Design of Platform-Independent IoT Applications in the Edge-Cloud Continuum

Fabrizio Marozzo
DIMES Department, University of Calabria
Rende, Italy
fmarozzo@dimes.unical.it

Andrea Vinci
ICAR-CNR National Research Council of Italy
Rende, Italy
andrea.vinci@icar.cnr.it

***Abstract*—In the dynamic edge cloud continuum, where platforms and vendors do not have shared standards, application development often depends on the initial platform choice. Furthermore, applications designed and deployed to leverage the cloud for data analytics and storage, with devices as data generators, face challenges in adapting to intermediate levels of the edge-cloud continuum (such as on-premise, near and far-edge). This paper presents a novel approach that uses a set of abstractions to conceptualize platform-independent applications. Applications are modeled as a composition of abstract services, which are independent of the place of distribution, be it edge or cloud, and specific platforms, and offer developers flexibility in the selection of services and providers during distribution, optimizing the use of resources and costs. Through detailed modeling, this paper uncovers how applications interact with abstract services, structured as workflows comprising interconnected tasks encompassing computation, communication, and storage. The interaction between these elements has a significant impact on both quality of service (QoS) and cost considerations, a key consideration in the design phase. Using such abstractions in modeling not only facilitates application development but also simplifies application deployment processes. A case study within the Industrial Internet of Things (IIoT) validates the modeling approach, illustrating how precise and abstract modeling of the application, independent of specific platforms, facilitates the development of real-world applications within the edge-cloud continuum.**

***Index Terms*—Edge Cloud Continuum, Service Composition, Abstract Design, Deployment Agnosticism, Requirements Analysis**

I. Introduction

In recent years, the edge-cloud continuum computing has emerged as a revolutionary paradigm, combining the strengths of edge computing and cloud computing. This approach capitalizes on the vast resource capacity offered by the cloud while harnessing local communication and computation capabilities at the edge. It has been successfully tested in various application domains such as smart cities, cognitive building [2], and Industry 4.0 [15], highlighting its capability of managing the distributed and collective nature of Internet of Things (IoT) applications, thus garnering significant interest from both academic and industrial communities.

However, in the dynamic edge-cloud continuum characterized by a lack of shared standards among platforms and vendors, application development heavily relies on the initial platform choice. Moreover, applications designed to utilize the cloud for data analytics and storage, with devices serving as data generators, face challenges in adapting to intermediate levels of the edge-cloud continuum (such as on-premise, near, and far-edge). These challenges are particularly evident in scenarios involving big data and machine learning. The complexities of processing and analyzing large volumes of data in real-time [4], coupled with the demand for sophisticated machine learning algorithms , emphasize the critical need for seamless integration across the continuum. Factors such as latency, bandwidth constraints, and computational resource availability further complicate the distribution of data processing tasks between edge and cloud environments [5].

The emergence of mature IoT technologies has led to the release of various edge computing platforms from major cloud vendors such as AWS Greengrass and Google Cloud IoT Edge, each lacking a shared standard. As a result, the design of applications across the edge-cloud continuum often depends on a specific target platform or infrastructure choice. Additionally, each edge platform comes with its own cloud service ecosystem, necessitating evaluation and selection. This constrains designers and developers, limiting flexibility and requiring an early commitment to a chosen ecosystem. Even after selecting a provider, careful consideration of services is required to align with application needs, including requirements, QoS, and cost factors. This applies throughout deployment and operation, where tasks must be allocated across heterogeneous nodes while meeting QoS requirements and minimizing costs [9].

This paper introduces a novel approach leveraging abstraction to conceptualize platform-independent applications, which are composed of abstract services independent of deployment location (edge or cloud) and specific platforms, whether public (e.g., AWS, Microsoft Azure) or private (e.g., OpenStack). The proposed methodology not only spans across diverse cloud infrastructures but also within a singular infrastructure, recognizing the diverse concrete services implementing similar functionalities, each with unique characteristics that necessitate careful consideration. This supports optimal service

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").

selection during deployment. Our conceptualization unveils the inherent interactions among abstract services within an application, illustrating their interdependencies and interactions. Furthermore, it elucidates how various tasks within an application, such as data collection and model training, activate concrete services and utilize them in specific ways. This contributes to harnessing the full potential of computation in the edge-cloud continuum.

To evaluate our proposed modeling approach, we present a case study in the Industrial Internet of Things (IIoT). This case study explores the dynamic edge-cloud continuum architecture within IIoT, demonstrating how machine learning models deployed at the edge address real-world industrial challenges. We focus on automating the sorting of various plastics in a recycling plant, traditionally a labor-intensive and error-prone task. By leveraging machine learning for rapid image processing, we offer a promising solution to automate classification. Initially, we illustrate a platform-independent design for the application within the edge-cloud continuum using a simple client-server architecture, lacking specifications regarding the execution distribution whether on devices, edge, or cloud. Subsequently, we demonstrate how this abstract design can be implemented on a tangible platform such as AWS, leveraging its available services across the continuum. Through this case study, we illustrate how precise and abstract modeling of the application, independent of specific platforms, facilitates the development of real-world applications within the edge-cloud continuum.

The remainder of the paper is structured as follows. Section II offers a concise review of related works. Section III outlines the reference edge-cloud continuum architecture under consideration, and section IV presents the key abstractions for modeling edge-cloud applications. In Section V, we present a case study showcasing the application of these abstractions in designing a solution for managing plastic sorting in a recycling facility. Finally, Section VI presents our conclusions and outlines avenues for future research.

II. Related work

Regarding the analysis of the state of the art in the edge-cloud continuum, several surveys shed light on this evolving concept. One notable work is by Moreshini et al. [14], where they emphasize the growing attention and adoption of the cloud continuum concept among practitioners, academics, and funding agencies. However, they highlight the ambiguity surrounding its definition, with various studies offering contrasting descriptions. To address this challenge, the authors conducted a systematic mapping study of the literature to understand the concept's evolution and provide a comprehensive definition. Their work culminated in a comprehensive definition that brings together common aspects of the cloud continuum, giving practitioners and researchers a clearer understanding of this emerging paradigm.

Kong et al. [10] delve into the transformative role of edge computing in the IoT landscape, addressing the challenges posed by conventional cloud-centric approaches. They highlight the promise of edge computing to decentralize data processing, thereby alleviating cloud loads and reducing network latency. Emphasizing the need for systematic exploration, they analyze the impact of edge computing on the evolution of IoT and classify recent advances, ranging from fundamental principles to practical applications. Their comprehensive survey identifies emerging trends, highlights the need for continued research, and outlines avenues for future exploration, offering valuable insights into the dynamic intersection of edge computing and IoT.

Chang et al.'s survey [6] explores the evolution of telecommunications technologies, highlighting the inadequacies of traditional cloud network architectures amid growing demands. They advocate a shift toward edge computing, placing network functions closer to end users to mitigate backhaul burdens and latency issues. This transition introduces complexity in service orchestration and resource management at the edge. Their comprehensive review includes the architectures, benefits, and standardization efforts of edge computing, followed by a detailed exploration of cutting-edge techniques in service orchestration and resource management. Addressing the challenges of open research, their survey offers valuable insights to drive progress in the dynamic realm of edge computing.

Kong et al.'s study [11] elucidates edge computing's pivotal role in addressing Internet of Everything (IoE) challenges. They identify edge computing deficiencies within IoE, emphasizing service migration, security, and deployment hurdles. Advancements in AI, blockchain, and microservices offer promising solutions. They advocate for interdisciplinary integration, highlighting edge computing's transformative potential in various sectors. Their survey provides an up-to-date overview, covering edge computing's definition, models, and characteristics. It explores emerging trends and innovative solutions, underscoring the importance of technology integration.

Alongside the primary survey works within this research area, we have also analyzed prominent studies dedicated to modeling frameworks tailored for specific and general application scenarios. Each of these works endeavors to tackle various aspects and challenges associated with the development of edge-cloud applications.

The work in [8] presents ITEMa, an Iot-based smarT Ecosystem Modeling Approach, which is tailored to taking into account issues related to the design of edge-based systems and the location of execution of the designed components. The ITEMa approach provides a visual notation allowing to highlight the modeled entities along with their interactions and the exchanged information while catching the behaviors of the modeled entities at a glance.

In [7] the Smart Environment Metamodel (SEM) Framework is proposed, which aims at supporting the analysis and design of Smart Environments leveraging the Internet of Things and the Edge Computing paradigms. It provides two different design points of view, the first related to functional aspects and the second oriented to capturing data requirements. Both the approaches in [7], [8] do not explicitly model communications and do not provide tools for identifying quality

of service indexes and requirements.

The work in [12] presents a methodology based on Docker and Kubernetes that enables containerization and orchestration of robotic applications in heterogeneous and hierarchical edge-cloud continuum architectures. The authors show that by exploiting their approach they obtained reduced load on the robot's HW with minimal network overhead thanks to the optimized distributed edge-cloud system. The design methodology is, however, application-specific, and the paper does not discuss its eventual adoption in general contexts.

The EdgeFlow IoT framework is proposed in [3]. EdgeFlow is conceived to assist the developer in the application development process, by providing a methodology suitable for realizing latency-sensitive IoT applications, based on an extension of the Flow-based programming paradigm. The framework focuses on capturing latency and timing requirements and does not consider other valuable aspects such as data locality or deployment cost.

The paper in [16] proposes an IoT application development framework, named IADev, which uses attribute-driven design and Model-Driven Development (MDD). The IADev framework is composed of two steps, including iterative architecture development using attribute-driven design and generating models to guide the transformation using MDD. The provided development framework is proven to be able to support the development of cloud-centric IoT applications, but it is not specifically tailored for supporting the design of applications that take advantage of the capabilities of the edge-cloud continuum.

With respect to the above-mentioned works, the presented approach aims to be general enough to provide a useful tool for designing and developing edge-cloud applications, supporting the separation of concerns and the identification of quality of service requirements, and the selection of the more fitting provider/service for implementing and deploying the target application.

III. Edge-Cloud Continuum Architecture

The architecture of an edge-cloud continuum system is characterized by multiple layers, each of which serves distinct purposes and contributes to the overall efficiency of data processing and analysis. At the bottom layer, known as the *device layer*, are devices like smartphones, GPS units, and on-board cameras, which generate data transmitted to the nearest edge server. Above this lies the *edge layer*, comprising hardware components such as gateways and micro data centers, essential for collecting and partially processing data from the device layer. Optionally, a *fog layer* can handle tasks between the edge and cloud layers, enhancing computational resources and efficiency. Meanwhile, the *cloud layer* provides an extensive range of compute and storage resources, dynamically allocated for tasks beyond the capabilities of edge servers. This hierarchical arrangement ensures seamless data flow and effective resource utilization in the edge-cloud continuum system [13].

Recently, a new edge-cloud architecture has been proposed, which addresses several critical aspects [1]. These include the growing demand for sovereign edge and cloud technologies, the expected increase in electrical power consumption due to widespread cloud adoption, and the rising need for on-premise micro data centers to ensure privacy and low latency. This alternative architecture introduces new intermediate layers in the edge-cloud continuum, facilitating the flow of data from devices to the cloud:

- *On-Device*: at the foundational level, IoT devices themselves emphasize data processing directly on the device. This layer handles initial processing, including filtering and storage, supporting local learning tasks and potential integration into federated learning frameworks.
- *On-Premise*: consisting of private edge nodes, this layer is situated within local facilities, offering a good level of proximity and autonomy. Close to data sources such as gateways, street lights, or base stations, these nodes are located within a small distance, typically less than a kilometer, and require real-time connectivity to dedicated business systems.
- *Far Edge*: representing computing resources deployed even closer to IoT devices, public small aggregation nodes or physical containers within this layer cover specific areas, typically less than a hundred kilometers and sometimes near-premise, to address hotspots. They ensure rapid data aggregation and preprocessing, optimizing data transmission to the higher layers of the architecture.
- *Near Edge*: positioned closer to the cloud than traditional edges, public mini data centers within this layer, situated several hundred kilometers away from devices, enable more immediate data processing. They also facilitate multi-tenancy, enhancing responsiveness in handling real-time data streams.
- *Cloud*: serving as the central hub with vast processing resources, public data centers in this layer offer multi-tenancy and are typically located more than a thousand kilometers away from devices. Here, complex analytics tasks and global learning initiatives are supported, ensuring scalability and reliability.

In order to address evolving edge-cloud demands, strategic measures are essential. New public cloud data centers must decentralize capacities and ensure uniform latency experiences. Concurrently, the development of public edge data centers, forming a capillary network, optimizes proximity benefits and reduces costs through shared infrastructure. Additionally, on-premise micro data centers, potentially utilizing physical containers, efficiently address privacy and latency requirements. These initiatives collectively strengthen the edge-cloud continuum, meeting diverse needs while enhancing efficiency and accessibility.

IV. Modeling edge-cloud application

In our methodology, an *application* is envisioned as a cohesive ensemble of software components operating within the dynamic landscape of the cloud/edge continuum. These components work together to achieve common goals, with

ownership and management generally vested in a single stakeholder, despite potential interactions with third-party services. Each application is characterized by its unique set of *constraints*, encompassing factors such as execution cost and environmental impact, which it inherently owns.

Within this framework, the functionalities pursued by an application are conceptualized as *abstract services*, representing its core functionalities. These abstract services are systematically modeled through the delineation of *workflows* comprising interconnected tasks with specific dependencies. *Functional* and *non-functional requirements* are attributed to each abstract service, complemented by *Quality of Service* (*QoS*) metrics that gauge its operational efficiency. Thresholds are established to ascertain acceptable behavior for each abstract service.

The *tasks* that compose an *abstract service* are the software components that are necessary to provide the *abstract service* itself, and can also be shared between different *abstract service*s. Each *task* is described by a behavior, a set of functional and non-functional requirements, as well as QoS indexes. We identify three main kinds of tasks, namely, *computation* task, *communication* task, and *storage* task.

- A *computation task* represents a component primarily responsible for processing input data and delivering the resulting output to other components. Such tasks encompass various operations, including data processing, model training, and the utilization of learned models. Moreover, they encapsulate functionalities such as user interface and integration with third-party services. Requirements for computation tasks may include specifications regarding computation capabilities (such as CPU, GPU, or TPU), memory requirements, and throughput. QoS metrics associated with computation tasks typically include parameters such as execution time for each request or the volume of data processed within a specified timeframe.
- A *storage task* represents a component responsible for managing data persistence, including various types of database management systems such as Relational, NoSQL, Graph-Oriented, and Object-Oriented databases. Examples of requirements for a storage task include available memory, capacity for storing and managing objects, query execution times, persistence policies, data geographical location, and other relevant factors.
- A *communication task* represents a component responsible for managing communication and information exchange among other components. It encompasses various communication protocols, such as publish-subscribe, direct communication, and peer-to-peer (P2P), and is subject to requirements and QoS considerations related to factors like latency, throughput, security, and accountability. Additionally, a communication task facilitates interaction with physical or smart objects through appropriate protocols or channels. This may involve interfacing with physical sensors, actuators, or more complex devices deployed directly in a physical environment.

Modeling an application with the proposed abstractions can significantly simplify both application development and the subsequent deployment phase. Once an abstract model not tied to a specific platform has been created, it becomes possible to explore the catalogs of services offered by various edge/cloud providers. This exploration aims to identify services in line with the requirements outlined in the application model. Subsequently, for each chosen provider, it is possible to explore different concrete compositions of services and implementation options to select configurations that optimize QoS or minimize the expected costs. Ultimately, the designer can make an informed decision by selecting the supplier and configuration that best aligns with his expectations.

V. Case Study: Plastic Sorting in a Recycling Facility

To validate our modeling approach outlined in Section IV, we present a case study within the Industrial Internet of Things (IIoT). In this domain, the convergence of physical and digital technologies is transforming traditional industrial processes. Our study focuses on leveraging machine learning models deployed at the edge to tackle real-world challenges encountered in industrial settings, such as sorting different types of plastic in recycling plants.

Traditionally, plastic sorting has been labor-intensive and error-prone, relying on manual methods. However, machine learning offers a promising solution by automating the classification process through rapid image processing. Our proposed architecture, a distributed system, utilizes both edge and cloud computing resources. Sensor-equipped devices deployed within the recycling facility capture images of plastic objects as they move along conveyor belts. These images are processed by machine learning models trained to discriminate various types of plastic, enabling automated sorting.

The integration of physical devices and real-time management of actuators is crucial in this context. Additionally, the system facilitates middle-term and long-term storage of events and video streams for performance monitoring and model enhancement. Proper application modeling enables efficient utilization of computational resources and services across the continuum, ensuring deployment meets the time requirements of processing, communication, and storage tasks.

In the following, we present a platform-independent design of the application, based on the model we defined. Subsequently, we demonstrate a feasible implementation leveraging AWS services across all levels of the edge-cloud continuum.

A. Platform-agnostic design of the application

Given the above-described scenario, it is possible to exploit the guidelines provided in Section IV to design a distributed application for edge-cloud architecture. This is depicted in Figure V. We can identify two main abstract services, that are the “plant management” and the “machine learning model enhancement”.

The *plant management* models the real-time management of the recycling plant and furnishes the plant management team

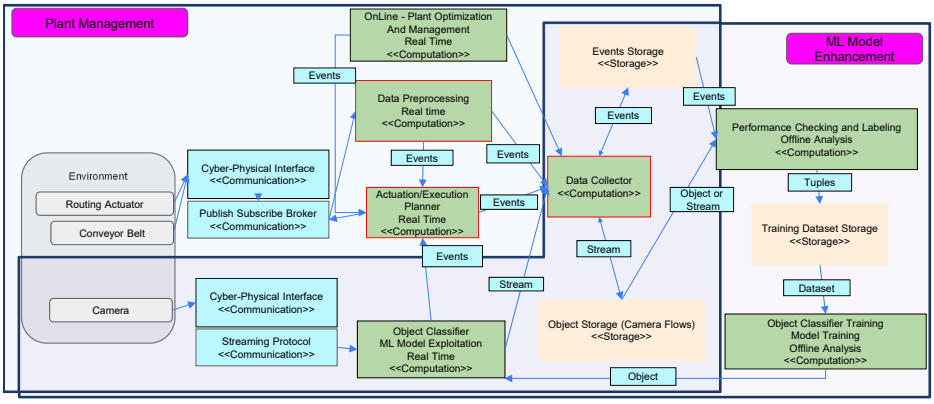

Fig. 1. Case Study modeling

functionalities for real-time performance analysis and fault detection, as well as an interface for controlling physical devices and actuators within the plant. From a bottom-up view, the plant comprehends a physical environment to control, which consists of one or more conveyor belts, routing mechanisms, and cameras, each provided with its specific sensing and actuation capabilities. The camera stream is elaborated within the *object classifier* computational task, which makes available information about the kinds of plastic that are currently running on the conveyor belts.

The conveyor belts and routing actuators provide information that is processed by a *data preprocessing* task, which is in charge of transforming and cleaning raw data, and making it available to the *actuation/execution planner* computational task. The latter also exploits the information gathered by the *object classifier* computational task to control the belts and the routing actuators to properly sort the items on the belts w.r.t. their material. All the events and streams required for this control process are sent to the *data collector* computational task, which is responsible for properly storage the information and making it available for further and future analysis, thus exploiting two storages, one tailored for managing events streams, such as a relational database, and another for managing large video streams acquired by the cameras. The *plant optimization and management* computational task is, in the end, in charge of providing to the plant management team all the interfaces and functionalities for the online visualization, analysis, and control of the plant, and thus operates with the *actuation/execution planner* and the *data collector* tasks.

The other modeled abstract service is the *ML model enhancement*, which is an offline service that is conceived to enhance the machine learning model exploited by the *object classifier* task during the plant lifetime. This abstract service runs off-line and comprehends a *performance checking and labeling* task that, given the previously collected data about the operating plant, provides functionalities to the plant management team to support the detection of miss-classified objects, and the labeling of newly acquired streams of data, which could also be done by exploiting more complex and off-line classification algorithms which are not efficient enough to be suitable for direct exploitation during plant control. The new labeled data are managed by a *training dataset storage* task, which makes it available to an *model trainer* computational task, which tries to enhance the detection model that will finally be provided to the *object classifier* computational task. All the communications between the computational and storage tasks are modeled by communication tasks, specifics for the kind of data that they should manage.

Given the above application model, it is possible to reason about the requirements and quality of service indexes of each task and the overall abstract services to be met in the concrete deployment. The modeling conducted provides an analysis of key features within our case study, including abstract services and tasks. However, it was modeled based on a simple client-server architecture, lacking specifications regarding the execution distribution—whether on devices, intermediate levels (e.g., on-premise, near-edge, and far-edge), or in the cloud. In the following section, we describe how these services can be implemented on a platform such as AWS.

B. Feasible implementation on AWS

AWS offers a comprehensive suite of services tailored for the cloud, ensuring seamless integration and deployment across diverse environments. From traditional offerings such as compute, data, and communication services to advanced solutions like machine learning services such as Amazon SageMaker, AWS provides robust capabilities for data processing and model deployment at scale.

Moreover, AWS caters specifically to the requirements of the edge-cloud continuum. In the physical environment domain, AWS equips users with tools and services for managing physical devices like actuators, cameras, and sensors. These components play a pivotal role in real-time data collection and monitoring, empowering organizations to gather insights directly from the field. AWS IoT Greengrass is a software that extends the capabilities of the cloud to local devices. It enables devices to collect and analyze data closer to the source of information, autonomously respond to local events, and securely communicate with each other over local networks.

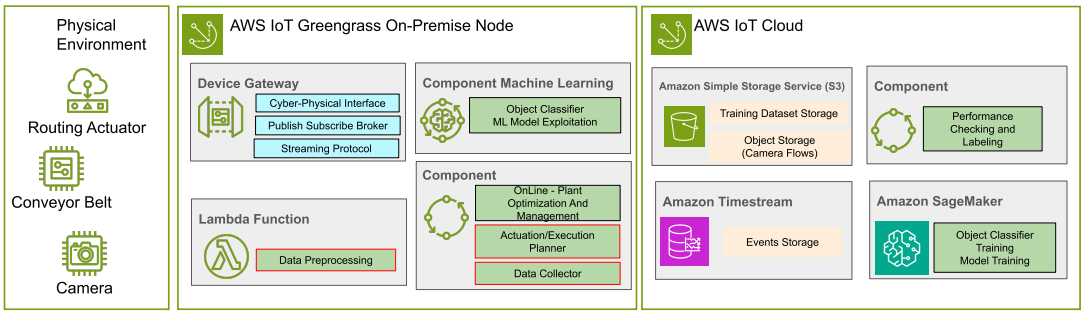

Fig. 2. Case Study AWS deployment

Local devices can also securely communicate with AWS IoT Core and export IoT data to AWS Cloud. With AWS IoT Greengrass, developers can utilize AWS Lambda functions and pre-built connectors to create serverless applications that are deployed on devices for local execution.

We have leveraged the services provided by AWS across three distinct levels (as depicted in Figure V-A). At the physical environment level, we manage routing actuators, conveyor belts, and cameras. These devices generate data that is crucial for real-time monitoring and decision-making. The intermediate level, managed on-premise by AWS IoT Greengrass, facilitates interaction with devices through a device gateway. Here, Lambda functions execute data preprocessing tasks, and machine learning models are applied locally for object classification. Additionally, components for plant optimization and management, including actuator execution planning and data collection, are also present. In the cloud, we utilize Amazon S3 for storing training models used to train SageMaker ML models. We also utilize Amazon Timestream for event storage, performance checking, and labeling.

Of course, this represents just one possible combination of AWS services aligned with our abstract model. There exist alternative services that could be selected based on specific project requirements. Ultimately, the choice of a particular service over another depends on various factors, provided there are no constraints preventing the selection of a specific service.

VI. Conclusion

The paper presented an approach and a set of abstractions designed to facilitate application development within the edge-cloud continuum. This methodology involves identifying abstract services, which are modeled as workflows comprising computation, communication, and storage tasks. Each task is tailored to address different application requirements and quality of service metrics. The objective is to establish a model that assists in selecting the appropriate execution platform within the edge-cloud continuum and choosing the most suitable services within that platform's ecosystem.

In future work, we intend to enhance this modeling framework by incorporating tools to formally define requirements and quality of service criteria at both the abstract service and task levels. This advancement will enable the automation of the process of selecting the best provider and configuration for a given application, potentially achieved through the implementation of optimization or heuristic algorithms.

References

[1] European industrial technology roadmap for the next generation cloud-edge offering, May 2021.

[2] Marica Amadeo, Franco Cicirelli, Antonio Guerrieri, Giuseppe Ruggeri, Giandomenico Spezzano, and Andrea Vinci. When edge intelligence meets cognitive buildings: The COGITO platform. *Internet Things*, 24:100908, 2023.

[3] Cosmin Avasalcai, Bahram Zarrin, and Schahram Dustdar. Edge-flow—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] Yao Chiang, Yi Zhang, Hao Luo, Tse-Yu Chen, Guan-Hao Chen, Huan-Ting Chen, Yan-Jhu Wang, Hung-Yu Wei, and Chun-Ting Chou. Management and orchestration of edge computing for iot: A comprehensive survey. *IEEE Internet of Things Journal*, 2023.

[7] Franco Cicirelli, Giancarlo Fortino, Antonio Guerrieri, G. Spezzano, and Andrea Vinci. Metamodeling of smart environments: from design to implementation. *Adv. Eng. Informatics*, 33:274–284, 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] 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.

[10] Linghe Kong, Jinlin Tan, Junqin Huang, Guihai Chen, Shuaitian Wang, Xi Jin, Peng Zeng, Muhammad Khan, and Sajal K Das. Edge-computing-driven internet of things: A survey. *ACM Computing Surveys*, 55(8):1–41, 2022.

[11] Xiangjie Kong, Yuhan Wu, Hui Wang, and Feng Xia. Edge computing for internet of everything: A survey. *IEEE Internet of Things Journal*, 9(23):23472–23485, 2022.

[12] Francesco Lumpp, Marco Panato, Nicola Bombieri, and Franco Fummi. A design flow based on docker and kubernetes for ros-based robotic software applications. *Trans. on Embedded Computing Systems*, 2023.

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

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

[15] Garima Nain, KK Pattanaik, and GK Sharma. Towards edge computing in intelligent manufacturing: Past, present and future. *Journal of Manufacturing Systems*, 62:588–611, 2022.

[16] 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.