flexible simulation of traffic with microservices, agents & REST. International Journal of Parallel, Emergent and Distributed Systems, 38(6). DOI: 10.1080/17445760.2023.2242183

[15] Crawford, N., et al. (2024). BMW Agents—A Framework for Task Automation Through Multi-Agent Collaboration. DOI: 10.48550/ARXIV.2406.20041

[16] Liu, Z., Yu, H., Fan, G., & Chen, L. (2022). Reliability modelling and optimization for microservice-based cloud application using multi-agent system. IET Communications, 16(10). DOI: 10.1049/cmu2.12371

Oleh Chaplia was born in Lviv, Ukraine. He is a PhD student in the Specialized Computer Systems Department at Lviv Polytechnic National University, where he received his BSc and MSc degrees in Computer Engineering. Since earning his master's degree in 2015, he has worked in software engineering, specializing in designing and developing enterprise-grade cloud solutions with innovative technologies and high-quality architectures. His research interests include cloud computing, distributed systems, and artificial intelligence.

Halyna Klym doctor of technical sciences, professor, professor of the department of specialized computer systems of the Institute of Computer Technologies, Automation and Metrology of Lviv Polytechnic National University. In 2016, she received a Doctor of Science degree in Physical and Mathematical Sciences at Lviv Polytechnic National University. She soluconducts lecture courses on the design of ultra-large integrated circuits and methods and means of automated design of computer systems. She is an author of more than 170 scientific articles in international publications.

Edgars Elsts is a PhD in Physics, Senior scientist at the Institute of Solid State Physics, University of Latvia, one of the world's leading experts in the field of solid-state radiation physics, sensor materials for cyber-physical systems.