Generative AI for Optimizing Service Mapping in the Edge-Cloud Continuum

Loris Belcastro, Fabrizio Marozzo*, Aleandro Presta

* University of Calabria, Rende, Italy
*{lbelcastro, fmarozzo, aleandro.presta}@dimes.unical.it

***Abstract*—The increasing volume of data generated by Internet of Things (IoT) devices at the network edge introduces critical challenges for centralized cloud platforms, such as latency, bandwidth bottlenecks, and limited scalability. The edge-cloud continuum offers a distributed computing paradigm that mitigates these issues by allowing services to be deployed across multiple levels, from edge devices to centralized cloud servers. However, effectively composing and deploying applications in such environments remains a complex task due to platform heterogeneity, non-standardized integration, and strict application-level constraints. This paper presents a framework based on generative AI, which leverages Large Language Models (LLMs) for optimizing service mapping across the edge–cloud continuum. The approach starts with a high-level abstract application description, provided by the developer as a workflow of interconnected abstract elements. Developers also define relevant constraints, such as latency, power consumption, cost, data locality, and security, for each elements of the workflow. Using LLM prompting and a structured catalog of concrete services from multiple providers, the framework employs generative reasoning to automatically identify and map suitable cloud and edge services to each abstract component. This mapping aims to satisfy the defined functional and non-functional requirements, ensuring compliance with constraints and optimizing overall Quality of Service (QoS). The proposed solution abstracts away platform dependencies and supports deployment across heterogeneous edge and cloud infrastructures. Experimental evaluations demonstrate that LLM-assisted service mapping enables more flexible, performant, and cost-effective deployments, paving the way for scalable and intelligent application orchestration in distributed environments.**

***Index Terms*—Edge-cloud continuum, Service composition, Abstract design, Requirements analysis, Platform interoperability**

I. Introduction

The rapid proliferation of Internet of Things (IoT) devices is revolutionizing how data is processed and services are delivered, enabling real-time analytics and decision-making across domains such as smart cities, healthcare, and manufacturing. However, the immense volume of data generated at the network edge introduces critical challenges for traditional centralized cloud infrastructures, including high latency, bandwidth limitations, and scalability bottlenecks [3], [19]. In response, the edge-cloud continuum has emerged as a distributed computing paradigm that blends local edge resources with centralized cloud capabilities [6]. This model enables latency-sensitive processing at the edge, while offloading computationally intensive tasks to the cloud, offering better responsiveness, efficiency, and flexibility.

Despite its promise, deploying applications across the edge-cloud continuum remains a non-trivial task. Developers must cope with highly heterogeneous infrastructures, proprietary APIs, and a lack of standardized deployment models [8], [17]. Major cloud providers such as Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure offer edge solutions, but their non-interoperable ecosystems make cross-provider deployment complex and error-prone. Although open-source projects like OpenStack and OpenNebula aim to reduce fragmentation, seamless multi-tier orchestration is far from solved.

This paper introduces a generative AI-based framework designed to optimize service mapping within the edge–cloud continuum by utilizing Large Language Models (LLMs). The framework begins by accepting a high-level abstract description of an application from developers, expressed as a workflow comprising interconnected abstract elements. Additionally, developers specify constraints pertinent to each workflow element, including latency, power consumption, cost, data locality, and security. A key novelty of this approach lies in the use of LLMs to bridge the gap between abstract design and real-world deployment. Using structured prompts and curated service catalogs from multiple providers, the framework applies generative reasoning to translate each abstract service into a concrete implementation while ensuring compliance with all specified constraints. Once instantiated, these services are mapped to suitable execution layers (e.g., on-premise, far edge, or cloud) based on resource availability and optimization objectives.

We validate the effectiveness of our approach through a smart city scenario involving intelligent taxi dispatch in urban environments, aiming to optimize passenger pickup predictions through real-time analytics. By abstracting platform dependencies and automating complex architectural and operational decisions, our framework enables adaptive, constraint-aware application deployment across heterogeneous infrastructures, offering a scalable and intelligent alternative to traditional manual or heuristic-based strategies.

The remainder of the paper is structured as follows. Sec-

This work was supported by the research project "INSIDER: INtelligent ServIce Deployment for advanced cloud-Edge integRation" granted by the Italian Ministry of University and Research (MUR) within the PRIN 2022 program and European Union - Next Generation EU (grant n. 2022WWSCRR, CUP H53D23003670006).

tion II provides a brief review of related work in the domain of modeling frameworks for distributed applications. Section III discusses the reference edge–cloud continuum architecture and explores its implications for deploying distributed applications. Section IV introduces the proposed generative AI framework for modeling edge–cloud continuum applications. Section V presents the use case employed to evaluate the proposed modeling approach. Finally, Section VI summarizes our conclusions and outlines avenues for future research.

II. Related work

The exponential growth of IoT devices has intensified the demand for efficient and scalable data processing across the edge-cloud continuum. This architectural paradigm must overcome latency and bandwidth constraints while managing heterogeneous infrastructures. Key challenges include handling diverse computational capabilities, ensuring interoperability, and optimizing resource allocation across distributed layers. Recent research addresses these challenges by proposing frameworks and tools that support the dynamic and heterogeneous nature of edge-cloud systems. Notably, new methodologies enhance existing modeling capabilities by enabling adaptive service placement [9], real-time communication modeling, and QoS-aware orchestration. These capabilities are critical for deploying scalable, context-aware applications in complex domains such as smart cities, industrial IoT, autonomous vehicles, and UAV networks [4]–[7], [13], [16], [18]. Other studies provide comprehensive analyses of specific applications and enabling technologies, comparing paradigms for the edge-cloud continuum in terms of their features, advantages, and challenges [1], [12], [15], [20]. Designing robust, efficient applications across this continuum remains challenging, since developers face complex decisions about service decomposition, allocation, and orchestration across heterogeneous environments. Existing modeling frameworks support the design of applications for the continuum [2], [10], [21], but they offer limited mechanisms for modeling communication workflows, enforcing Quality of Service (QoS) constraints, or guiding optimal deployment strategies.

To address these limitations, we introduce a generative AI framework for optimized service mapping across the edge-cloud continuum. Our approach integrates the power of LLMs to assist in the dynamic generation, refinement, and validation of application models. LLMs are leveraged not only to interpret high-level developer intents and translate them into deployment configurations, but also to provide adaptive recommendations for service placement based on contextual requirements, resource constraints, and QoS targets. By incorporating natural language-based reasoning and pattern recognition, our framework supports a semantically aware and platform-agnostic design process that bridges the gap between conceptual modeling and executable deployment.

In contrast to existing solutions, our framework enables: (i) a clear separation of architectural and operational concerns; (ii) declarative specification of QoS and communication requirements; and (iii) automated, intelligent selection of suitable edge and cloud services. Furthermore, its built-in platform abstraction layer, coupled with generative AI capabilities, ensures better scalability, interoperability, and seamless integration across heterogeneous infrastructures, overcoming obstacles faced by current platform-specific approaches [9].

III. Edge-Cloud Continuum Architecture

The edge-cloud continuum architecture consists of several layers, each designed to optimize data processing and analysis based on proximity to the data source and the extent of processing required. Figure 1 illustrates the edge-cloud continuum architecture, where each layer plays a distinct role in data processing and analysis. These layers are classified based on the proximity of computational resources to the data source and the extent of processing performed at each level [11].

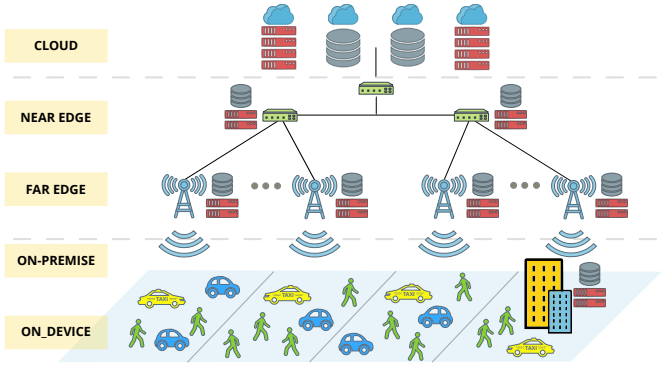

Fig. 1. Extended edge-cloud architecture

The *cloud* layer comprises centralized computing infrastructures located in remote data centers, offering on-demand resources such as virtual machines and storage over the Internet. This layer is ideal for intensive data processing due to its scalability and cost efficiency. Positioned closer to the cloud is the *near edge*, which consists of computing resources like mini data centers or regional clouds. These facilities act as intermediaries, enabling more immediate data processing and reducing latency for devices located several hundred kilometers away. Even closer to the devices is the *far edge*, where computing nodes are deployed near mobile phone towers or industrial facilities. This proximity ensures faster data aggregation and preliminary processing, enhancing the responsiveness of applications. The *on-premise* layer includes data processing nodes located within local facilities or at end-user sites, such as farms or stadiums. This setup provides a balance between proximity and autonomy, delivering fast and stable connectivity to business systems. Closest to the data source is the *on-device* layer, covering edge devices like IoT sensors and smartphones. These devices generate data and perform local processing, reducing latency and bandwidth usage, which is a critical feature for real-time applications like autonomous vehicles and smart industries. This distributed architecture enhances modern computing systems by addressing latency, privacy, and data localization requirements, enabling real-time responsiveness and scalability in applications.

Amazon Web Services (AWS) stands out among major cloud providers as offering the most advanced services for the edge-cloud continuum. This leadership is attributed to Amazon's extensive and highly distributed infrastructure, which enables seamless integration and support across all levels of the edge-cloud paradigm. AWS provides a comprehensive suite of tools and technologies, extending its computing capabilities from the central cloud to the edge with efficiency and scalability. For example, AWS's global network, spanning over 33 geographic regions, supports seamless deployment across the edge-cloud continuum, providing tailored solutions for every layer. At the near edge level, *Local Zones* position services closer to densely populated areas and major IT hubs, enhancing performance and reducing latency. Moreover, AWS offers various solutions to bring cloud services closer to data sources, such as *Wavelength*, which integrates computing resources at the far edge within 5G networks, and *Outposts* and *IoT Greengrass*, which enable seamless integration, local storage, and processing at the on-premise and on-device levels, respectively. While AWS leads in this space, other providers are also investing heavily in building continuous architectures for the edge-cloud, complemented by open-source initiatives like OpenStack and OpenNebula, which promote flexible and community-driven solutions for distributed computing.

IV. Generative AI Framework for Application Modeling in the Continuum

Building upon the conceptual framework introduced in previous work [9], [14], we have developed a functional software framework that enables platform-independent modeling, planning, and deployment of edge-cloud applications. This framework supports the full application lifecycle, from abstract workflow design to concrete deployment configuration, across heterogeneous infrastructures. The main modules and execution flow of the software framework are illustrated in Figure 2.

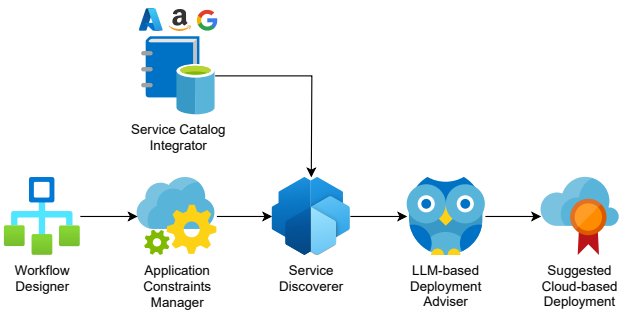

Fig. 2. Main modules and execution flow of the software prototype.

A. Abstract Workflow Designer

At the core of the system is the *Workflow Designer*, a visual web interface for composing application logic as interconnected abstract elements, representing devices, services, communication protocols, storage or tasks (e.g., data collection, transformation, or processing). Using the *Workflow Designer*, we can define an abstract workflow composed of the following types of elements:

- *Devices*: components that capture and generate data, such as GPS-equipped taxis, passenger smartphones, cloud-based and device-based traffic monitors, and web-based event monitors.
- *Computation*: components responsible for data processing and model inference, including services for real-time traffic monitoring, passenger request analysis, and taxi next-location prediction.
- *Networking*: infrastructure for communication between devices and processing elements, including cyber-physical interfaces, web APIs, publish-subscribe brokers, and streaming protocols.
- *Storage*: persistent and transient storage systems for raw data, contextual information (e.g., public events), and trained models.

These abstract components are annotated with metadata describing resource needs (latency, power, bandwidth), functional role (e.g., ETL, inference, integration), and semantic tags (e.g., IoT, analytics, machine learning), as illustrated in Table I. Notably, no assumptions are made about the specific deployment layers (cloud, edge, or on-device) or cloud providers, ensuring flexibility across infrastructures.

B. Application Constraints Manager

To accommodate deployment needs, the *Application Constraints Manager* enables users to specify both global and per-elements requirements, including latency, bandwidth, energy consumption, and persistence. These constraints guide all subsequent stages in the pipeline, since the abstract elements are later mapped to concrete implementations via a structured cloud service catalog, ensuring compliance with functional requirements and satisfaction of these constraints.

C. Service Catalog Integrator

The prototype integrates real-world service options through the *Service Catalog Integrator*, which organizes metadata-rich catalogs of cloud-native services from platforms such as AWS, Azure, and Google Cloud Platform (GCP). These catalogs contain both quantitative attributes (e.g., execution latency, bandwidth, and energy usage) and qualitative features (e.g., supported data formats, security levels). Notably, the service catalog explicitly represents the compatibility of each service with various layers of the edge-cloud continuum, which is essential for discovering suitable services according to deployment constraints and performance requirements.

D. Candidate Service Discovery

Following workflow definition, the system engages the *Service Discoverer*, a component which employs different strategies for identifying suitable cloud-native services based on the description of each abstract workflow element. In particular, this component supports mathematical optimization techniques, based on linear programming, and reasoning capabilities provided by large language models. Leveraging the

curated catalog of services from the *Service Catalog Integrator*, this component discovers eligible cloud-native services based on functional compatibility (e.g., analytics, messaging, storage), compliance with annotated QoS constraints (e.g., execution time, energy consumption), and semantic similarity with domain-specific tags (e.g., streaming, ETL, prediction, document storage). When an LLM is used, it is explicitly tasked through a structured prompt that defines the input and output format expected for each abstract service. A simplified version of the JSON-based prompt used to drive this task is shown below:

Role: Cloud architecture expert.

Task: Act as a cloud multi-cloud expert. I have a workflow composed of abstract services that I wish to map to concrete services from a given catalog. Each abstract service in the workflow (provided as JSON) includes {id: unique identifier, name: abstract service name, type: the type of the corresponding concrete service, description: to assist in service selection, parameters: constraints to guide selection}. Your tasks are as follows:

1) *Assign Layers:* Using the provided performance JSON, assign a field called *abstractservice_layer* to each abstract service by matching its parameters to those of a corresponding layer in the performance data.
2) *Service Matching:* From the provided catalog JSON, associate up to *n* compatible concrete services with each abstract service. Only select services that match the type field of the abstract service.

Ouput:: The output should be a plain JSON object (no code formatting) with a "result" field containing an array of elements, each with this fields: {*abstractservice_id*, *abstractservice_name*, *abstractservice_type*, *abstractservice_description*, *abstractservice_layer*, *services*: a list of up to *n* objects, each with *service_id*, *service_name*, *service_type*, *service_layers*}. Ensure accurate parameter-layer matching and strict type compatibility when selecting services.

The output of this module is a list of candidate concrete services that can be used to implement the different workflow element across various execution layers of the edge-cloud continuum. For example, local inference tasks can be executed directly on devices using AWS IoT Greengrass, while regional processing may be handled at the far edge using platforms like AWS Wavelength or Azure IoT Edge. Meanwhile, global tasks such as model training and data storage are delegated to cloud-native services, including Amazon SageMaker and Amazon S3.

E. LLM-Based Deployment Advising

To determine the optimal deployment configuration, a set of candidate services is provided to the *Deployment Adviser*, which leverages the reasoning capabilities of large language models. These models analyze the compatibility and feasibility of deploying the identified services, discovered earlier by the *Service Discoverer*, across heterogeneous cloud environments such as AWS, Azure, and GCP. The LLMs are guided by a structured prompt specifically designed to produce practical deployment insights, ensuring that the final configuration is both effective and adaptable to multi-cloud scenarios.

Role: Cloud architecture expert.

Task: You are a multi-cloud expert specializing in AWS, Azure, and Google Cloud Platform. Your job is to analyze a list of abstract services used in a workflow and determine the single best-fitting cloud service from the options provided for each service. For each abstract service in the discovered workflow, analyze its context and choose the best cloud service from the *cloud_service_catalog*. Choose based on functional fit, best practices, and deployment compatibility. Do not return any explanations, notes, or markdown. Output must be strictly valid JSON according to the given output structure.

Discovered Workflow: = [{

- *abstractservice_id*: intr - unique identifier of the service;
- *abstractservice_name*: string - name of the abstract service;
- *abstractservice_layer*: string - deployment layer (e.g., cloud, far edge, on-premise);
- *abstractservice_description*: string - description of the service's functionality;
- *abstractservice_metadata*: Array⟨string⟩ - characteristics and capabilities of the service:
- *cloud_service_catalog*: list of candidate services from different cloud provider = [{ *provider_name*: string - cloud provider name (AWS, Azure, GCP); *service_name*: string - name of the candidate cloud service implementation }]

}]

Output Structure: [{*abstractservice_id*: int, *abstractservice_name*: string, *abstractservice_layer*: string, *best_service*: string}]

Output Example: [{*abstractservice_id*: 0, *abstractservice_name*: 'Web Interface', *abstractservice_layer*: 'far edge', *best_service*: 'Amazon API Gateway'}, { *abstractservice_id*: 1, *abstractservice_name*: 'Publish-Subscribe Broker', *abstractservice_layer*: 'far edge', *best_service*: 'Amazon Simple Notification Service' }, ...]

In addition, LLMs can automatically generate infrastructure-as-code (IaC) artifacts and detailed service definitions tailored to the selected cloud provider, such as AWS CloudFormation templates or Terraform configurations. This capability streamlines and accelerates the deployment process by reducing manual effort, minimizing configuration errors, and ensuring alignment with provider-specific best practices.

V. Use Case: LLM-Assisted Deployment for Smart City Applications

We applied the proposed solution to a smart city scenario involving intelligent taxi dispatch in urban environments, aiming to optimize passenger pickup predictions through real-time analytics. The process began with the design of an abstract workflow using the *Workflow Designer* module, where devices, services, and communication protocols were modeled independently of the deployment infrastructure. After the abstract workflow was completed, we defined both constraints and metadata through the *Application Constraints Manager* for each elements. These included execution latency targets, energy efficiency preferences, and network usage limits, which guided the discovery and selection of suitable services.

Each component was described with metadata such as execution time, power consumption, and bandwidth needs, along with semantic tags (e.g., ETL, analytics, machine learning, IoT). The full list of workflow's elements, annotated with constraints and metadata, is shown in Table I.

As discussed above, the core of the deployment process was then handled by two modules: the *Service Discoverer* and the *Deployment Adviser*. The *Service Discoverer* performs an initial filtering of services from a multi-cloud catalog (AWS, Azure, GCP), matching abstract components to concrete candidates based on metadata compatibility and service

Workflow Element	Type	Constraints			Metadata
		Execution Time	Power Consumption	Network Bandwidth	
Taxi	D	very low	low	very low	device, mobile
Cyber-Physical Interface	C	moderate	moderate	moderate	messaging, integration
Device-based Traffic Monitor	D	very low	low	very low	IoT, sensing
Cloud-based Traffic Monitor	D	very low	low	very low	IoT, online sources
Event Monitor	D	very low	low	very low	IoT, public events
Passenger	D	very low	low	very low	IoT, end-user
Web Interface	C	moderate	moderate	moderate	internet access, UI
Publish-Subscriber Broker	C	moderate	moderate	moderate	pub-sub, messaging
Streaming Protocol	C	moderate	moderate	moderate	streaming, networking
Public Event Collector	P	high	high	very high	ETL, scraping
Real-Time Traffic Monitoring (Web)	P	high	high	very high	web sensors, analytics
Web Data Storage	S	high	high	very high	noSQL, document store
Real-Time Traffic Monitoring	P	moderate	moderate	moderate	road analytics, compute
Device Data Storage	S	high	high	very high	time series, IoT
Taxi Monitoring	P	moderate	moderate	moderate	ETL, preprocessing
Passenger Monitoring	P	moderate	moderate	moderate	ETL, preprocessing
Location Prediction	P	moderate	moderate	moderate	machine learning, prediction
ML Train	P	high	high	very high	training, model update
ML Storage	S	high	high	very high	model weights, persistence

TABLE I
APPLICATION CONSTRAINTS WITH SOME ANNOTATED TAGS FOR EACH COMPONENT. TYPE LEGEND: D = DEVICE, C = COMMUNICATION, P = PROCESSING/COMPUTATION, S = STORAGE. METADATA COLUMN LISTS FUNCTIONAL OR SEMANTIC TAGS ASSIGNED TO EACH COMPONENT.

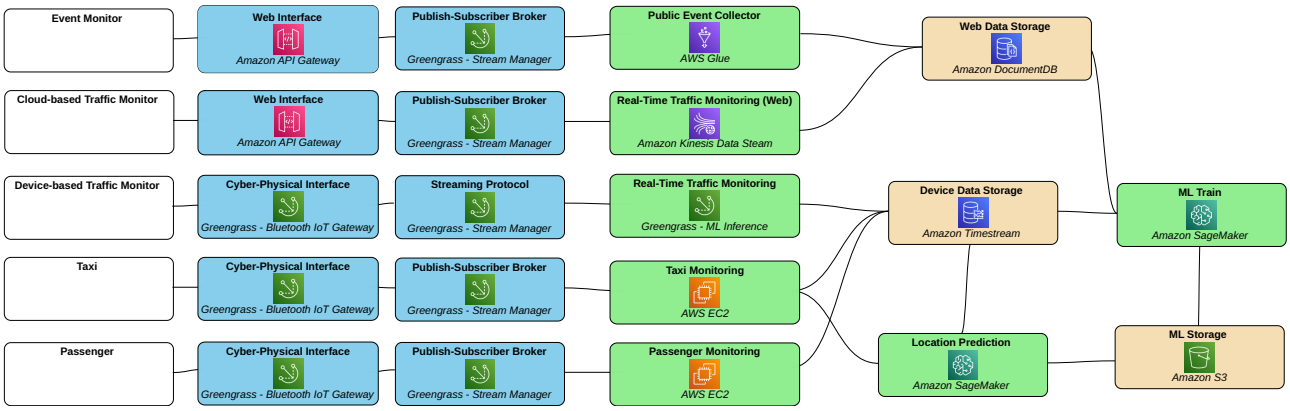

Fig. 3. An overview of the recommended deploy on AWS.

features. This reduced the search space to a short list of potential services for each workflow element. The *Deployment Adviser* then leveraged a large language model (*GPT-4o*) to select the most appropriate service from each shortlist. Using structured prompts, the LLM considered the context of each service (functional type, performance profile, constraints, tags) and recommended an optimal mapping that satisfied the given deployment criteria. The outcome of this process is the deployment configuration shown in Figure 3, where AWS was selected as the target platform. In particular, the LLM recommended concrete services that matched both functional and QoS requirements, suggesting the usage of several services at the different layers of the continuum. A few examples from the figure include:

- *Amazon Greengrass IoT Gateway* for executing machine learning inference on local taxi devices, ensuring low-latency, on-device prediction in federated mode.
- *Amazon Kinesis Data Stream* for real-time stream ingestion and messaging, matching the streaming and pub-sub components.
- *Amazon SageMaker* for centralized training and model update tasks, leveraging high compute availability in the cloud.
- *Amazon DocumentDB* and *Amazon S3* for transient and persistent storage respectively, balancing performance and cost across different layers.
- *Amazon API Gateway* to expose the web interface and REST endpoints, ensuring security and scalability.

This use case demonstrates the practicality and effectiveness of using LLMs to augment deployment planning across the edge-cloud continuum. By abstracting away infrastructure complexity and relying on AI-assisted decision making, developers can achieve fast, constraint-aware service mappings that are both flexible and optimized for heterogeneous envi-

ronments.

VI. Conclusion

This paper presented a generative AI-based modeling framework that enables platform-independent design and optimized deployment of applications across the edge–cloud continuum. By leveraging Large Language Models (LLMs) and structured service catalogs, the framework supports the complete application lifecycle, from abstract workflow modeling to concrete service mapping, while satisfying complex application-level constraints such as latency, energy consumption, and bandwidth. A key strength of the framework is its ability to separate application logic from the underlying infrastructure. This allows for flexible and constraint-aware service deployment across heterogeneous platforms. Its modular design includes components for abstract workflow modeling, constraint management, intelligent service discovery, and LLM-based deployment support. Together, these features provide a scalable and adaptable solution for managing distributed applications in dynamic multi-cloud environments.

The practical application in a smart city use case demonstrated the feasibility and benefits of LLM-assisted deployment planning, which can effectively simplify infrastructure decisions, enabling fast, flexible, and constraint-aware service mapping across diverse edge–cloud environments.

Future work will focus on enhancing the automation and adaptability of the framework. This includes integrating predictive machine learning models for proactive service selection, enabling dynamic adaptation in response to real-time context changes, and expanding support for hybrid multi-cloud environments. We also plan to enrich the framework with LLM-driven capabilities for generating infrastructure-as-code (IaC) artifacts, such as deployment scripts and configuration files compatible with Ansible, Terraform, or AWS CloudFormation, further streamlining the path from abstract modeling to executable deployment.

References

[1] Mohammad Aazam, Sherali Zeadally, and Khaled A Harras. Fog computing architecture, evaluation, and future research directions. *IEEE Communications Magazine*, 56(5):46–52, 2018.

[2] Daniel Balouek-Thomert, Pedro Silva, Kevin Fauvel, Alexandru Costan, Gabriel Antoniu, and Manish Parashar. Mdsc: modelling distributed stream processing across the edge-to-cloud continuum. In *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, UCC '21, New York, NY, USA, 2022. Association for Computing Machinery.

[3] Loris Belcastro, Riccardo Cantini, Fabrizio Marozzo, Alessio Orsino, Domenico Talia, and Paolo Trunfio. Programming big data analysis: Principles and solutions. *Journal of Big Data*, 9(4), 2022.

[4] Loris Belcastro, Fabrizio Marozzo, and Alessio Orsino. Hybrid edge/-cloud solutions for supporting autonomous vehicles. In *Advances in Autonomous Vehicle Systems*. River Publishers, 2023.

[5] Loris Belcastro, Fabrizio Marozzo, Alessio Orsino, Aleandro Presta, and Andrea Vinci. Developing cross-platform and fast-responsive applications on the edge-cloud continuum. In *15th IFIP Wireless and Mobile Networking Conference (WMNC 2024)*, pages 589–594, 2024.

[6] Loris Belcastro, Fabrizio Marozzo, Alessio Orsino, Domenico Talia, and Paolo Trunfio. Edge-cloud continuum solutions for urban mobility prediction and planning. *IEEE Access*, 11:38864–38874, 2023.

[7] Loris Belcastro, Fabrizio Marozzo, Alessio Orsino, Domenico Talia, and Paolo Trunfio. Using the compute continuum for data analysis: Edge-cloud integration for urban mobility. In *2023 31st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 338–344. IEEE, 2023.

[8] Loris Belcastro, Fabrizio Marozzo, Alessio Orsino, Domenico Talia, and Paolo Trunfio. Navigating the edge-cloud continuum: A state-of-practice survey. *arXiv preprint arXiv:2506.02003*, 2025.

[9] Loris Belcastro, Fabrizio Marozzo, Aleandro Presta, Rosa Varchera, and Andrea Vinci. Developing platform-agnostic iiot applications in edge-cloud environments. In *International Conference on Industry 4.0 & Smart Manufacturing 2024 (ISM 2024)*, 2024.

[10] Chuntao Ding, Ao Zhou, Yunxin Liu, Rong N Chang, Ching-Hsien Hsu, and Shangguang Wang. A cloud-edge collaboration framework for cognitive service. *IEEE Transactions on Cloud Computing*, 10(3):1489–1499, 2020.

[11] European Commission. European industrial technology roadmap for the next generation cloud-edge offering. Technical report, European Commission, May 2021.

[12] Panagiotis Gkonis, Anastasios Giannopoulos, Panagiotis Trakadas, Xavi Masip-Bruin, and Francesco D'Andria. A survey on iot-edge-cloud continuum systems: Status, challenges, use cases, and open issues. *Future Internet*, 15(12), 2023.

[13] Dragi Kimovski, Narges Mehran, Christopher Emanuel Kerth, and Radu Prodan. Mobility-aware iot application placement in the cloud–edge continuum. *IEEE Transactions on Services Computing*, 15(6):3358–3371, 2021.

[14] Fabrizio Marozzo and Andrea Vinci. Design of platform-independent iot applications in the edge-cloud continuum. In *20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, 2024.

[15] Sergio Moreschini, Fabiano Pecorelli, Xiaozhou Li, Sonia Naz, David Hästbacka, and Davide Taibi. Cloud continuum: The definition. *IEEE Access*, 10:131876–131886, 2022.

[16] Adrián Orive, Aitor Agirre, Hong-Linh Truong, Isabel Sarachaga, and Marga Marcos. Quality of service aware orchestration for cloud–edge continuum applications. *Sensors*, 22(5):1755, 2022.

[17] Daniel Rosendo, Alexandru Costan, Patrick Valduriez, and Gabriel Antoniu. Distributed intelligence on the edge-to-cloud continuum: A systematic literature review. *Journal of Parallel and Distributed Computing*, 2022.

[18] Shashank Shekhar and Aniruddha Gokhale. Dynamic resource management across cloud-edge resources for performance-sensitive applications. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 707–710. IEEE, 2017.

[19] Domenico Talia, Paolo Trunfio, Fabrizio Marozzo, Loris Belcastro, Riccardo Cantini, and Alessio Orsino. *Programming Big Data Applications: Scalable Tools and Frameworks for Your Needs*. World Scientific, 2024. ISBN: 978-1-80061-504-5.

[20] Jianyu Wang, Jianli Pan, Flavio Esposito, Prasad Calyam, Zhicheng Yang, and Prasant Mohapatra. Edge cloud offloading algorithms: Issues, methods, and perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–23, 2019.

[21] Xiaokang Wang, Laurence T Yang, Xia Xie, Jirong Jin, and M Jamal Deen. A cloud-edge computing framework for cyber-physical-social services. *IEEE Communications Magazine*, 55(11):80–85, 2017.