Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Edge Deployment for Smart Cities Applications in Kubernetes: Mechanisms and Optimization Techniques

KAOUTHER GASMI[1], PAOLO BURGIO [1,2], MARCO MAMEI [1,3]
[1]Artificial Intelligence Research and Innovation Center AIRI, University of Modena and Reggio Emilia, Italy (e-mail: kaogasmi@unimore.it, paolo.burgio@unimore.it, marco.mamei@unimore.it)
[2]Dipartimento di Scienze Fisiche, Informatiche e Matematiche, University of Modena and Reggio Emilia, Italy
[3]Dipartimento di Scienze e Metodi dell'Ingegneria, University of Modena and Reggio Emilia, Italy

Corresponding author: Kaouther Gasmi (e-mail: kaogasmi@unimore.it).

This work was supported in part by PR-FESR EMILIA ROMAGNA Project Support System for Sustainable Smart Cities - S4C. Grant (PG/2023/308180). CUP (E27G22000310007), and by EU project "A catalyst for EuropeaN ClOUd Services in the era of data spaces, high-performance and edge computing(NOUS)", Grant Agreement Number 101135927.

ABSTRACT Edge computing is a distributed computing framework where computation, storage, and network resources are moved close to the data sources. Applications in Smart Cities can use Edge Computing to meet their requirements in terms of latency, real-time processing and scalability. Kubernetes has become the *de-facto* standard to deploy services in such a domain and a service placement process selects the appropriate edge nodes with sufficient hosting capacity to deploy and run the services based on their demands, constraints and performance criteria. Smart cities' applications challenge this selection as placement approaches must consider applications' criticalities and users' mobility to satisfy applications requirements. In this paper, we review the current research conducted on service placement approaches in Kubernetes edge computing for smart cities applications. We provide a comprehensive and detailed survey, analyzing and classifying the current research in terms of mechanisms, optimization techniques, objectives and use cases being adopted. Additionally, we identify the main concerns and weaknesses of each research. After that, we elaborate on open issues and future research directions.

INDEX TERMS Edge computing, Service deployment and placement, Smart city, Survey

I. INTRODUCTION

SMART cities leverage key information and communication technologies (ICT) including the internet of things (IoT), AI, big data analytics, and others to improve their operational efficiency and quality of life for citizens by providing optimized smart services. The rapid proliferation of IoT technology has empowered smart cities to collect vast amounts of data in real time. IoT devices and sensors form the foundational infrastructure for smart cities (SC), continuously capturing and integrating data to enhance urban operations. However, managing and processing such massive-scale data requires robust computing and storage solutions. To address this challenge, cloud computing was introduced, offering on-demand storage and processing capabilities in a cost-effective and scalable manner. Its key features, including vast storage capacity and resource pooling, have facilitated widespread adoption across various domains. Despite its advantages, cloud computing has inherent limitations, such as high latency, lack of context-awareness, and limited support for mobility. These drawbacks pose significant challenges for time-sensitive applications in smart cities. For instance, Vehicle-to-Vehicle (V2V) communication systems, which are crucial for collision avoidance, are not well-suited to a centralized cloud-based architecture due to latency issues and inadequate mobility support.

To overcome these limitations, edge computing has emerged as a promising solution. By shifting data storage, processing, and analysis closer to the network edge—near the devices generating the data—edge computing enables resource-intensive and time-sensitive applications in smart cities. This decentralized approach ensures lower latency, improved real-time responsiveness, and enhanced computational capabilities, making it particularly valuable for applications that demand immediate processing and minimal delay.

In order to exploit the flexibility given by edge and cloud infrastructures, Microservice-based architecture (MSA) have been use to divide an application into small independent modules, called micro-services, which are deployed and managed independently and communicate over a network [1] [2] [3]. The development of MSA-based applications using container virtualization technology [4]is currently revolutionizing the way developers build and deploy their applications. Containers are the de facto alternative to traditional virtual machines, due to their high portability and low resource usage [4]. This approach consists of packaging the different services that make up an application in separate, intercommunicating containers, which are deployed in a cluster of physical or virtual machines. Many cloud service providers offer Containers as a service (CaaS) as a cloud service model that simplifies the deployment of containerized applications in the cloud [5]. From the perspective of the cloud service provider, a CaaS platform creates an abstraction layer that includes a container orchestration engine, typically based on the *de facto* standard Kubernetes [6] (see Figure 1). Kubernetes automates the deployment, scaling, and management of containerized applications, where each application's component is run in its own container. Kubernetes have been used by many organizations in a diverse area of underlying applications and have gained the trust of being the best option for the management and deployment of containerized applications [7].

As the computing paradigm moves towards edge and fog computing, Kubernetes face multiple key challenges when deployed in edge environment for smart cities,specifically. The Kubernetes orchestration system is not inherently well-suited for edge environments due to several key limitations. First, its design lacks topology awareness between nodes, which is critical for workloads that depend on frequent inter-node communication. Second, Kubernetes does not natively support network-related metrics, as it only allows resource requirements to be defined for CPU and memory. This omission makes it challenging to meet the demands of latency-sensitive and data-intensive applications, which require precise control over network latency and bandwidth [8]. Additionally, Kubernetes struggles with inefficient load balancing in distributed edge settings, often resulting in significant delays when requests are routed to remote nodes [9]. These shortcomings highlight the need for enhancements to better align Kubernetes with the unique demands of edge computing environments.

The diverse challenges associated with default Kubernetes scheduling in edge computing for smart cities underscore the need for a thorough examination of proposed approaches and mechanisms. These solutions must address the dual complexities of edge environments—dynamic, distributed, and resource-constrained—and the stringent requirements of smart city applications, such as low latency, high reliability, and efficient resource utilization. The primary motivation of this study is to systematically identify, analyze, and summarize the existing literature on scheduling strategies specifically designed or adapted for Kubernetes. By doing so, we aim to provide insights into how these strategies can bridge the gap between Kubernetes' default capabilities and the unique demands of edge computing in smart city contexts. To this end we try to answer the following key research questions:

- **RQ1:** How the edge computing support the smart cities applications?
- **RQ2:** What are challenges faced by Kubernetes when deployed at the edge in the context of smart cities?
- **RQ3:** What categorization can be assigned to application deployment methodologies in Kubernetes edge computing?
- **RQ4:** How do they solve the above challenges?

In the following of this paper we first describe the smart city application domain focusing on applications that benefit from service deployment on edge (Section 2). Then, we provide a background on edge computing architecture and application focusing on services provided by the edge to smart city application. This will answer RQ1 (Section 3). Then, we present the main topic of the paper and analyses edge service placement our scenario. First we describe the challenges faced by Kubernetes when deploying services on the edge - RQ2. Then we discuss what are the objectives that different service placement try to optimize - RQ3. We also provide an analysis to understand which placement mechanism best suit which use case - RQ4 (Section 4). Finally, we conclude highlighting challenges and future research directions.

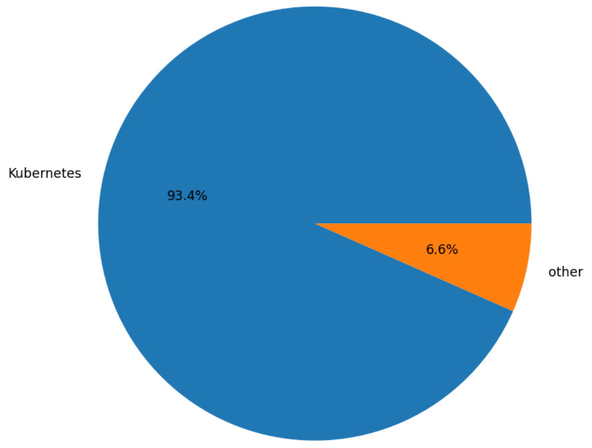


FIGURE 1: Applications deployment in edge environment using Kubernetes.

II. BACKGROUND

In this section, we provide the necessary background regarding the concepts of smart cities, edge computing and Kubernetes tool.

A. SMART CITIES

Smart city is an urban area that has been developed with advanced technologies such as IoT, AI, and data analytics to improve the quality of life for its inhabitants and business. The ultimate goal of a smart city is to improve the city's livability, sustainability, and efficiency, while also reducing the environmental impact and enhancing the economic growth of the city as outlined in Section 1.

The concept of smart city architecture is built upon several essential factors, including seamless communication, efficient data collection, robust data privacy and security measures, advanced networking capabilities, effective data management, and strategic business modeling. These elements work together to create a well-integrated and sustainable urban ecosystem that leverages technology to improve the quality of life for citizens.

A fundamental aspect of smart city architecture is ensuring interoperability, by enabling seamless integration and communication among various systems and devices, which means that the system must be capable of seamlessly connecting and interacting with a diverse array of heterogeneous devices. This includes various Internet of Things (IoT) sensors, smart infrastructure, communication networks, and data platforms, all of which must function cohesively despite differences in manufacturers, protocols, and technologies. This is essential for the efficient functioning of smart city services and applications.

By prioritizing interoperability, smart cities can facilitate real-time data exchange, enhance decision-making processes, and support automation across multiple sectors, such as transportation, energy management, public safety, and healthcare. This not only improves operational efficiency but also fosters innovation, scalability, and adaptability, ensuring that smart cities remain resilient and future-proof in an ever-evolving technological landscape.

Several Smart cities architectures have been proposed in the literature [10], [11], [12], [13], [14] basic architecture consisting of four layers described as follows (see Figure 2):

- **Perception layer:** referred also to the device layer, comprises diversity of sensors such as sensors network and physical devices including cameras, RFID and IoT devices. These devices are capable of perceiving, identifying, detecting, collecting, and exchanging data with other devices. This data is then pushed to the network layer winch provides connectivity and transport capabilities.
- **Network layer:** This layer acts as a bridge between the perception and application layers by establishing connection channels, handling data transmission and dealing with critical issues such as security and data privacy. Communication technologies such as Wireless Sensor Networks (WSN), Wireless Local Area Network (WLAN). Mobile Networks (4G, LTE), LoRaWAN can be used in this layer.
- **Data management layer:** This layer's functions include storing, processing, managing and handling of the large amount of heterogeneous data generated from connected devices (e.g perception layer). The processed data is sent automatically to the application layer through communication services for required actions. Big data analytics approach, Artificial Intelligence (AI), Machine Learning (ML) algorithms and Edge-cloud continuum can be applied to this layer.
- **Application Layer** This layer, often referred to as the user-end, is responsible for delivering end-to-end application support tailored to specific scenario requirements. In this layer, the application receives processed data and possibly sends instructions to actuating devices. The key technologies enablers at this level are standardized application layer protocols such as Constrained Application Protocol (CoAp), Message Queue Telemetry Transport (MQTT), Extensible Messaging and Presence Protocol (XMPP) Services, Representational State Transfer (RESTFUL) and Advanced Message Queuing Protocol (AMQP).Protocols standardization promote interoperability that enables devices and system exchange data seamlessly.

B. EDGE COMPUTING

Edge computing is a computing paradigm that involves processing data at the edge of the network, closer to the data source and end-users, rather than relying solely on centralized cloud computing. This approach addresses several challenges associated with traditional cloud computing, such as high latency, bandwidth constraints, and privacy concerns. By bringing computational power closer to IoT devices, smartphones, and other connected technologies, edge computing enhances response times, reduces bandwidth costs, and improves data security and privacy.

A typical edge architecture comprises an IoT layer (also known as perception layer) that consists of IoT devices, a Edge layer that consist of edge node, and a cloud layer with a cloud data center and services [15]. A three-layer structure is illustrated in Figure 3 and discussed in more detail below.

- **IoT layer:** The IoT layer is situated at the bottom and closest to end-users. This layer consists of all the IoT devices connected to the internet including several types of devices such as sensor, smart vehicles, smartphones, tablets, and others. In a typical set-up, these IoT devices only generate data and cannot process it. The primary purpose of this tier is to acquire, measure and collect data from these devices and send it to the upper layer. In addition, the computation resources of IoT devices assigned to user devices are limited, they have low computational power, constrained by battery life and low processing capacity. Hence, devices can request the upper layers to offload tasks that require many computation resources to run and accelerate the computation.
- **Edge layer:** Edge computing refers to a new computing model which implements the computation of tasks at the periphery of the network known as the edge layer.

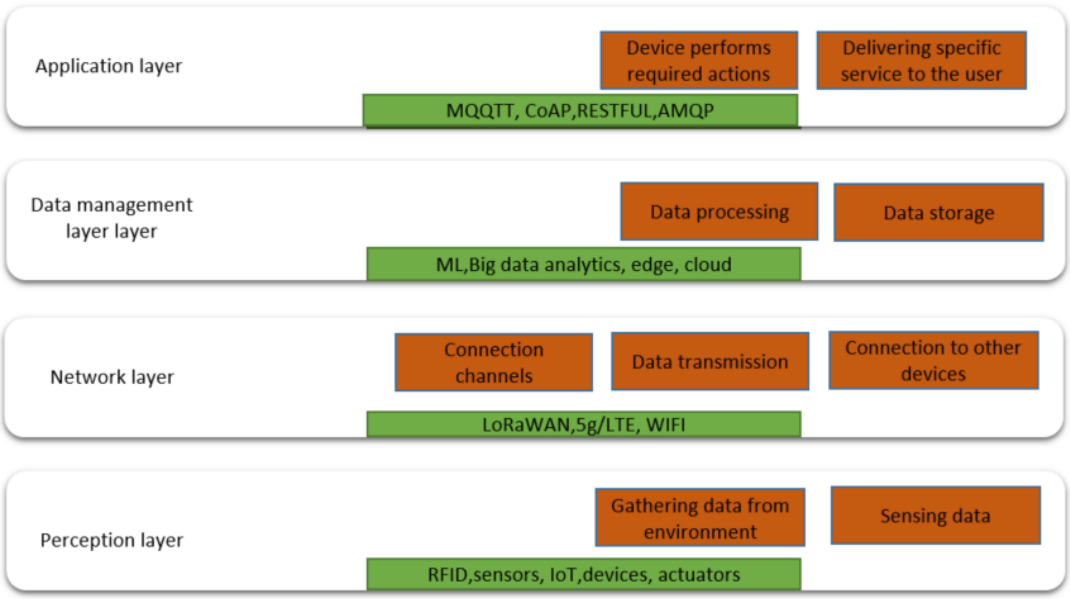

Application layer
Device performs required actions
Delivering specific service to the user
MQQTT, CoAP,RESTFUL,AMQP
Data management layer layer
Data processing
Data storage
ML,Big data analytics, edge, cloud
Network layer
Connection channels
Data transmission
Connection to other devices
LoRaWAN,5g/LTE, WIFI
Perception layer
Gathering data from environment
Sensing data
RFID,sensors, IoT,devices, actuators

FIGURE 2: Smart Cities Architecture comprises 4 main layers: perception layer, network layer, data management layer and application layer

Based on this concept, certain applications that may not require a significant amount of computing resources can be processed in the edge layer and not needed to be offloaded in the fog or the cloud. However, if resources at the edge layer are not available, the sensors will request to offload their tasks in the fog layer or the cloud layer for processing.
In the literature, three different practices of Edge computing have been employed: Cloudet, Fog Computing and Mobile Edge Computing (MEC)

- **Cloudlets**, introduced by Mahadev Satyanarayanan [16], refer to cluster of servers deployed in a single hope proximity of mobile users for enabling mobile devices to offload tasks in applications that requires intensive computations.
- **Fog:** consists of heterogeneous fog devices (e.g., access points, switches, routers, gateways, base stations, fog servers, etc.) that have relatively limited processing, storage, and communication capabilities. These fog nodes maintain a connection to cloud servers and can forward requests to cloud data centers. Fog nodes that have very limited processing, storage, and other capabilities are considered to be low-level nodes, while nodes with more abundant resources are classified as high-level nodes. Fog nodes may also be categorized as either stationary (if fixed at a certain location), or mobile, such as smartphones, vehicles, and drones,etc.
- **Mobile Edge Computing** was introduced by European Telecommunications Standards Institute (ETSI) to enable the placement of computing resources at the radio access network and core network of mobile telecommunication systems [17]. The mobile edge computing servers can be placed at different positions in the radio access network such as a macrocell BS [17], assigned to a group of BSs , or in a hierarchical fashion at different levels.

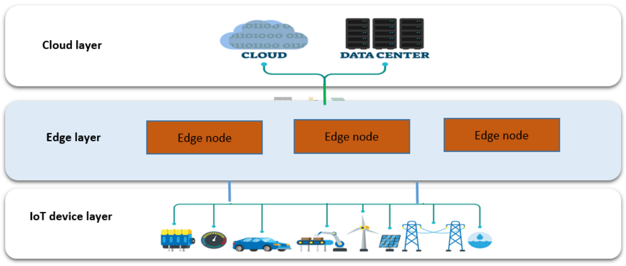


FIGURE 3: Edge Architecture comprises 3 main layers: IoT device layer, Edge layer and Cloud layer

- **Cloud layer:** The top-most layer of the edge-fog computing architecture is known as the Cloud layer. It includes various cloud servers and cloud data centers with significantly more computation power, efficient storage, and processing capabilities than the fog layer. This layer can perform complex processing and store large amounts of data [18] [15], [19].

C. KUBERNETES

Kubernetes is defined as "open-source system for automating deployment, scaling, and management of containerized applications" [6]. applications are packaged into *pods*, which constitute the smallest unit of deployment. A pod cloud contain either one container or a collection of containers. The

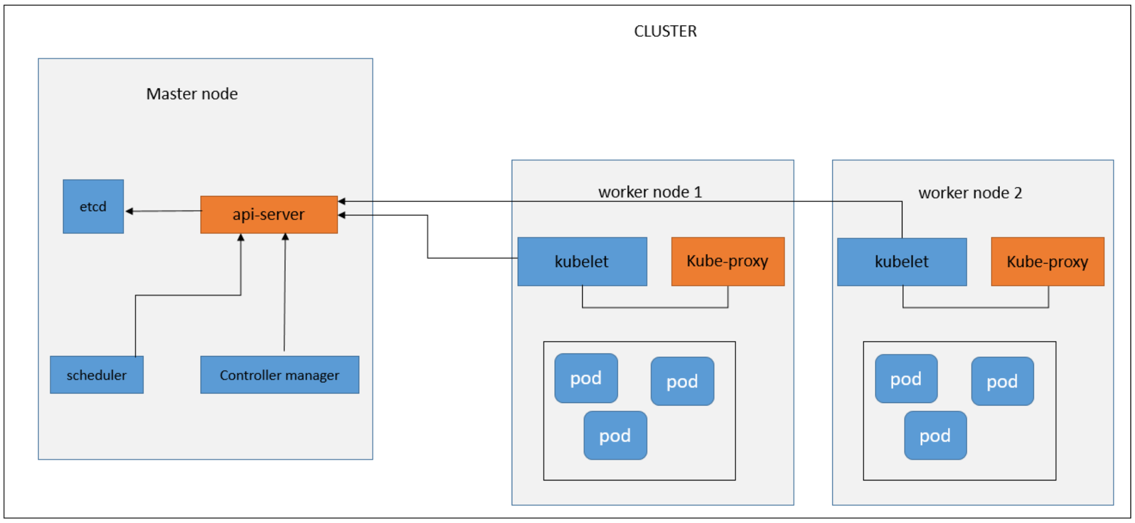


FIGURE 4: Kubernetes Architecture based on [6].In this survey we focus primarily on the scheduler component of the master node. To understand the impact of different schedulers on edge deployment in smart city scenatios.

containers inside a pod share the same IP Address and namespace while Containers in different pods are isolated from each another since they have different IP addresses and hostnames. Two types of nodes exist in Kubernetes:

- **Master Nodes** that run the so-called control plane components. These nodes are composed of the following components: *API server*, provide an entry point to control the entire Kubernetes cluster. *Kube-Scheduler*, is the default scheduling component in Kubernetes which is responsible for assigning a specific node in the cluster to each pod. The Kube-scheduler operations are further detailed in the next section. *Controller manager*, responsible for monitoring overall state of the cluster. If the state of the system changes, the Controller Manager communicates the desired state of the system through the API server.
- **Worker Nodes**, responsible for running pods. They are composed of the following components: *Kubelet*, *kube-proxy*, *container runtime*. Figure 4 shows the different interactions among these components.

The goal of this survey is to present the main services offered by edge computing platforms to the smart city scenarios and to show how novel Kubernetes-based solutions can support the deployment of such services.

III. EDGE COMPUTING IN THE CONTEXT OF SMART CITIES

In this section, we illustrate how edge computing address to the requirements of smart cities application listed in the Section 2. Then we report the common application supported by edge.

A. EDGE COMPUTING SERVICES ENABLING SMART CITY APPLICATIONS

The main edge services supporting smart city applications can be grouped as follows:

- **Low latency and real-time processing.** Edge computing can support latency-sensitive applications by enabling real-time data processing and analysis at the network's edge, which enable the deployment of delay-sensitive applications close to end devices. This will cope with the strict requirements of the future time-sensitive smart city applications [20].
- **Big Data management.** Edge computing provides constrained and decentralized infrastructures resources. EC support data management requirements by decentralizing data processing closer to data sources. In addition, it enables context-awareness, which allow a selective data storage, ensuring that only relevant data can be processed locally, while less pertinent data can be processed in the cloud. This reduce latency and load on network core [21].
- **Mobility support.** This is another service that motivates users to leverage EC. Mobility can be considered for computation, data and users. EC provides a stable connectivity between mobile devices through handover/hand-off techniques [22].
- **Reliability and availability support.** Edge computing enhances reliability and availability in smart cities by processing data closer to the source, reducing latency and improving response times. This decentralized approach allows for seamless communication and efficient management of services, such as intelligent transport systems and healthcare platforms. By managing services from edge can ensure that services remain robust and accessible even under high demand or network constraint, thus supporting the essential service require-

ments of smart cities effectively [23]

- **Resource management.** The main aspects of resource management in Edge computing include discovering all available resources, monitoring and provisioning them and resource mobility management . Resource discovery and monitoring aim to know which resources are available for use, where they are located and how long they will be available while goal of resource provisioning is to provide optimal use of devices by sharing resources among applications. In addition, EC provides services to support resources mobility such as edge nodes duplication, deploying them around the potential locations of devices [24] [25].
- **Edge Analytics.** Edge computing designed with the ability to perform data analytics near the source of data [26]. Edge computing allows for the processing of data and running of AI/ML algorithms directly on devices or local servers, enabling faster insights and actions without sending data to the cloud. For instance, Predictive maintenance in infrastructure, smart grids, and real-time analytics in traffic systems are enabled [27] [26].
- **Data Security and privacy.** Edge computing supports smart cities' security and privacy by enabling localized data processing, which reduces the transmission of sensitive information over networks, thereby minimizing exposure to potential breaches [20]. Additionally, edge computing can implement advanced security mechanisms, such as blockchain technology and federated learning to guarantee the user's data privacy [28] [29] [22].

This basically responds to **RQ1** illustrating what are the services provided by Edge computing to support Smart city applications.

B. COMMON APPLICATIONS SUPPORTED BY EDGE COMPUTING

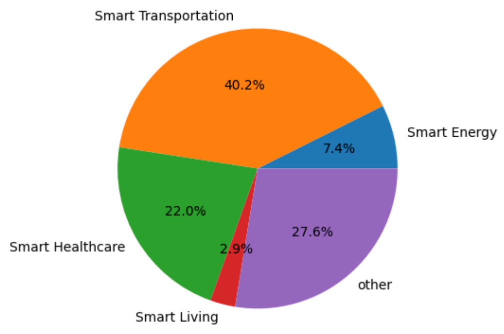


FIGURE 5: Edge enabled applications in smart cities, as extracted from systemaic searches on Google Scholar

Smart cities application can be grouped into five fundamental domains: smart transportation, smart energy, smart living and smart healthcare. A search of the Web search engine google scholar with keywords " Edge computing" AND " Smart city" AND " Application" shows the results in Figure 5. The figure shows that a large fraction of smart city applications belong to the above mentioned domains.

Smart Transportation Domain. Smart Transportation Systems (STS) represent a transformative shift from traditional transportation models to intelligent, on-demand mobility solutions. By leveraging advanced technologies, STS aims to enhance transportation efficiency, improve commuter experiences, and ultimately elevate the quality of life for citizens.

- **Autonomous Vehicles** Autonomous driving is transitioning from concept to reality, with projections indicating that 15% of vehicles sold by 2030 will be autonomous [30]. This evolution is driven by advancements in driving assistance and safety features, which reduce the need for human intervention in vehicle control. A critical requirement for safe autonomous driving is real-time data transmission between vehicles (V2V), infrastructure (V2I), and users (V2U) [30], enabling vehicles to perceive traffic patterns, make decisions, and act in real-time. Edge computing plays a vital role by processing sensor data (e.g., from cameras and radar) locally, facilitating immediate decision-making [31]. Additionally, AI models, particularly deep learning techniques, enhance real-time object detection and semantic segmentation, allowing autonomous systems to accurately identify pedestrians, cyclists, and other vehicles in complex urban environments [32]. Edge computing also supports the deployment of machine learning algorithms that improve decision-making capabilities. However, challenges such as geo-temporal variations, high mobility, and rapidly changing conditions complicate traffic and connectivity management. One proposed solution is leveraging vehicles as moving edge nodes to mitigate connectivity and handover issues.
- **Traffic safety** Smart cities leverage monitoring technologies and data-driven road safety systems to identify danger zones and implement preventative measures. For example, Forward Collision Warning Systems (FCW) [33] alert drivers in emergencies, enabling quicker reactions to avoid accidents. Timely delivery of such warnings is crucial for preventing collisions and enhancing road safety. Latency is a critical factor, especially for automated and connected vehicles, which require end-to-end latency between 10 ms and 100 ms for effective operation. Emergency messages, in particular, must be exchanged within 10 ms [34]. By deploying edge nodes near traffic cameras and sensors, real-time data can be processed without delay, enabling immediate decision-making by traffic monitoring systems [35] [36]. Furthermore, edge computing systems can analyze historical and real-time traffic data to identify patterns and predict

potential incidents before they occur [36].

Smart Energy Domain. Smart energy leverages recent advancements in Information and Communication Technology (ICT) to intelligently optimize energy consumption and costs, ultimately contributing to the development of a sustainable energy management system [33].

- **Smart grids** represent advanced energy distribution networks that integrate IoT-based devices, diverse sensors, communication technologies, and AI algorithms to manage and monitor energy flow from sources (e.g., solar, wind, hydro) to consumers (e.g., smart homes, buildings, industries) [37]. IoT devices, such as smart meters and sensors, collect real-time data on energy consumption and production across the grid. AI algorithms analyze this data to enable real-time energy monitoring, reduce energy waste, and prevent power outages. Smart grids also offer customers interactive energy usage services and accurate billing. To support low-latency, real-time monitoring, smart grids leverage edge and cloud computing for data storage and processing. Furthermore, as smart grids handle increasing volumes of sensitive user data, edge computing plays a crucial role in determining which data should be processed locally versus in the cloud. The reliable transfer of data between sensors and edge nodes underscores the need for a robust communication infrastructure that ensures real-time data delivery to all participating nodes.
- **Smart buildings** are a key component of smart cities, enhancing energy efficiency and sustainability. They utilize advanced energy management systems, such as intelligent lighting and HVAC systems, to optimize energy consumption and reduce waste. Smart meters enable real-time energy monitoring, while machine learning algorithms predict energy usage and optimize systems accordingly. Predictive maintenance algorithms also help identify potential equipment failures early, minimizing downtime and improving efficiency. Additionally, smart meters support dynamic pricing, adjusting energy costs based on demand. Smart buildings rely on edge and cloud computing for data storage and processing. However, like smart grids, they face privacy and security concerns due to the collection and storage of sensitive data.

Smart Living Domain. Smart living represents a comprehensive approach to improving the quality of life for urban citizens by integrating advanced technologies into residential environments. It encompasses a wide range of systems and services that utilize Information and Communication Technologies (ICT) to create more efficient, comfortable, and sustainable living spaces.

- **Smart homes :** Smart homes are a vital component of modern urban areas, integrating numerous IoT devices, sensors, and actuators connected via wireless networks to provide automated services like lighting control, security, surveillance, and energy management [38] [39]. These systems generate vast amounts of real-time data from interconnected devices, enabling immediate responses to user commands or environmental changes. For example, smart security systems use sensors and cameras to deliver instant alerts during potential intrusions [39]. However, the massive data generated requires robust storage and computational capabilities to support real-time processing, leading to significant computational demands [40]. Protecting this sensitive data from cyber threats is also critical [41]. To address these challenges, smart homes leverage both cloud and edge computing. Edge computing, in particular, processes data closer to its source, reducing latency and enabling rapid decision-making, such as real-time video analysis for security systems, which can send immediate alerts without cloud-related delays [42] [43].

Smart Healthcare Domain. Smart healthcare is a vital component in enhancing residents' quality of life while significantly improving the efficiency of healthcare delivery systems.

- **Remote patient monitoring:** use wearable devices (cardiac devices, airflow monitors, etc) that monitor vital signs of the patient remotely through wearable devices and IoT sensors. These devices collect data continuously, allowing real-time monitoring of patients by providing timely interventions and insights into patient's health status. This capability reduces the need for in-person visits, particularly for patients with chronics conditions.
- **Telemedicine:** allows healthcare providers to conduct virtual consultations with patients, facilitating access to medical advice and treatment without the need for physical appointments. Telemedicine frameworks in smart cities leverage IoT devices to create a connected healthcare ecosystem. Patients are equipped with smartphones and other digital devices can easily connect with healthcare providers for diagnosis, monitoring, etc. Advanced intelligence techniques enhance these frameworks by coordinating IoT resources and communication channels, resulting in a reliable enterprise ecosystem for smart cities. This integration build a resilient healthcare system for remote patients and communities [44].
- **AI based-diagnostics:** Smart healthcare also encompasses the use of AI for diagnostics and treatment recommendations. AI algorithms can analyse large datasets from various sources, including medical imaging and patient records, to assist healthcare providers in making informed decisions. For example, AI-assisted diagnostic tools can identify patterns in medical images that may be indicative of specific conditions, improving the accuracy and speed of diagnoses. Additionally, AI can personalize treatment plans based on individual patient data, enhancing the effectiveness of care. AI algorithms can analyze vast amounts of patient data to identify patterns that may indicate the onset of diseases, allow-

ing for earlier interventions and personalized treatment plans. The incorporation of AI not only enhances diagnostic accuracy but also streamlines administrative tasks, thereby reducing the burden on healthcare professionals.

In Table 1 we illustrate what are the main edge services enabling different smart city applications.

In the next section we present a review of the custom scheduling solutions based on Kubernetes that have been designed to address these limitations and aim to enhance Kubernetes' ability to efficiently manage resources and meet specific requirements for cities applications deployed across distributed edge environments.

IV. KUBERNETES SCHEDULING AND EDGE COMPUTING

In this section, we describe the Kubernetes default scheduler functionalities, then we answer the **RQ2** by identifying its limitations in edge environments for smart cities applications.

A. KUBERNETES SCHEDULING

The scheduler decision making process conducts two operations: *filtering* and *scoring*. The filtering operation, where the scheduler verifies which nodes are capable of running the pod by applying a set of filters, also known as *predicates*. The scheduler supports the predicates listed here [6]. The purpose of filtering is to solely consider nodes meeting all specific requirements of the pod further in the scheduling process. The scoring operation, where the scheduler ranks each remaining node to find the best fit for the pod provisioning based on one or more constraints, referred as *priorities*.The scheduler supports the priorities listed here [6].

B. LIMITATIONS

Although the default Kubernetes scheduler provides flexible and powerful feature,it faces several limitations in the context of edge computing for smart cities due to the unique characteristics and requirements of edge environments and smart cities applications. The key challenges are listed below.

- **Network and latency considerations** : Smart city applications, such as real-time traffic management, video analytics, and emergency response systems, often require ultra-low latency. The default scheduler does not adequately consider network latency and bandwidth, which are critical for latency-sensitive applications. It focuses on CPU and RAM usage for scheduling decisions [8] [45] [46]. Delays in scheduling decisions can lead to increased response times, which is unacceptable for time-critical applications in smart cities
- **Node Heterogeneity**: Kubernetes' default scheduling strategy is designed for homogeneous cloud environments and struggles with the heterogeneity of edge computing nodes, which have varying computing power and network bandwidth. This can lead to inefficient pod assignments and increased delays [47] [48] [49].
- **Fault tolerance**:The default scheduler lacks advanced fault tolerance mechanisms necessary for edge environments. This includes the ability to predict and recover from failures proactively, which is crucial for maintaining service continuity [50] [51].
- **Load Balancing**: Kubernetes employs a default load balancing mechanism that distributes workloads across nodes to optimize resource usage. The native load balancing strategy involves kube-proxy, which distributes client requests across all pods of an application. However, this can lead to inefficiencies, such as increased latency when requests are forwarded to remote nodes, especially in geographically dispersed edge environment. [52] [53] [54].

  In the next section, we review the advanced approaches that have been proposed so far in the literature to enhance the Kubernetes scheduling to overcome the mentioned challenges.

V. KUBERNETES-BASED APPLICATION DEPLOYMENT SOLUTIONS IN THE SMART CITIES CONTEXT

In this section, we try to answer the **RQ3.** *What categorization can be assigned to application deployment methodologies in Kubernetes edge computing?*

A. EXISTING SURVEYS

Several surveys in the literature discuss how edge and fog computing supports smart city applications [55] [56]. In [55] the authors discuss technologies and architectural frameworks that enable edge computing in the context of smart cities and highlight the key challenges related to edge computing deployments in smart cities around the world. The work in [56] reviews and compares various fog computing platforms and their capabilities for supporting different smart city use cases. Applications, challenges, architectures, and emerging trends about Fog/edge based smart cities are discussed here [57].

Other works have been published in the recent literature related to orchestration architecture in cloud edge environments [8] In [8], the authors reviewed edge-cloud orchestration architectures recently developed using Kubernetes. They analyzed how Kubernetes addresses essential requirements for edge orchestration in smart cities context and identifies the shortcomings for a successful deployment in smart cities. David and al. [58], review existing cloud edge orchestration architecture, tools, and platforms, highlighting their strengths and limitations. It categorizes solutions according to their focus.

Custom Kubernetes scheduling is discussed here [59] [60] [9]. In [59] a taxonomy for Kubernetes scheduling is introduced, categorizing existing techniques, and identifying gaps in current research and Highlights the limitations of the default Kubernetes scheduler. This study [60] explores various scheduling strategies and innovations in Kubernetes, highlighting their objectives and methodologies. [9] provides a classification of the existing literature on custom Kuber-

TABLE 1: Critical edge services enabling applications in smart cities.

Domain	Applications	Latency and real-time processing	Reliability and availability	Mobility support	Data handling	Data analytics	resource management	data privacy
Smart transportation	Autonomous driving	✓	✓	✓	✓	✓	✓	
	Smart traffic monitoring	✓	✓	✓	✓	✓		
Smart energy	Smar grids	✓	✓	✓	✓	✓	✓	
	Smart building	✓	✓	✓	✓	✓		
Smart living	Smart homes	✓	✓	✓	✓	✓	✓	✓
Smart healthcare	Remote patient monitoring	✓	✓	✓	✓	✓	✓	
	Telemedicine	✓	✓	✓	✓	✓	✓	
	AI based- diagnostic	✓	✓	✓	✓	✓	✓	✓

netes schedulers, categorizing them according to scheduling objectives, workload types, and environments. Table 2 summarizes the previous review and surveys.

This survey reviews the primary literature on custom Kubernetes scheduling strategies that have been developed for edge computing, particularly for smart cities applications.

B. KUBERNETES-BASED APPLICATION DEPLOYMENT SOLUTIONS

The Extending Kubernetes scheduling algorithm applications deployment in edge environments involves advanced methodologies. We categorize these methodologies into three categories; Heuristic-based, machine learning-based techniques and others methodologies. The heuristic approaches are widely used in traditional scheduling, are generally of low complexity, and generate a satisfactory schedule in a reasonable amount of time. machine learning based. These methodologies have the ability of learning and training from big data to make intelligent scheduling decisions based on what they have learned. Deep reinforcement learning is a subfield of machine learning that uses a layered structure of artificial neural networks to learn and make intelligent decisions is the most applied in the extended Kubernetes scheduling. In this section, we review the recent research studies from the 2020 to 2025 period belonging to the mentioned categories.

C. HEURISTIC-BASED METHODOLOGIES

In the literature, the heuristic based scheduling methodologies are aware of different optimization metrics associated to the infrastructure and the application Metrics such as network latency, bandwidth are related to the infrastructure context. Metrics such as response time and deadline are usually used to measure the applications requirements.

The authors in [62], [63], [64], [65], [66], In [67].address application deployment by developing network-aware scheduling strategies in which the default Kubernetes scheduler is extended to consider the state of the network infrastructure to make resource provisioning decisions. These enhancements aim to reduce network latency and improve the execution of delay-sensitive applications. In instance, in [62] the authors Propose two schedulers. The first scheduler is in charge to assign a node to service based on the latency between the selected node and the end device, aiming to reduce the end-to-end latency and a close placement proximity of application services to the end users. The second scheduler evaluates the initial placement decision made by the first scheduler and triggers services migration when end-user location change ensuring that service are optimally placed. The self-driving vehicle scenario-based experiments show that the average computing and networking costs was reduced by 10% to 25% compared to the default Kubernetes scheduling. Similarly, In [63] the real-time state of both the infrastructure and the application are considered in scheduling decisions. The scheduling process leverages resource availability and network awareness,e.g. communication interactions between microservices, for its decisions to improve the end-to-end latency. A rescheduler to improve application placement is also considered if network conditions change over time.The scheduling strategy is implemented by extending the Kubernetes scheduling logic. The results indicate that the average end-to-end network latency is improved compared with the default Kubernetes scheduler. The authors in [64] present a proximity-aware placement framework named "VioLinn" for latency-sensitive IoT services.The framework prioritizes placing tasks on the closest devices to minimize communication latency. The monitoring component estimates the latency from the current device to all other devices in the orchestration area, ensuring that tasks are executed on the nearest available resources. In addition, the study focuses on selecting spare edge devices to handle increasing demands in edge computing environments. To achieve to this, They extended Kubernetes capabilities by designing heuristic algorithms to make decisions based on predefined criteria, such as minimizing cost or maximizing QoS. VioLinn employs a distributed orchestration model, where each edge device can handle incoming tasks independently. Additionally, an area orchestrator is selected within each orchestration area to manage the number of edge devices and service replicas needed for the current load. The paper evaluates the proposed system using a Kubernetes-based testbed and simulations. The evaluations demonstrate that dynamic provisioning of spare resources is viable in terms of QoS and show the scalability of the method. The work in [65] ElasticFog uses a heuristic method to dynamically allocate computational resources based on real-time network traffic information for containerized applications in fog computing. This approach

TABLE 2: Existing related surveys

Survey	Year	Purpose
[8]	2022	Reviewing Kubernetes-based edge orchestration architectures for smart cities
[55]	2019	Discussing technologies and architectural frameworks that enable edge computing in the context of smart cities
[56]	2022	compares various fog computing platforms and their capabilities for different smart city use cases
[58]	2023	Review existing Cloud continuum orchestration frameworks, tools, and platforms, highlighting their strengths and limitations
[59]	2022	Introduce a taxonomy for Kubernetes scheduling, categorizing existing techniques, and identifying gaps in current research
[60]	2023	Explores various scheduling strategies and innovations in Kubernetes
[9]	2023	Classification of existing literature on custom Kubernetes schedulers on objectives, workload types, and environments
[61]	2024	The authors discuss Applications, challenges, architectures, and emerging trends in Fog/edge based smart cities
Our work	2025	we review custom Kubernetes scheduling strategies in edge environments.

is designed to improve throughput of user requests and reduce network latency compared to the default Kubernetes mechanism. The authors present a network-aware scheduler that deploys a service in the node that minimizes the Round Trip Time as long as it has enough bandwidth for the service. The proposed network-aware scheduling approach in Kubernetes for fog computing applications achieves reductions of 80% in network latency compared to the default scheduling mechanism [66]. In [67] Polaris scheduler leverages a cluster topology graph to make the scheduling decisions aware of the state of the network infrastructure for delay-sensitive services in edge computing environments, the scheduler is implemented as an extension to the default Kubernetes scheduling mechanism. The simulation results indicate that the average latency is reduced compared to the defaults Kubernetes scheduling mechanism. Similarly, [68]. a network- aware scheduling decisions is performed based on the perceived end-to-end latency between the end user and the service location, ensuring that optimal allocation is achieved through iterative adjustments. The node selection is based on the minimization of the nominal round trip time estimated on the basis of the target location for the service. The experiment results have shown that the latency experienced by the end-user is reduced in a finite time without degrading the quality of service.

Furthermore, we observe that several research studies have considered service requirements by including metrics such as response time and deadline [69], [70]. For example, the authors in [69] Suggest a cloud-edge deployment solution with a focus on service-aware scheduling for time-sensitive services. The scheduling algorithms make provisioning decisions by enabling the prioritization of critical services and the preemption on pods with lower priority with the main objective of meeting service deadline. The proposed architecture includes a continuous service monitoring component for observing the performance and state of services, ensuring that the service allocation remains optimal.They compared their custom scheduler to the default Kubernetes scheduler. The results have shown that the custom scheduler reveals a remarkable real-service allocation on the cluster nodes during the deployment process. Similarly, in [70] MicroFog is a Kubernetes-based framework designed to optimize the placement of containerized application across geo-distributed federated fog and cloud computing environments with main objective of minimizing service response time. They extended Kubernetes by implementing placement policies in both centralized and distributed modes. In the centralized mode, a primary Control Engine is deployed to handle the entire infrastructure. In contrast, the distributed mode involves multiple controller engines spread across clusters, each contributing to the processing of placement requests. The second mode shows that the performance of the application is improved compared to the first mode, by reducing service response times by up to 54%s.

D. MACHINE LEARNING-BASED METHODOLOGIES

RL-based scheduling frameworks are designed to optimize resource allocation, across distributed edge-cloud environments. In instance,

The authors in [71] proposed an RL-based deployment framework for latency-sensitive micro-services in Kubernetes multi-cluster environments. it incorporates a control-plane management system for deploying applications across multiple Kubernetes clusters and a global topology manager for managing the deployment across different computing layers. The framework leverages the DeepSets based PPO andd DQN algorithms to generalize learning policy across different cluster sizes without retraining. A latency-aware reward function is considered in the agent's decision to assign services to edge nodes. An edge node is selected if the latency constraints are met. If edge or fog nodes lack the necessary resources to handle a deployment request, the cloud is used as an alternative. The proposed framework employs a federation layer, . The results demonstrate that the proposed approach achieved high rewards for both strategies compared to A2C and maskable PPO algorithms. The authors in [72] present KaiS, a learning-based scheduling framework for Kubernetes-oriented edge-cloud networks. KaiS leverages the coordinated multi-agent actor-critic algorithm to handle request dispatch within edge clusters. KaiS computes a resource context for each agent. This context filters out invalid dispatch actions based on the current availability of resources at edge nodes to ensure that requests are only dispatched to nodes with sufficient resources. This approach allows the framework to adapt to varying request arrival patterns and system scales. The reward function for agents is designed to optimize the scheduling process by balancing load and improving the throughput rate (the ratio of requests completed within specified delay to the total number of requests). To simplify the orchestration resources across the network, kaiS uses GNNs with multiple policy networks to reduce the orchestration dimensionality. The also framework adopted a two-Time-Scale scheduling mechanism to harmonize request dispatch and service orchestration. KaiS can enhance the average system throughput rate by 15.9% while reducing scheduling cost by 38.4% compared to baselines. Topology awareness in the context of edge-cloud networks and service orchestration refers to the understanding and utilization of the network's structure and the relationships between its components to optimize performance.

The use of GNNs in the paper hilights the role of topology awareness in embedding system state information. GNN can capture the relationshipss and intereactions between nodes, which is essentizl for effective orchestration and request dispatch is a distrubuted netwok. [73] presents a reinforcement learning (RL)-based scheduling framework designed to optimize the cross-cluster scheduling of both AI-intensive and memory-intensive services across multiple Kubernetes clusters in End-Edge-Cloud environments and manages the heterogeneity of devices, tasks, and computing capabilities (GPU, NPU, TPU, FPGA). An adaptive Q-Learning algorithm is employed to make scheduling decisions, leveraging task characteristics and environmental constraints to minimize task completion and waiting times. The framework was evaluated on a real-world test-bed, with experimental results demonstrating its ability to maximize rewards and significantly reduce task completion times.

In [74] suggested RL based-edge-cloud Kubernetes multi-cluster deployment framework with a focus on RL-based scheduling for real-time micro-services in a vehicular fog computing network with purpose to ensure service availability in diffrent locations. The framework leverages a Deep Q-learning algorithm to predict the placement of micro-services. To ensure service availability and low latency, a given micro-service can be pushed to the Kubernetes cluster or cached at the RSU. The hungarian algorithm is used to solve the Vehicular Container Placement problem by efficiently assigning microservices to vehicles. The proposed framework uses a service mesh architecture to manage communication between multiple micro-services across different clusters. They leverage Hungarian algorithm to find micro-services mapping scheme in the vehicles network. The simulation results shown a significant improvement in response time for a request as well as the deployment time

This subsection presents a set of works where the proposed custom scheduler is not limited to the boundaries of a single cluster, but it is rather targeted towards multi-cluster scenarios. An example is the Kubernetes based-MARL scheduling framework proposed in Reference [75], where Multi-Agent reinforcement learning is designed as customized schedulers on Kubernetes to schedule distributed deep learning jobs in large-scale edge-cloud cluster. The framework utilizes hierarchical GNNs to encode the network topology with the goal of minimizing job completion time and reducing communication overhead in the cluster. The simulation results indicate that MARL can achieve better policy convergence and higher learning speed compared to single-agent reinforcement learning approaches. 20% n terms of the job completion time is achieved.

E. OTHER MECHANISMS

The authors in [76] propose a novel scheduling approach by combining two existing approaches, DRAGON and OASIS to propose a scheduler with auto-scaling capabilities for distributed deep learning models training. It aims to prioritize the scheduling of resource-intensive services deployed in Kubernetes environments. The scheduling framework leverages auto-scaling functions to dynamically adjusting the resources allocated to improve the speed of training and manage resources efficiently. Their methodology assigns weights to determine service priority, allowing the scheduler to allocate resources to more critical tasks with higher resource requirements, ensuring that critical tasks are completed faster. Research indicates that compared to the default Kubernetes scheduler, the results of the experiments show that the training speed is increased by over 26%. (optimizing resource usage, critical tasks are completed faster) (optimizing resource usage, critical tasks can be completed faster, highly suitable for services that requi efficient management of computational resources for traininig learning models)

VI. ADDRESSING CHALLENGES AND APPLICATIONS

In this section, we answer **RQ4.** *How do the different scheduling mechanisms address the above challenges?*

- *Network and latency considerations* Network-aware scheduling has been proposed by most of studies. These schedulers incorporating network topology, in real-time network conditions (e.g., proximity,latency, and bandwidth) and applications requirements (e.g., response time, deadline) into scheduling decisions to reduce communication delays and improve response times. By adapting to changing network conditions, these solutions ensure that latency-sensitive workloads are always placed on the optimal nodes, even the network state fluctuates. Multi-cluster scheduling are also employed, where each cluster has its own local scheduler making decisions based on the specific conditions of their edge environment. These localized scheduling reduces the need for communication with a central controller which minimizes latency and enables faster decision-making for latency-sensitive applications. Finally, predictive scheduling has been used to predict future latency patterns based on historical data and real-time inputs such as network traffic, user mobility. These algorithms ensure that latency-sensitive workloads are proactively placed in optimal locations, reducing delays caused by sudden changes in network conditions. Service Mesh technique has been proposed by several solutions to manage and optimize traffic between microservices. By optimizing traffic routing, the service mesh reduces communication latency between services, improving the overall performance of latency-sensitive applications. Edge caching technique has been also used, where data are cached at edge nodes, reducing the need to fetch data from distant cloud servers. The scheduler can prioritize placing workloads on nodes with cached data, enabling faster processing for latency-sensitive applications like video streaming or real-time analytics.

- *Load balancing* Deep reinforcement learning algorithm adapts scheduling based on current cluster load, reducing response times and improving load distribution in edge environments.
- *Node heterogeneity* .Several innovative approaches have been proposed to enhance Kubernetes' scheduling capabilities in heterogeneous environments [77] [78] [79]. Despite these advancements, there are still open issues in Kubernetes scheduling for heterogeneous environments. These include the need for more flexible and adaptive scheduling frameworks that can dynamically respond to changing conditions and the integration of more sophisticated AI techniques to further optimize resource allocation and performance. Continued research and development in this area are essential to fully leverage the potential of Kubernetes in diverse and complex computing environments.
- *Fault Tolerance*. Several studies propose custom scheduling algorithms that incorporate real-time data and environmental parameters to improve fault tolerance and performance in edge environments [80] [81] [82]. However, Current solutions do not fully address fault tolerance, which is essential for maintaining service reliability in smart city applications. Implementing proactive fault-tolerant systems, such as those using machine learning for fault prediction and stateful service migration, can significantly enhance the reliability of Kubernetes in edge settings. QoS-aware scheduling models that extend traditional fault-tolerant approaches can help meet the time constraints and reliability requirements of edge applications.

VII. OPEN ISSUES AND FUTURE RESEARCH DIRECTIONS

A. ARCHITECTURE

The efficient processing of massive data in the context of smart cities, while satisfying the requirements of their applications requires an efficient orchestration of the of edge-cloud continuum. Most of the reviewed solutions follow a centralized orchestration approach. This approach offers a number of key advantages, such as significantly reducing the complexity of implementation and offering consistent decision-making. However, the centralized orchestration approach raises significant drawbacks. For example, decision-making and updates from the central controller may introduce latency. It introduces a single point of failure, and lack of scalability as the number of nodes and tasks increases. A decentralized approach can address the limitations mentioned above [71] [83], e.g., local decision-making minimizes latency, as decisions are made closer to where tasks are executed, supporting real-time applications. For example, real-time traffic management systems can quickly adjust traffic lights based on local congestion without waiting for decisions from a centralized control. Additionally, the system can scale more easily, as local orchestrators operate independently or collaboratively without overwhelming the central orchestrator. Also the absence of a single point of failure improves reliability and availability. Data privacy can also be improved, as data remains locally, while only aggregated models can be shared.

B. HEURISTIC-BASED SCHEDULING TECHNIQUES

- **Low latency and reliability.** Heuristic scheduling has been investigated to minimize end-to-end latency. Several of research leverage rescheduling techniques to dynamically adapt to environment changes and dynamic workloads by reallocating resources in responses to these changes. This helps in reducing latency by ensuring that applications are placed on nodes that minimize the delay experienced by end-user and receive the necessary computational power to function effectively without delays. This is particularly important in smart cities where low latency is required, in particular, for those latency-tolerant applications include environmental monitoring applications, that can operate effectively even with intermittent connectivity or delayed data delivery such as . However, most of custom schedulers dot not adequately consider the stringent latency and reliability requirements of edge applications such as those involving Vehicle-to-vehicle communication to avoid collisions. Some solutions attempt to address these issues, but no real-world scenarios were provided.
- **Scalability.** Most research has been conducted on small clusters, which may not accurately reflect the challenges faced in larger, real-world deployments.The heuristic algorithms may face scalability issues. They might not efficiently handle large-scale environments, resulting in higher latency. Issues such as scalability concerns, and practical implementation need to be addressed to fully leverage rescheduling for real-time applications. This requires ongoing research to develop more scalable algorithms that can handle large-scale, real-world workloads without compromising on performance.

C. RL-BASED SCHEDULING TECHNIQUES

.

TABLE 3: Application deployment solutions in smart city based on Kubernetes

	Orchestration			Performance Metrics	Evaluation Method	Application	Contributions
ref	Scheduling Method	Offloading Model	Architecture				
[71]	RL	Edge-to-Edge Edge-to-Cloud	Decentralized	Latency	Simulation	IoT app.	Generalization of RL algorithms.
[62]	Heuristic	Edge-to-edge Edge-to-cloud	Centralized	Deployment cost	Experiments	Self-driving vehicle	Users moving is considered
[63]	Heuristic	Edge-to-edge Edge-to-cloud	Centralized	Latency	Simulation	Smart urban	Runtime state is considered
[62]	Heuristic	Edge-to-edge Edge-to-cloud	Centralized	Deployment cost	Experiments	Self-driving vehicle	Users moving is considered
[68]	Heuristic	Edge-to-edge	Centralized	Latency	Experiments	G. uS app	Migration is considered
[67]	Heuristic	Edge-to-edge	Centralized	Latency	Simulation	Smart traffic management	Network SLOs are considered
[72]	RL	Edge-to-Edge Edge-to-Cloud	Decentralized	Throughput	Experiments	Real workload	Adaptability to dynamic environments
[73]	RL	Edge-to-Edge Edge-to-Cloud	Decentralized	Completion time	Test-bed	Image recognition	Heterogeneity -aware scheduling
[74]	RL	Edge-to-Edge	Decentralized	Deployment time	Simulation	Self-driving Applications	Inter-services communication is considered
[84]	DRL	Edge-to-Edge	Centralized	Latency	Simulation	Real-world scenarios	Cluster upgrades is considered
[85]	TOPIS	Edge-to-Edge	Centralized	Latency Throughput	Simulation	HTTP user requests	Network-awareness
[86]	DRL	Edge-to-Edge	Centralized	Load balancing Response time	Simulation	Random workload	Network-awareness
[64]	Heuristic	Edge-to-Edge	Decentralized	QoS Cost	Testbed	Random workload	Proximity-awareness
[69]	Heuristic	Edge-to-Edge Edge-to-Cloud	Centralized	Latency Cost	Testbed	Experiment	Real-time services is considered
[76]	DRAGON	Edge-to-Edge	Centralized	Resource usage	Simulation	Random workload	Resource-intensive services is considered
[70]	Heuristic	Edge-to-fog	Centralized	Latency	Experiments	Smart Health Monitoring Traffic Routing	Distributed placement is considered

- **Multi-agent architecture.** RL-based approaches have been shown to adapt to varying request patterns and system scales and their ability to improve resource utilization and reduce load imbalance significantly compared to native Kubernetes schedulers. RL-based approaches approach like the coordinated multi-agent architecture has been shown how well suited to handle the dynamic and distributed nature of edge -cloud computing environments by allowing local decision-making and reducing the need for centralized control, thus improving response time. Moreover, it has shown its ability to scale effectively with the system size and adapt to diverse system structures as it doesn't rely on a single point of control. This scalability is essential for edge environments where the number of devices and tasks can vary significantly.
- **Multi-cluster Kubernetes Scheduling**. Several researchers employed multi-cluster scheduling approach for latency-sensitive applications placement. By distributing tasks across multiple clusters, researchers can achieve better load balancing and resource utilization also it helps placing tasks closer the data source, hence latency is reduced and the throughput workloads is improved. In addition, it can minimize QoS violations and improve system reliability by dynamically adjusting to workload demands and network conditions.
- **Evaluation method.** there is a lack of real-world data sets for training and evaluation. Most studies rely on synthetic or simulated datasets, which may not capture the complexities of real-world workloads.
  The combination of multi-cluster Kubernetes scheduling and multi-agent reinforcement learning may provide a powerful framework for managing real-time smart cities applications in dynamic, distributed environments. As mentioned here It enables efficient resource utilization, fault tolerance, cost optimization, and continuous adaptation to changing conditions, ensuring high-quality service delivery with minimal operational overhead.

VIII. CONCLUSION

This paper evaluated the most recent edge application deployment methodologies in Kubernetes. To perform our evaluation, we based our survey on the essential requirements of smart cities applications and edge environment. This paper contributes a state-of-the-art overview of established scheduling solutions based on Kubernetes for smart city applications that satisfy the requirements of both application and infrastructure.The survey highlights which requirements have been successfully addressed by existing solutions and identifies key issues that still remain unresolved.

REFERENCES

[1] S. Srirama, M. Adhikari, and S. Paul, "Application deployment using containers with auto-scaling for microservices in cloud environment," J. Netw. Comput. Appl., vol. 160, p. 102629, 2020.

[2] H. Zhao, S. Deng, Z. Liu, J. Yin, and S. Dustdar, "Distributed redundant placement for microservice-based applications at the edge," IEEE Transactions on Services Computing, vol. 15, pp. 1732–1745, 2019.

[3] D. Yu, Y. Jin, Y. Zhang, and X. Zheng, "A survey on security issues in services communication of microservices-enabled fog applications," Concurrency and Computation: Practice and Experience, vol. 31, 2019.

[4] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: A state-of-the-art review," IEEE Transactions on Cloud Computing, vol. 7, pp. 677–692, 2019.

[5] G. Ambrosino, G. B. Fioccola, R. Canonico, and G. Ventre, "Container mapping and its impact on performance in containerized cloud environments," 2020 IEEE International Conference on Service Oriented Systems Engineering (SOSE), pp. 57–64, 2020.

[6] "kubernetes Homepage," 2025. [Online]. Available: https://kubernetes.io/

[7] C. Carrión, "Kubernetes scheduling: Taxonomy, ongoing issues and challenges," ACM Computing Surveys, vol. 55, pp. 1 – 37, 2022.

[8] C. Huang, R. Lu, and K. R. Choo, "Vehicular fog computing: Architecture, use case, and security and forensic challenges," IEEE Communications Magazine, vol. 55, no. 11, pp. 105–111, Nov 2017.

[9] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," IEEE Wireless Communications, vol. 26, no. 1, pp. 87–93, 2019.

[10] A. Mishra, A. Jha, B. Appasani, and A. Paul, "Emerging technologies and design aspects of next generation cyber physical system with a smart city application perspective," International Journal of System Assurance Engineering and Management, vol. 20, no. 1, pp. 416–464, 2023.

[11] S. E. Bibri, J. Krogstie, A. Kaboli, and A. Alahi, "Smarter eco-cities and their leading-edge artificial intelligence of things solutions for environmental sustainability: A comprehensive systematic review," Environmental Science and Ecotechnology, vol. 19, p. 100330, 2024.

[12] I. Rafiq, "Iot applications and challenges in smart cities and services," The Journal of Engineering, vol. 2023, pp. –(0), April 2023. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/tje2.12262

[13] S. Ketu and P. Mishra, "A contemporary survey on iot based smart cities: Architecture, applications, and open issues," Wireless Personal Communications, vol. 2022, p. 2319–2367, April 2022. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/tje2.12262

[14] S. Khemakhem and L. Krichen, "A comprehensive survey on an iot-based smart public street lighting system application for smart cities," Franklin Open, p. 100142, 2024.

[15] F. AIT SALAHT, F. Desprez, and A. Lebre, "An overview of service placement problem in Fog and Edge Computing," Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, LYON, France, Research Report RR-9295, Oct. 2019. [Online]. Available: https://hal.inria.fr/hal-02313711

[16] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, and K. Ha, "The role of cloudlets in hostile environments," IEEE Pervasive Computing, vol. 12, no. 4, pp. 40–49, 2013.

[17] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," IEEE Communications Surveys & Tutorials, vol. 20, no. 1, pp. 416–464, 2018.

[18] M. Afzali, A. Mohammad Vali Samani, and H. R. Naji, "An efficient resource allocation of iot requests in hybrid fog–cloud environment," The Journal of Supercomputing, vol. 80, no. 4, pp. 4600–4624, 2024.

[19] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, "A job scheduling algorithm for delay and performance optimization in fog computing," Concurrency and Computation: Practice and Experience, vol. 32, no. 7, p. e5581, 2020, e5581 cpe.5581. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5581

[20] A. Kumar, A. Upadhyay, N. Mishra, S. Nath, K. R. Yadav, and G. Sharma, "7. privacy and security concerns in edge computing-based smart cities," 2022.

[21] A. Garg, A. Kumar, and A. Rastogi, "1. security and privacy of application of smart cities," 2024.

[22] W. Gao, O. Tavallaie, S. Chen, and A. Zomaya, "Federated learning as a service for hierarchical edge networks with heterogeneous models," 2024. [Online]. Available: https://arxiv.org/abs/2407.20573

[23] G. Karthick, G. Mapp, and J. Crowcroft, 9. Building an Intelligent Edge Environment to Provide Essential Services for Smart Cities, 2023.

[24] T. P. Da Silva, T. Batista, F. Lopes, A. R. Neto, F. C. Delicato, P. F. Pires, and A. R. Da Rocha, "Fog computing platforms for smart city applications: A survey," ACM Trans. Internet Technol., vol. 22, no. 4, Dec. 2022. [Online]. Available: https://doi.org/10.1145/3488585

[25] X. Niu, X. Cao, and C. Yu, "3. amspm: Adaptive model selection and partition mechanism for edge intelligence-driven 5g smart city with dynamic computing resources," ACM Transactions on Sensor Networks, 2024.

[26] J. Zhang, X. Zhang, and J. Zhang, "4. task-oriented communication for edge video analytics," IEEE Transactions on Wireless Communications, 2024.

[27] S. K. Das and P. Raj, "2. edge computing and analytics: A new computing paradigm for better user experience," 2024.

[28] Y. Zhang, R. Hou, and L. Liu, "6. research on security and privacy protection of edge computing based on blockchain technology," 2023.

[29] A. Kumar, A. Upadhyay, N. Mishra, S. Nath, K. R. Yadav, and G. Sharma, "7. privacy and security concerns in edge computing-based smart cities," 2022.

[30] "The Road to Autonomous Driving," TE connectivity GErmany GMbh, Tech. Rep., 01 2018.

[31] X. Li, T. Chen, Q. Cheng, S. Ma, and J. Ma, "Smart applications in edge computing: Overview on authentication and data security," IEEE INTERNET OF THINGS JOURNAL, vol. 8, no. 6, pp. 4063–4080, MAR 15 2021.

[32] H. Maleki, M. Basaran, and L. Durak-Ata, "Reinforcement learning-based decision-making for vehicular edge computing," in 29TH IEEE CONFERENCE ON SIGNAL PROCESSING AND COMMUNICATIONS APPLICATIONS (SIU 2021). IEEE; IEEE Turkey Sect, 2021, 29th IEEE Conference on Signal Processing and Communications Applications (SIU), ELECTR NETWORK, JUN 09-11, 2021.

[33] M. Zhu, X. Wang, and J. Hu, "Impact on car following behavior of a forward collision warning system with headway monitoring," Transportation Research Part C: Emerging Technologies, vol. 111, pp. 226–244, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X18318679

[34] S. Jun, Y. Kang, J.-H. Kim, and C. Kim, "Ultra-low-latency services in 5g systems: A perspective from 3gpp standards," 2020.

[35] J. Lin, W. Yu, X. Yang, P. Zhao, H. Zhang, and W. Zhao, "An edge computing based public vehicle system for smart transportation," IEEE Transactions on Vehicular Technology, vol. 69, no. 11, pp. 12 635–12 651, 2020.

[36] R. Ke, Z. Cui, Y. Chen, M. Zhu, H. Yang, and Y. Wang, "Edge computing for real-time near-crash detection for smart transportation applications," arXiv preprint arXiv:2008.00549, 2020.

[37] J. Powell, A. McCafferty-Leroux, W. Hilal, and S. A. Gadsden, "Smart grids: A comprehensive survey of challenges, industry applications, and future trends," Energy Reports, vol. 11, pp. 5760–5785, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484724003299

[38] O. Taiwo and A. E. Ezugwu, "Internet of things-based intelligent smart home control system," Security and Communication Networks, vol. 2021, pp. 1–17, 2021.

[39] E. Davies and V. Anireh, "Design and implementation of smart home system using internet of things," Advances in Multidisciplinary & Scientific Research Journal Publication, vol. 7, pp. 33–42, 2019.

[40] K. Purna Prakash, Y. V. P. Kumar, K. Ravindranath, G. Pradeep Reddy, M. Amir, and B. Khan, "Artificial neural network-based data imputation for handling anomalous energy consumption readings in smart homes," Energy Exploration & Exploitation, vol. 42, pp. 1432–1449, 2024.

[41] A. Qashlan, P. Nanda, X. He, and M. Mohanty, "Privacy-preserving mechanism in smart home using blockchain," IEEE Access, vol. 9, pp. 103 651–103 669, 2021.
[42] T. Anees, Q. Habib, A. S. Al-Shamayleh, W. Khalil, M. Obaidat, and A. Akhunzada, "The integration of wot and edge computing: issues and challenges," Sustainability, vol. 15, p. 5983, 2023.
[43] Y. Xiao and B. Liu, "Blockchain-based public audit scheme for smart home data integrity," 3rd International Conference on Internet of Things and Smart City (IoTSC 2023), 2023.
[44] S. T. Ahmed, V. Kumar, and J. Kim, "Aitel: ehealth augmented-intelligence-based telemedicine resource recommendation framework for iot devices in smart cities," IEEE Internet of Things Journal, vol. 10, pp. 18 461–18 468, 2023.
[45] J. Santos, C. Wang, T. Wauters, and F. de Turck, "Diktyo: Network-aware scheduling in container-based clouds," IEEE Transactions on Network and Service Management, vol. 20, pp. 4461–4477, 2023.
[46] S. Mondal, Z. Zheng, and Y. Cheng, "On the optimization of kubernetes toward the enhancement of cloud computing," Mathematics, 2024.
[47] W. Lai, Y.-C. Wang, and S.-C. Wei, "Delay-aware container scheduling in kubernetes," IEEE Internet of Things Journal, vol. 10, pp. 11 813–11 824, 2023.
[48] ——, "Delay-aware container scheduling in kubernetes," IEEE Internet of Things Journal, vol. 10, pp. 11 813–11 824, 2023.
[49] C. Chiaro, D. Monaco, A. Sacco, C. Casetti, and G. Marchetto, "Latency-aware scheduling in the cloud-edge continuum," NOMS 2024-2024 IEEE Network Operations and Management Symposium, pp. 1–5, 2024.
[50] M.-N. Tran, X. T. Vu, and Y. Kim, "Proactive stateful fault-tolerant system for kubernetes containerized services," IEEE Access, vol. 10, pp. 102 181–102 194, 2022.
[51] A. Javed, J. Robert, K. Heljanko, and K. Främling, "Iotef: A federated edge-cloud architecture for fault-tolerant iot applications," Journal of Grid Computing, vol. 18, pp. 57 – 80, 2020.
[52] X. Wang, K. Zhao, and B. Qin, "Optimization of task-scheduling strategy in edge kubernetes clusters based on deep reinforcement learning," Mathematics, 2023.
[53] Y. Sun, H. Xiang, Q. Ye, J. Yang, M. Xian, and H. Wang, "A review of kubernetes scheduling and load balancing methods," 2023 4th International Conference on Information Science, Parallel and Distributed Systems (ISPDS), pp. 284–290, 2023.
[54] R. Gao, X. Xie, and Q. Guo, "K-tahp: A kubernetes load balancing strategy base on topsis+ahp," IEEE Access, vol. 11, pp. 102 132–102 139, 2023.
[55] Q. D. La, M. V. Ngo, T. Q. Dinh, T. Q. Quek, and H. Shin, "Enabling intelligence in fog computing to achieve energy and latency reduction," Digital Communications and Networks, vol. 5, no. 1, pp. 3 – 9, 2019, artificial Intelligence for Future Wireless Communications and Networking. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864818301081
[56] M. Aazam, E.-N. Huh, and M. St-Hilaire, "Towards media inter-cloud standardization–evaluating impact of cloud storage heterogeneity," Journal of Grid Computing, vol. 16, no. 3, pp. 425–443, 2018.
[57] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1628–1656, 2017.
[58] T. N. Gia, M. Jiang, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fog computing in healthcare internet of things: A case study on ecg feature extraction," in 2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing. IEEE, 2015, pp. 356–363.
[59] Yu Cao, Songqing Chen, Peng Hou, and D. Brown, "Fast: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation," in 2015 IEEE International Conference on Networking, Architecture and Storage (NAS), Aug 2015, pp. 2–11.
[60] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Méndez, C. E. Chung, Y. Te Wang, T. Mullen, and T. P. Jung, "Augmented brain computer interaction based on fog computing and linked data," in 2014 International Conference on Intelligent Environments. IEEE, 2014, pp. 374–377.
[61] N. Daneshfar, N. Pappas, V. Polishchuk, and V. Angelakis, "Service allocation in a mobile fog infrastructure under availability and qos constraints," in 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 2018, pp. 1–6.
[62] S. Rac and M. Brorsson, "Cost-aware service placement and scheduling in the edge-cloud continuum," ACM Transactions on Architecture and Code Optimization, vol. 21, no. 2, pp. 1–24, 2024.
[63] A. Marchese and O. Tomarchio, "Orchestrating microservices-based applications inthe cloud-to-edge continuum," in Cloud Computing and Services Science, M. van Steen, D. Ferguson, and C. Pahl, Eds. Cham: Springer Nature Switzerland, 2024, pp. 170–187.
[64] K. Toczé, A. J. Fahs, G. Pierre, and S. Nadjm-Tehrani, "Violinn: Proximity-aware edge placementwith dynamic and elastic resource provisioning," ACM Transactions on Internet of Things, vol. 4, pp. 1 – 31, 2022.
[65] N. Nguyen, L.-A. Phan, D.-H. Park, S. Kim, and T. Kim, "Elasticfog: Elastic resource provisioning in container-based fog computing," IEEE Access, vol. 8, pp. 183 879–183 890, 2020.
[66] J. Santos, T. Wauters, B. Volckaert, and F. Turck, "Towards network-aware resource provisioning in kubernetes for fog computing applications," 2019 IEEE Conference on Network Softwarization (NetSoft), pp. 351–359, 2019.
[67] T. Pusztai, S. Nastic, A. Morichetta, V. C. Pujol, P. Raith, S. Dustdar, D. Vij, Y. Xiong, and Z. Zhang, "Polaris scheduler: Slo- and topology-aware microservices scheduling at the edge," in 2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC), 2022, pp. 61–70.
[68] C. Centofanti, W. Tiberti, A. Marotta, F. Graziosi, and D. Cassioli, "Taming latency at the edge: A user-aware service placement approach," Computer Networks, vol. 247, p. 110444, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128624002767
[69] R. Rosmaninho, D. Raposo, P. Rito, and S. Sargento, "Edge-cloud continuum orchestration of critical services: A smart-city approach," arXiv preprint arXiv:2407.17314, 2024.
[70] S. Pallewatta, V. Kostakos, and R. Buyya, "Microfog: A framework for scalable placement of microservices-based iot applications in federated fog environments," Journal of Systems and Software, vol. 209, p. 111910, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121223003059
[71] J. Santos, M. Zaccarini, F. Poltronieri, M. Tortonesi, C. Sleianelli, N. Di Cicco, and F. De Turck, "Efficient microservice deployment in kubernetes multi-clusters through reinforcement learning," in NOMS 2024-2024 IEEE Network Operations and Management Symposium, 2024, pp. 1–9.
[72] S. Shen, Y. Han, X. Wang, S. Wang, and V. C. M. Leung, "Collaborative learning-based scheduling for <italic>kubernetes</italic>-oriented edge-cloud network," IEEE/ACM Trans. Netw., vol. 31, no. 6, p. 2950–2964, Apr. 2023. [Online]. Available: https://doi.org/10.1109/TNET.2023.3267168
[73] W. Shen, W. Lin, W. Wu, H. Wu, and K. Li, "Reinforcement learning-based task scheduling for heterogeneous computing in end-edge-cloud environment," Cluster Computing, vol. 28, no. 3, p. 179, 2025.
[74] A. Nsouli, W. El-Hajj, and A. Mourad, "Reinforcement learning based scheme for on-demand vehicular fog formation," Vehicular Communications, vol. 40, p. 100571, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209623000013
[75] X. Zhao and C. Wu, "Large-scale machine learning cluster scheduling via multi-agent graph reinforcement learning," IEEE Transactions on Network and Service Management, vol. 19, pp. 4962–4974, 2021.
[76] M. F. Bestari, A. I. Kistijantoro, and A. B. Sasmita, "Dynamic resource scheduler for distributed deep learning training in kubernetes," in 2020 7th International Conference on Advance Informatics: Concepts, Theory and Applications (ICAICTA), 2020, pp. 1–6.
[77] W. Lai, Y.-C. Wang, and S.-C. Wei, "Delay-aware container scheduling in kubernetes," IEEE Internet of Things Journal, vol. 10, pp. 11 813–11 824, 2023.
[78] S. Yang, Y. Ren, J. Zhang, J. Guan, and B. Li, "Kubehice: Performance-aware container orchestration on heterogeneous-isa architectures in cloud-edge platforms," 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), pp. 81–91, 2021.
[79] I. Harichane, S. A. Makhlouf, and G. Belalem, "Kubesc-rtp: Smart scheduler for kubernetes platform on cpu-gpu heterogeneous systems," Concurrency and Computation: Practice and Experience, vol. 34, 2022.
[80] M.-N. Tran, X. T. Vu, and Y. Kim, "Proactive stateful fault-tolerant system for kubernetes containerized services," IEEE Access, vol. 10, pp. 102 181–102 194, 2022.
[81] L. Toka, "Ultra-reliable and low-latency computing in the edge with kubernetes," Journal of Grid Computing, vol. 19, 2021.

[82] W. Lai, Y.-C. Wang, and S.-C. Wei, “Delay-aware container scheduling in kubernetes,” IEEE Internet of Things Journal, vol. 10, pp. 11 813–11 824, 2023.

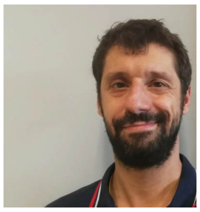
[83] P. Kokkinos, “Towards the realization of converged cloud, edge and networking infrastructures in smart megacities,” in 2022 IEEE Symposium on Computers and Communications (ISCC), 2022, pp. 1–5.

[84] H. Cui, Z. Tang, J. Lou, W. Jia, and W. Zhao, “Latency-aware container scheduling in edge cluster upgrades: A deep reinforcement learning approach,” IEEE Transactions on Services Computing, vol. 17, no. 5, pp. 2530–2543, 2024.

[85] Y. Qiao, J. Xiong, and Y. Zhao, “Network-aware container scheduling in edge computing,” Cluster Computing, vol. 28, no. 2, Nov. 2024. [Online]. Available: https://doi.org/10.1007/s10586-024-04733-8

[86] X. Wang, K. Zhao, and B. Qin, “Optimization of task-scheduling strategy in edge kubernetes clusters based on deep reinforcement learning,” Mathematics, vol. 11, no. 20, p. 4269, 2023.

KAOUTHER GASMI received her PhD in Communication Systems from Tunis El Manar University, Tunisia in 2018. She also holds an engineering degree in Computer networking obtained from the National Engineering School of Tunis, Tunisia. Her research interests primary focuses on smart cities, machine learning and cloud/edge computing. She is currently a postdoctoral researcher at the University of Modena and Reggio Emilia (UniMoRe) in Italy.



PAOLO BURGIO (Member, IEEE) received the PhD degree in electronics engineering jointly between the University of Bologna and the University of Southern-Brittany, in 2013. His research topics are next-generation predictable systems based on heterogeneous many-cores and GP-GPUs, with an eye on compilers, and parallel programming models. Since 2014 he joined HiPeRT Lab with Univ. of Modena, where he currently coordinates the activities on autonomous vehicles and drones, and smart cities. He is co-founder of the HiPeRT srl startup.

MARCO MAMEI received the M.S. degree in computer engineering from the University of Modena and Reggio Emilia, in 2001 and the Ph.D. degree in computer engineering from the same university in 2004. From 2019 he is full professor of computer engineering the the University of Modena and Reggio Emilia, He is the author more than 150 articles, and 8 patents. His research interests include Internet of Things systems and applications, data analysis and machine learning. He is currently involved in the projects: PR-FESR EMILIA ROMAGNA Project Support System for Sustainable Smart Cities - S4C, EU project “A catalyst for EuropeaN ClOUd Services in the era of data spaces, high-performance and edge computing(NOUS)” and EU project "Ecomobility".

• • •