Developing Cross-Platform and Fast-Responsive Applications on the Edge-Cloud Continuum

Loris Belcastro*, Fabrizio Marozzo*, Alessio Orsino*, Aleandro Presta*, and Andrea Vinci†

* University of Calabria, Rende, Italy
† National Research Council of Italy, Institute for High Performance Computing and Networks, Rende, Italy
*{lbelcastro, fmarozzo, aorsino, apresta}@dimes.unical.it, †andrea.vinci@icar.cnr.it

***Abstract*—The large volume of data generated by Internet of Things (IoT) devices at the network edge poses significant challenges. Indeed, centralized cloud platforms struggle to manage this data flow due to issues such as latency, high bandwidth usage, and scalability problems. To address these challenges, the edge-cloud continuum has emerged as a powerful computing model that combines the strengths of edge and cloud computing, enabling data to be processed closer to its source while leveraging cloud resources for resource-intensive tasks. However, the wide adoption of edge-cloud environments is hindered by the lack of shared standards across different platforms and vendors, resulting in highly platform-specific applications. Although major cloud service providers are developing bridging and cloud services, the absence of common standards hinders communication and integration between platforms. This paper introduces a new modeling framework designed to enable developers to define applications for execution across the edge-cloud continuum in an abstract manner, independent of the deployment platform. The framework assists in identifying suitable deployment configurations and optimal service selection, ensuring that quality of service (QoS) requirements are met and supporting the development of flexible and scalable applications for seamless integration across the continuum. To demonstrate the effectiveness of this approach and how the framework facilitates the development of applications within the edge-cloud continuum, we present a use case in the domain of smart cities.**

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

I. Introduction

The proliferation of Internet of Things (IoT) devices has led to an unprecedented volume of data being generated at the edge of the network. This surge in data presents substantial challenges for traditional centralized cloud platforms, which are increasingly struggling to meet the demands for low latency, high bandwidth usage, and scalability [13]. As the number of IoT devices continues to grow, the limitations of these centralized systems are becoming more pronounced, highlighting issues related to data processing and transmission [4]. In response to these challenges, the edge-cloud continuum has emerged as a powerful computing model that leverages the strengths of both edge and cloud computing, enabling data processing to occur closer to its source while still utilizing the extensive computational resources of the cloud for more resource-intensive tasks [5]. By distributing data processing tasks between the edge and the cloud, this model aims to reduce latency, optimize bandwidth usage, and enhance scalability.

Despite its promise, the wide adoption of edge-cloud continuum infrastructures faces significant challenges. Major cloud service providers, such as Amazon, Google, and Microsoft, are actively investing in bridging the gap between edge and cloud by developing services that extend cloud capabilities to IoT devices, enabling local computing, data management, and analytics applications. However, the lack of shared standards across different platforms and vendors limits interoperability and hinders the seamless integration of services, resulting in the development of highly platform-specific applications. Indeed, each edge service integrates with its own set of cloud services, necessitating careful evaluation and selection from developers to meet application requirements, quality of service (QoS), and cost considerations [12].

This paper introduces a novel modeling framework designed to enable developers to define applications for execution across the edge-cloud continuum in an abstract manner, independent of the deployment platform. These applications are composed of abstract services that are agnostic to their deployment location (edge or cloud) or specific cloud providers (e.g., AWS, Google Cloud, Microsoft Azure), thus enabling the cross-platform development of applications in the compute continuum. Starting from abstract modeling, the framework facilitates the identification of suitable deployment configurations within specific cloud platforms by evaluating various concrete services that provide the requested functionalities. This capability to discern between different services enables the optimal selection of services for actual deployment.

We evaluate our modeling approach with a use case in the field of smart cities, which is one of the main areas where the edge-cloud continuum is used to support critical applications. In smart cities, vast amounts of data are generated by a multitude of interconnected devices, such as sensors, cameras, mobile devices, and smart meters, which monitor and manage urban infrastructure and services. This data must be processed

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) and by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

in real time to ensure efficient traffic management, energy distribution, waste management, public safety, and other critical functions. Traditional cloud platforms alone cannot meet the low latency and high bandwidth requirements for real-time processing of this data. Instead, by exploiting the continuum of resources from the cloud to the edge, smart cities can process data locally, reducing the latency and bandwidth demands on centralized cloud infrastructure. For instance, traffic cameras can analyze video feeds at the edge to detect accidents or congestion, enabling fast response actions. Similarly, energy consumption data from smart meters can be processed locally to optimize energy distribution in real-time, reducing wastage and improving efficiency. Furthermore, the edge-cloud continuum enables smart cities to dynamically scale data processing, with the cloud providing additional computational resources during peak times, such as major public events, to maintain consistent performance and reliability.

Particularly, in this paper we focus on a specific use case in smart cities where geotagged data, generated by the movement of passengers and taxis at the network edge, is processed and analyzed within the compute continuum. This real-time analysis, using machine learning algorithms, offers solutions to various daily life problems, including predicting areas where taxis are more likely to find passengers. We design an abstract client-server configuration for the application to run across the edge-cloud continuum, independent of where the processing occurs (on devices, edge, or cloud). We will subsequently show how this abstract design can be implemented on Amazon Web Service (AWS) using its services, highlighting how platform-agnostic abstract modeling facilitates the development of practical applications within the compute continuum.

The remainder of the paper is structured as follows. Section II provides a brief review of related works in the domain of modeling frameworks and smart cities. Section III describes an edge-cloud continuum architecture and explores its implications for deploying applications. Section IV presents the proposed framework for modeling edge-cloud continuum applications. Section V presents the use case used to evaluate the proposed modeling approach. Finally, Section VI presents our conclusions and outlines avenues for future research.

II. Related work

Several studies have focused on defining modeling frameworks for specific and generic application scenarios, each addressing various challenges in the development of edge-cloud applications.

The ITEMa approach [8] introduces a visual notation system aimed at modeling IoT-based smart ecosystems, emphasizing the design of edge systems and the locations where components should be executed. The EdgeFlow IoT framework [3] assists in developing latency-sensitive IoT applications by extending the flow-based programming paradigm. It captures latency and timing requirements and includes a resource allocation technique for deploying and validating constrained IoT applications. However, it does not account for other considerations such as data locality or deployment costs. IADev [18] employs attribute-driven design and model-driven development for IoT applications, supporting cloud-centric development but not specifically targeting edge-cloud continuum applications. In contrast to these works, our approach aims to provide a versatile tool for designing and developing edge-cloud applications, facilitating concern separation, QoS requirement identification, and optimal provider/service selection for deployment.

In the domain of smart cities, urban mobility data can be utilized in various ways to enhance citizens' quality of life while improving cities' efficiency and sustainability [16]. In particular, predicting vehicular trajectories has garnered significant interest due to its potential applications in intelligent transportation systems [11], traffic monitoring [21], and transportation management [6]. This work focuses on the problem of mobility prediction for taxis using geotagged data. The aim is to predict taxi locations with high passenger demand, potentially reducing route costs and traffic congestion. Unlike fixed-route public transportation, taxis plan routes after passenger drop-off [22], and real-time location monitoring via GPS trackers enables trajectory analysis for predicting the next taxi locations [20]. Existing methods include frequent pattern mining and association rules [2], [7], stochastic models using Markov chain processes [19], and predominantly machine learning techniques, which have been extensively employed to address the challenges associated with accurately predicting vehicle movements. In particular, deep learning methods, specifically recurrent neural networks (RNNs) and their variants like Long Short-Term Memory (LSTM) networks, have shown substantial promise in vehicular trajectory prediction [15]. Altché and Fortelle [1] demonstrated the effectiveness of LSTMs in capturing temporal dependencies in vehicle trajectories. Similarly, Park et al. [17] proposed a novel LSTM-based approach incorporating spatiotemporal features to improve prediction accuracy in complex driving scenarios. Convolutional Neural Networks (CNNs) have also been explored to represent traffic scenes and predict vehicle trajectories [9]. Moreover, hybrid models combining CNNs and RNNs, as discussed by Deo and Trivedi [10], leverage the strengths of both architectures to enhance prediction performance.

III. Edge-Cloud Continuum Architecture

The edge-cloud continuum architecture comprises layered computing resources, each positioned to optimize data processing and analysis efficiency based on proximity to data sources:

- *Cloud*: Centralized data centers providing scalable, cost-efficient, compute-intensive processing and storage over the internet.
- *Near edge*: Smaller public data centers located closer than traditional edge, facilitating timely data processing; may include mini data centers or regional clouds.
- *Far edge*: Nodes near devices, like mobile towers, for rapid data aggregation and preliminary processing.

- *On-premise*: Private edge nodes in local facilities (e.g., stadiums), offering close, stable connectivity and real-time data processing.
- *On-device*: IoT devices and other edge hardware performing local processing to minimize latency and bandwidth, essential in real-time applications (e.g., autonomous vehicles).

This distributed setup enhances modern computing by addressing latency, privacy, and data locality needs. Currently, *Amazon Web Services* (AWS) leads in supporting this architecture, offering tailored services across layers: *AWS Local Zones* at the *near edge*, *AWS Wavelength* for 5G-integrated *far edge* processing, and AWS Outposts and IoT Greengrass for *on-premise* and *on-device* operations.

IV. Framework for Modeling Edge-Cloud Applications

The proposed framework employs a platform-independent and cross-platform approach by abstracting application modeling for the edge-cloud continuum from the actual deployment [14]. This is achieved by defining an application as a workflow of abstract services, decoupled from the deployment layer (e.g., cloud or edge) and independent of specific services from cloud providers like AWS or Microsoft Azure, or open-source cloud platforms such as OpenStack or OpenNebula. These abstract services, which may include components like data collection, transformation, and processing, are then mapped to a catalog of services offered by a given cloud service provider. This mapping provides guidance on how these tasks can be implemented using actual services while adhering to functional constraints. The main components of the proposed framework are outlined in Figure 1 and described below:

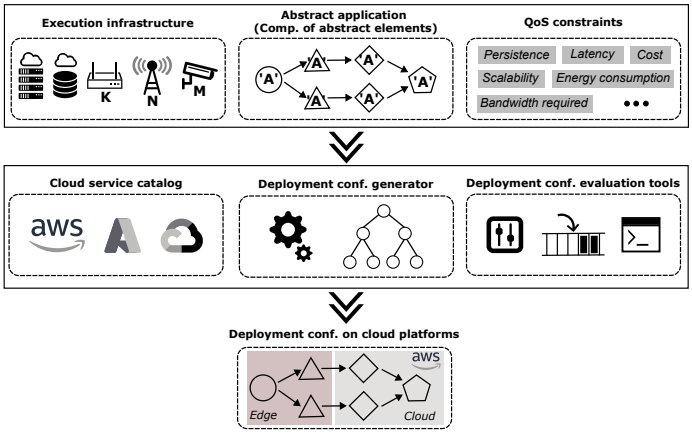

Fig. 1. An overview of the proposed modeling framework.

- **Execution infrastructure**: this component models the execution infrastructures for both cloud and edge, taking into account the different technologies, hardware types, and network topology through which they can be realized. For instance, the virtual machine sizes, the number of devices, the on-premises resources, and the available execution layers (cloud, far edge, near edge, on-premises, or on-device) are used to define a prototype to model a wide range of cloud/edge technologies and infrastructure.
- **Abstract application**: this component models the application as a composition of abstract services with associated tasks and dependencies, modeling both functional and non-functional requirements among services. Services can run on different execution layers and may be adapted, partitioned, optimized, or compressed.
- **QoS constraints**: this component defines the constraints that the application should meet, such as latency, bandwidth usage, operational cost, the persistence of data storage, scalability, energy consumption, and others.
- **Cloud service catalog**: this includes service catalogs from various cloud providers (e.g., AWS, Microsoft Azure, and Google Cloud), that are mapped to the abstract services defined in the previous step to meet application requirements. It contains both quantitative (e.g., latency, energy consumption) and qualitative (e.g., persistence, supported data formats, suitable execution layers) information about each service, which helps in matching abstract to actual services.
- **Deployment configuration generator**: this component builds deployment configurations for the abstract application, given the desired QoS constraints, the real execution infrastructure, and the actual cloud service catalog. It generates valid deployment configurations that consider the distribution of workloads across the edge-cloud continuum architecture.
- **Deployment configuration evaluattion tools**: this component evaluates application deployment configurations by analyzing performance parameters and requirements. Simulation tools or linear programming systems can be used to automate the evaluation.
- **Deployment configurations on cloud platforms**: after identifying suitable configurations, this component assists in deploying the selected configuration on the chosen cloud platform with its services.

V. Use Case: Analysis of Geotagged Data for Enhancing Urban Mobility in Smart Cities

To evaluate the effectiveness of the modeling approach discussed in Section IV, we present a use case in the smart city domain, which leverages machine learning techniques to tackle real-world challenges in urban transportation, specifically focusing on predicting areas with high passenger demand to optimize taxi dispatch.

We propose an architecture that exploits the distributed computing resources along both the edge and cloud. Specifically, GPS-equipped devices installed in taxis capture real-time taxi positions (i.e., GPS coordinates) and other relevant information, including the taxi identifier, speed, occupancy status (i.e., whether the taxi is occupied or not), and previous trips made by the taxi. This data is preprocessed through cleaning, filtering, and transformation methods, in order to be subsequently given as input to a machine learning model

specifically trained to predict high-demand areas for efficient taxi dispatch. Within this workflow, there are two potential approaches for prediction using machine learning models:

- The taxi sends all preprocessed data to the cloud, which uses a global model to make predictions and sends the results back to the taxi.
- Following a typical federated learning approach, the taxi may have a local machine learning model trained on its own trip data, which it can use to make low-latency and privacy-preserving predictions. In this case, processed data at the edge is used directly for predictions from the local model, and only model updates are sent to the cloud.

Additionally, the system should be able to store data collected during taxi operations (e.g., GPS trajectories, passenger pick-up and drop-off points) and exceptional events such as public events, to improve the prediction model during subsequent retraining phases.

Given the application scenario described above and following the modeling approach defined in Section IV, we first define an abstract workflow, shown in Figure 2, whose nodes can be of the following types:

- **Device**: these are components that capture and generate data. Examples for this use case include GPS-equipped taxis, traffic and event monitors, and customer devices.
- **Networking**: these elements handle communication and data transfer between physical devices and computational components. Examples include cyber-physical and web interfaces, publish-subscribe brokers, and communication protocols.
- **Computation**: these elements process, analyze, and manage data to derive insights and predictions. Examples include real-time monitoring and machine learning models for location prediction.
- **Storage**: these components store data for both real-time processing and long-term analysis. Examples include persistent storage for historical data and temporary storage for real-time data.

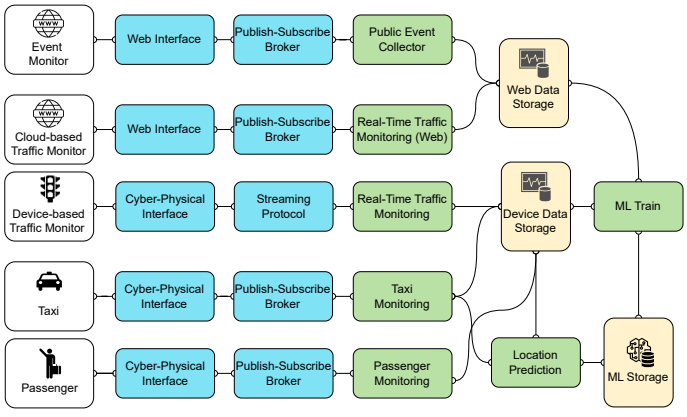

Fig. 2. Abstract workflow of a use case related to taxi next location prediction in smart cities using the proposed modeling approach. White nodes are the *device* elements, light blue nodes are *networking* elements, green nodes are *computation* elements, and yellow nodes are *storage* elements.

Particularly, the devices included in the proposed workflow are the *GPS-equipped taxis*, *passenger devices*, *traffic monitors*, and *event monitors*. Taxis provide real-time location data, which is used for tracking their movements and feeding the location prediction model. Passenger devices, such as smartphones, send request data that includes their current location and desired destination, helping match taxis to passengers. Traffic monitors can be *device-based* and *cloud-based*. Device-based monitors capture real-time traffic conditions using road sensors, while cloud-based monitors gather information from various online sources. Both types of monitors influence taxi availability and routing by ensuring that predictions incorporate current road conditions. Another important condition to consider is the presence of public events, such as soccer matches, concerts, and shows, which can increase traffic congestion or cause a surge in customer requests. To monitor these events, we include a component named *event monitor*, which uses information from the internet.

The networking components ensure communication and data transfer between the devices and the computational components. The *cyber-physical interfaces* connect physical devices, like taxis and passenger devices, to the network, enabling data exchange. Communication is managed by the *publish-subscribe broker*, which ensures that data from taxis, passengers, and monitors is correctly routed to the appropriate computational elements. *Streaming protocols* facilitate real-time data transmission, supporting low-latency interactions and fast updates across the network. In addition to the cyber-physical interfaces, some virtual devices, such as the event monitor and the cloud-based traffic monitor, use *web interfaces* to retrieve data from the internet.

The computation components process, analyze, and manage the collected data. The *real-time traffic monitoring* component plays a critical role in monitoring traffic and congestion conditions on the road, allowing the system to adapt dynamically. Similarly, the *public event collector* scrapes information about public events from the web and stores it to combine these data with those from the taxis for model learning. The *taxi monitoring* and the *passenger monitoring* components collect data from taxis and passengers respectively and are responsible for cleaning and transforming the raw data into a usable format for the predictive model, filtering out noise, and normalizing data formats. The processed data is then used to determine the optimal locations for taxis to find new passengers, which is the core functionality of the prediction system. This is achieved by the *location prediction* component, which predicts the best locations for taxi pick-ups based on the preprocessed data of the taxi and external information. The data can also be used by the *ML train* component to regularly update the machine learning model, ensuring that the predictions remain accurate over time by incorporating feedback and new patterns detected in the data. The periodic retraining is managed by balancing the need for up-to-date models with the system's real-time performance requirements.

Finally, the storage layer supports both real-time processing and long-term analysis. The *ML storage* component stores

the machine learning model (e.g., the weights of the neural network), ensuring that the models are accessible for both re-training and real-time predictions. The *data storage* component keeps event data, such as traffic conditions, events, and passenger requests, available for real-time decision-making and historical analysis, which helps in refining and improving future predictions. In particular, we include a *device data storage* for data produced by taxis, passengers, and road traffic monitors, and a *web data storage* to persistently store data from events and cloud-based traffic monitors.

It is worth highlighting that our modeling approach relies only on a basic client-server architecture, without explicitly specifying where the computation occurs—whether on devices, intermediate levels (e.g., on-premise, near edge, and far edge) or in the cloud. Moreover, our approach is cloud-agnostic, enabling the cross-platform deployment of abstract services. As an example, in the following section, we describe how the workflow and its abstract services can be implemented on the AWS cloud platform.

Given the above abstract model, we can also define some requirements and QoS metrics for each task and the overall abstract services that need to be met in the concrete deployment. For example, physical devices are highly dependent on low latency for real-time data transmission to ensure timely and accurate location updates and traffic information. Additionally, these devices must minimize energy consumption to extend their operational lifespan and reduce overall energy costs. Networking components, on the other hand, require high bandwidth to handle continuous data streams from multiple sources, ensuring efficient communication and data transfer. Low latency is also crucial for real-time interactions, and scalability is essential to manage the increasing data volumes from a growing number of devices (e.g., passengers). Computation components must operate with low latency to provide fast-responsive processing and predictions. These components need scalable computational resources to handle dynamic loads and increasing data volumes. Energy-efficient processing is also crucial to reduce energy consumption, particularly for real-time event monitoring and model prediction. Finally, storage components require reliable persistence for both historical data and machine learning models to ensure long-term analysis and consistent model retraining. These components need to be scalable to handle growing data storage needs without performance degradation.

A. Implementation of the use case

AWS offers a complete suite of cloud services, ranging from basic services such as computing, storage, and networking to advanced solutions like machine learning and deep learning services, ensuring integration and deployment across diverse environments. Furthermore, AWS is likely the cloud provider that has invested the most in adapting its infrastructure to specifically meet the requirements of the edge-cloud continuum. To bring cloud services to the end-user device level, AWS offers tools and services for managing physical devices such as actuators, cameras, and sensors. In particular, *AWS IoT Greengrass* extends the capabilities of the cloud to local devices, allowing them to collect and analyze data closer to the source, respond autonomously to local events, and communicate securely via local networks. Additionally, on-premises devices can securely communicate with *AWS IoT Core* and export IoT data to the AWS Cloud. AWS IoT Greengrass also supports the FaaS (*Function-as-a-Service*) paradigm: developers can define *AWS Lambda* functions in the cloud and, through connectors, deploy and execute these functions locally on fleets of physical devices in a serverless manner. *AWS Amplify* is a set of tools and services that simplifies the development of mobile apps with shared backend services and features, such as authentication, data storage, and real-time updates, which are accessible across various device platforms including web, iOS, and Android.

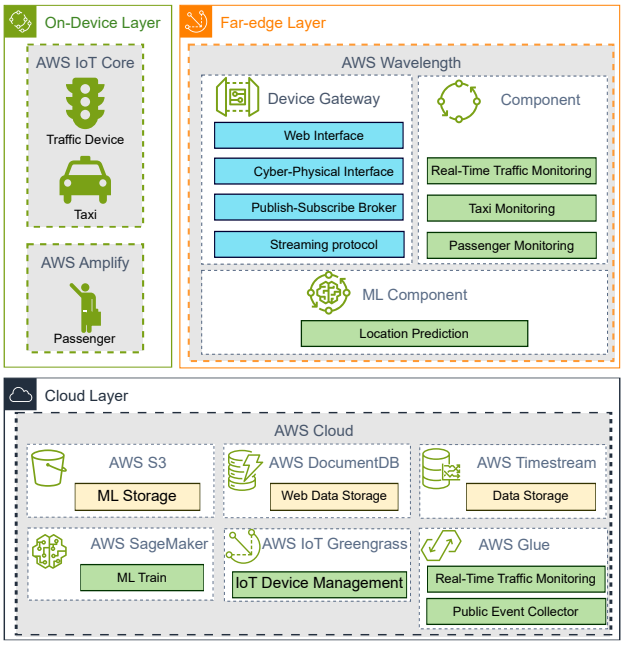

Fig. 3. Deployment of the abstract workflow on AWS.

According to the flow defined by the modeling framework proposed in this work, after defining the execution infrastructure, the abstract application, and the QoS constraints, we can exploit the information contained in the service catalog to identify optimal configurations for deploying the application on a specific cloud platform, also exploiting linear optimization techniques or information coming from simulations. Using this approach and leveraging the services provided by AWS, we obtained the suitable configuration illustrated in Figure V, which deploys the application across three distinct levels of the edge-cloud continuum (i.e., device, far edge, and cloud).

At the device level, we manage traffic monitoring devices (e.g., cameras and traffic lights with sensors) and taxis. These devices collect traffic data crucial for real-time monitoring and decision-making. We assumed these devices are equipped with specific IoT hardware, like Raspberry Pi, which features low energy consumption, sensors for data collection, and

communication interfaces for data transmission to servers for processing. We use *AWS IoT Core*, a managed cloud service, to enable secure and reliable interaction between these connected devices and cloud applications. Device management is facilitated by *AWS IoT Greengrass*, which provides a local device gateway and allows AWS Lambda functions to be executed locally for data preprocessing and aggregation before sending the data to the nearest server located at the far edge layer. We used the AWS Wavelength service to bring AWS computing resources closer to the data sources, enabling the deployment of next-location prediction models that are optimized for the local context of the devices at the far edge level. *AWS Wavelength* is a managed service that brings AWS compute and storage services to the edge of telecommunications networks, by integrating AWS infrastructure directly into data centers of telecommunication providers. In the cloud, we leverage Amazon S3 to store models trained using *AWS SageMaker*. These models are trained on data related to specific geographic areas and are periodically deployed on associated nodes of the AWS Wavelength architecture. Additionally, we utilize *AWS Timestream* for storing data coming at regular intervals from devices and *AWS DocumentDB*, a document database service that is compatible with MongoDB, for storing traffic/event data from web sources as JSON-like documents, gathered using *AWS Glue ETL* jobs. It is worth noticing that this represents just one possible combination of AWS services aligned with our abstract model, as there exist alternative services that could be selected based on specific project requirements.

VI. Conclusion

This paper presents a novel modeling framework that enables platform-independent application development within the edge-cloud continuum by defining abstract services as workflows of computation, networking, and storage elements aligned with specific constraints and QoS metrics. This abstraction enhances flexibility and scalability, supporting optimal service selection across cloud infrastructures and enabling efficient platform and service choices within the continuum.

Future work will refine the framework by creating tools to define abstract service requirements and QoS criteria, aiming for automated selection of optimal services and configurations via optimization or heuristic algorithms. Machine learning will further enhance service selection by predicting performance based on historical data, while a dynamic adaptation mechanism will monitor and respond in real-time to fluctuations in service availability, workload, and user demands, ensuring QoS standards are consistently met.

References

[1] Florent Altché and Arnaud de La Fortelle. An lstm network for highway trajectory prediction. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*, pages 353–359. IEEE, 2017.

[2] A. Altomare, E. Cesario, C. Comito, F. Marozzo, and D. Talia. Trajectory pattern mining for urban computing in the cloud. *IEEE Transactions on Parallel and Distributed Systems*, 28(2):586–599, 2017.

[3] Cosmin Avasalcai, Bahram Zarrin, and Schahram Dustdar. Edgeflow—developing and deploying latency-sensitive iot edge applications. *IEEE Internet of Things Journal*, 9(5):3877–3888, 2021.

[4] 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.

[5] 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.

[6] Francesco Branda, Fabrizio Marozzo, and Domenico Talia. Ticket sales prediction and dynamic pricing strategies in public transport. *Big Data and Cognitive Computing*, 4(4), 2020. ISSN: 2504-2289.

[7] Eugenio Cesario, Fabrizio Marozzo, Domenico Talia, and Paolo Trunfio. Sma4td: A social media analysis methodology for trajectory discovery in large-scale events. *Online Social Networks and Media*, 3-4:49–62, 2017.

[8] Franco Cicirelli, Antonio Guerrieri, Alessandro Mercuri, Giandomenico Spezzano, and Andrea Vinci. Itema: A methodological approach for cognitive edge computing iot ecosystems. *Future Gener. Comput. Syst.*, 92:189–197, 2019.

[9] Zhiyong Cui, Ruimin Ke, Ziyuan Pu, and Yinhai Wang. Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction. *arXiv preprint arXiv:1801.02143*, 2018.

[10] Nachiket Deo and Mohan M Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *IEEE conf. on computer vision and pattern recognition workshops*, pages 1468–1476, 2018.

[11] George Dimitrakopoulos and Panagiotis Demestichas. Intelligent transportation systems. *IEEE Veh. Technology Magazine*, 5(1):77–84, 2010.

[12] Hesham El-Sayed, Abdelhamid Mellouk, Laurent George, and Sherali Zeadally. Quality of service models for heterogeneous networks: overview and challenges. *annals of telecommunications-annales des télécommunications*, 63:639–668, 2008.

[13] Fabrizio Marozzo, Alessio Orsino, Domenico Talia, and Paolo Trunfio. Edge computing solutions for distributed machine learning. In *IEEE International Conference on Cloud and Big Data Computing (CBDCom)*, pages 1–8, 2022.

[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] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 637–646, 2009.

[16] Sara Paiva, Mohd Abdul Ahad, Gautami Tripathi, Noushaba Feroz, and Gabriella Casalino. Enabling technologies for urban smart mobility: Recent trends, opportunities and challenges. *Sensors*, 21(6):2143, 2021.

[17] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 1672–1678. IEEE, 2018.

[18] Wajid Rafique, Xuan Zhao, Shui Yu, Ibrar Yaqoob, Muhammad Imran, and Wanchun Dou. An application development framework for internet-of-things service orchestration. *IEEE Internet of Things Journal*, 7(5):4543–4556, 2020.

[19] Punit Rathore, Dheeraj Kumar, Sutharshan Rajasegarar, Marimuthu Palaniswami, and James C Bezdek. A scalable framework for trajectory prediction. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3860–3874, 2019.

[20] Alberto Rossi, Gianni Barlacchi, Monica Bianchini, and Bruno Lepri. Modelling taxi drivers' behaviour for the next destination prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(7):2980–2989, 2019.

[21] Nam Tang, Cuong Do, Tien Ba Dinh, and Thang Ba Dinh. Urban traffic monitoring system. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence: 7th International Conference, ICIC 2011, Zhengzhou, China, August 11-14, 2011, Revised Selected Papers 7*, pages 573–580. Springer, 2012.

[22] Jing Yuan, Yu Zheng, Liuhang Zhang, XIng Xie, and Guangzhong Sun. Where to find my next passenger. In *13th Int. Conference on Ubiquitous Computing*, UbiComp '11, pages 109–118, New York, NY, USA, 2011. ACM.