Using the Compute Continuum for Data Analysis: Edge-cloud Integration for Urban Mobility

Loris Belcastro, Fabrizio Marozzo, Alessio Orsino, Domenico Talia, Paolo Trunfio
University of Calabria
Rende, Italy
{lbelcastro, fmarozzo, aorsino, talia, trunfio}@dimes.unical.it

***Abstract*—More and more in recent years, IT companies have adopted edge-cloud continuum solutions to efficiently perform analysis tasks on data generated by IoT devices. As an example, in the context of urban mobility, the use of edge solutions can be extremely effective in managing tasks that require real-time analysis and low response times, such as driver assistance, collision avoidance and traffic sign recognition. On the other hand, the integration with cloud systems can be convenient for tasks that require a lot of computing resources for accessing and analyzing big data collections, such as route calculations and targeted advertising. Designing and testing such hybrid edge-cloud architectures are still open issues due to their novelty, large scale, heterogeneity, and complexity. In this paper, we analyze how the compute continuum can be exploited for efficiently managing urban mobility tasks. In particular, we focus on a case study related to taxi fleets that need to find locations where they are more likely to find new passengers. Through a simulation-based approach, we demonstrate that these solutions turn out to be effective for this class of problems, especially as the number of connected vehicles increases.**

***Index Terms*—Edge-cloud architecture, IoT infrastructure, Edge computing, Urban computing, Smart cities, Urban mobility**

I. Introduction

Thanks to the large-scale availability of Internet-of-Things (IoT) devices, nowadays it is possible to collect large volumes of data from different sources, such as sensors, cameras, wearable devices and smartphones [1]. The characteristics of these data, such as large volume, high velocity, and format heterogeneity, make their collection, storage and analysis really challenging. For this reason, much effort has been made to develop new technologies, architectures and algorithms for extracting valuable information quickly and accurately [2].

In most cases, the applications used today for processing data from IoT devices are highly centralized and leverage cloud platforms to perform all major operations involving data collection, storage, processing, and analysis. However, using only the cloud may produce serious inefficiencies in terms of network traffic, latency times and optimization of energy consumption. These issues become particularly critical for some kinds of applications, such as those in the medical and security fields, where it is vital to have low-latency response times to avoid serious problems (e.g., fatal accidents) [3]. Because of this, researchers and IT companies have proposed the adoption of the edge computing paradigm for processing data closer to where it is generated, so as to achieve low latency, privacy preservation and scalability. These benefits can be complemented by those provided by the cloud, which allows to aggregate big data persistently and perform compute-intensive analyses using a large amount of computing resources, from which the concept of edge-cloud compute continuum [4].

Even in the context of urban mobility, the use of edge-cloud solutions can prove to be extremely effective in managing the different tasks that are generated. For example, tasks like driver assistance, collision avoidance and traffic sign recognition, which require real-time analysis and low response times, can benefit from edge computing. Differently, tasks like diagnostic data collection and analysis, route calculations and targeted advertising, which require a lot of computing resources and access to large datasets, can benefit from the use of cloud computing. However, designing and testing large-scale and multi-layer edge-cloud architectures are still open issues, especially for architectures composed of several components based on different technologies and software stacks [5]. For these reasons, simulation-based approaches result to be a powerful and flexible tool for reproducing and testing edge-cloud architectures, avoiding the risks, costs and failures associated with extensive field experimentation [6].

This paper analyzes how the compute continuum can be exploited for efficiently managing tasks related to urban mobility in large-scale computing environments. In particular, we focus on a case study related to taxi fleets that need to find locations where they are more likely to find new passengers. First, a detailed description of the application scenario is provided, in which geotagged data generated by taxis is analyzed by machine learning algorithms. Then, an edge-cloud continuum architecture is modeled to efficiently handle a large number of IoT devices and execute machine learning algorithms. The paper concludes with an experimental evaluation of different design choices (e.g. number of devices, task type, orchestration policies) in terms of processing time, network delay, task failure, and computational resource usage.

The structure of the paper is as follows. Section II discusses related work. Section III describes the edge-cloud continuum architecture. Section IV presents a case study and a performance evaluation by using two different orchestration policies. Finally, Section V concludes the paper.

II. Related work

The widespread availability of IoT devices has posed a challenge for big data analytics in IoT environments, especially for processing and analyzing huge amounts of heterogeneous data produced by such devices. These tasks require new methods and tools to extract knowledge and in particular machine learning algorithms to identify patterns and correlations in data [7], [8]. However, when these algorithms are run by devices with limited resources (e.g., memory, CPU, bandwidth, and energy power), it is crucial to guarantee a trade-off between performance (e.g., the accuracy of the learning model) and amount of resources required for computation [9]. In this scenario, the compute continuum has emerged as an efficient solution to process and analyze the data generated by IoT devices [10], by complementing edge resources with those provided by the cloud. However, it must be noticed that analyzing and validating these solutions would require a large number of tests on IoT environments using a large number of devices, which makes it an infeasible approach. As discussed by D'Angelo et al. [11], using simulation and modeling (M&S) techniques can overcome this issue. These techniques can help in designing and evaluating IoT environments by simulating the structure and behavior of real-world systems, thus making it possible to manage their complexity.

In recent years there has been a growing interest in exploiting simulation for the design of IoT solutions to support a large set of urban mobility applications, including those using geotagged data [12]. Due to the ubiquitous spread of end devices equipped with GPS trackers, location-based applications can benefit from the enormous amount of geotagged data produced. Those data, usually modeled as trajectories [13], permit to obtain valuable insights from mobility patterns for location-based services, such as traffic optimization and targeted advertising. As an example, analyzing trajectories generated by taxis to predict the next destination on a journey can reduce route costs and traffic jams in modern cities. In fact, unlike other public transports, such as buses or subways, taxis do not follow fixed routes but plan their routes once a passenger has been left [14]. In modern smart cities, taxis are equipped with GPS trackers, which allow real-time monitoring of vehicles in a specific area. Those geotagged data are usually sent to an operative center that will route user calls to the nearest taxi in order to reduce travel costs. In this scenario, trajectory analysis can be used to know in advance where a taxi will move after the end of a ride, by predicting where users will request the service, also known as the next destination prediction problem. In particular, this problem can be modeled as a short- or long-term trajectory prediction task, where the current coordinates are exploited to predict the next position or a complete route respectively [15]. The simplest way to predict the next destination is to split an area of interest into a grid of cells and predict the next visited cell based on the current one, thereby modeling the problem as a multiclass classification [16]. Several solutions have been proposed to predict the next position of a moving object. Most of them are based on frequent patterns and association rules and define a trajectory as an ordered sequence of locations [17]. Nevertheless, the need to extract meaningful patterns from large-scale and high-dimensional trajectory data has prompted the use of advanced analytical techniques, usually based on both supervised and unsupervised machine learning methods [18]. As an example, Rathore et al. [15] proposed a scalable clustering and Markov chain-based framework for trajectory prediction, which can handle a large number of trajectories in a dense road network, while Lin et al. [19] employed a neural network-based model.

In terms of tools and software solutions, different simulators of IoT environments have been proposed in recent years, such as iFog-Sim [20], IoTSim [21], FogNetSim++ [22], and EdgeCloudSim [23]. Among these, EdgeCloudSim is particularly suitable for modeling urban mobility scenarios, as it supports different architectures, devices, and mobility. For these reasons it was chosen for this study (see Section IV).

III. System Architecture

Although cloud computing provides high scalability with dynamic resource allocation, it may raise performance issues as a result of the centralization of data collection and processing [24], [25]. An edge-cloud continuum architecture might address these issues by enabling efficient and fast management of the massive volume of data generated by IoT devices. In particular, these architectures enhance computation capabilities and scalability while reducing network congestion and failed tasks. For these reasons, such architectures can also have a significant impact on urban mobility applications. Figure 1 shows a three-layer edge-cloud continuum architecture for supporting urban mobility. The edge-cloud continuum leverages all the resources from the edge of the network (i.e., IoT devices) to the core (i.e., cloud data centers) [26]. Specifically:

- At the *device layer* the IoT devices are leveraged by vehicles to share information during their movements across different urban cells. Such vehicles, including taxis, produce a very high volume of data through embedded modules, such as GPS trackers or infotainment devices. Those data are then sent to the edge server of the current cell.
- At the *edge layer* the devices that compose the edge infrastructure process data coming from the device layer as long as it has sufficient computing and storage resources. The edge layer may include heterogeneous hardware components (e.g., gateways, micro data centers), which serve as elements of the infrastructure that collect and partially process raw data generated at the device layer.
- At the *cloud layer* large sets of computing and storage resources can be dynamically allocated for executing tasks that cannot be performed by edge servers. From the perspective of the client, the cloud is an abstraction for remote, scalable provisioning of computation and storage resources. For these reasons, it has emerged as an effective computing paradigm to meet the challenge of processing big data in limited time, as well as to provide an efficient data analysis environment [27].

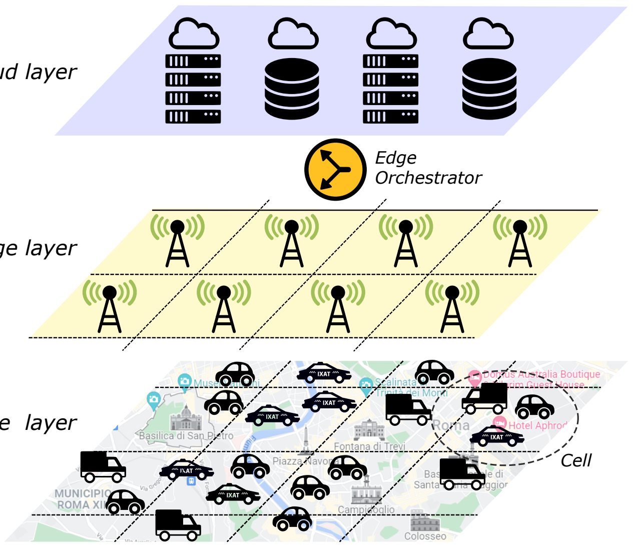


Fig. 1: The edge-cloud continuum architecture.

The edge layer includes a key component called *Edge Orchestrator (EO)*, which is responsible for managing and coordinating the execution of tasks, determining whether each task will run on the edge or cloud. It can be programmed to apply various orchestration policies in order to optimize the overall performance of the architecture, taking into consideration many parameters, such as network congestion, data volume to be processed, status and workload level of both edge nodes and cloud. Two orchestration policies were employed in this work, namely *network-based (edge/cloud-NB)* and *utilization-based (edge/cloud-UB)*, whose pseudocode is shown in Algorithm 1.

Algorithm 1 Edge Orchestrator

```
1: Initializing EO and orchestration policy $p$.
2: procedure GETSERVER($task, coord, \theta_1, \theta_2$)
3:     $cell \leftarrow getCell(coord)$
4:     $edgeS \leftarrow getEdgeServer(cell)$
5:     $layer \leftarrow null$
6:     if $p == Utilization\_Based$ then
7:         $edgeUtilization \leftarrow getEdgeUtilization()$
8:         if $edgeUtilization > \theta_1$ then
9:             $layer \leftarrow CLOUD$
10:        else
11:            $layer \leftarrow EDGE$
12:        end if
13:    else
14:        $wanDelay \leftarrow getUpDelay(task.getDevice(), CLOUD)$
15:        $wanUBW \leftarrow getBandwidthUtilization(wanDelay)$
16:        if $wanUBW < \theta_2$ then
17:            $layer \leftarrow CLOUD$
18:        else
19:            $layer \leftarrow EDGE$
20:        end if
21:    end if
22:    return $(layer == EDGE) ? edgeS : cloud$
23: end procedure
```

In particular, for each task to be scheduled, the cell and the associated edge server are identified from the coordinates of the IoT object generating that task (lines 3-4). Then, the desired orchestration policy (i.e. utilization-based or network-based) is applied to decide where the incoming task must be executed. Specifically, the utilization-based policy schedules tasks based on the utilization of edge nodes (lines 6-12). If the average edge utilization is greater than a fixed threshold (i.e., θ_1), the incoming task is offloaded to the cloud (lines 8-9); otherwise, it is assigned to the edge layer (lines 10-11). Instead, the network-based orchestration policy (lines 14-21) measures the network delay from the device that generated the task to the cloud (line 14), which is leveraged to determine the percentage of used bandwidth compared to the maximum bandwidth (line 15). If it is less than a fixed threshold (i.e., θ_2), the incoming task is offloaded to the cloud (lines 16-17); otherwise, it is assigned to the edge layer (lines 18-19). In the end, according to the chosen layer, the task is assigned to the cloud or to the edge server of the current cell (line 22). Therefore, the decision whether to offload a task to the cloud or perform it on the edge server is driven by two main parameters, i.e. the two thresholds θ_1 and θ_2. These thresholds can be chosen according to conventions often used on cloud platforms [28]–[30] to determine when to scale the computing resources (e.g., 80% of the total resources).

IV. Experimental evaluation

To evaluate the performance of the proposed edge-cloud continuum architecture for supporting urban mobility applications, we used the EdgeCloudSim simulator. It is a Java-based, open-source, and discrete event-based simulator designed for modeling IoT devices and applications, and edge-cloud continuum architectures. In particular, we considered three tasks:

- *Data collection*: it consists in collecting and preprocessing the data generated by taxis.

Parameter	Description	Value
Simulation time (min.)	Duration of the simulation in seconds.	36
Number of devices	Number of devices (i.e., taxis) used in the simulation scenario.	5k-12.5k
Edge servers	Number of edge servers.	100
MIPS for edge server VM	Computing processor's speed of edge servers.	2.5k
MIPS for cloud VM	Computing processor's speed of cloud.	300k
WLAN bandwidth (Mbps)	Bandwidth between devices and edge servers.	300
WAN bandwidth (Mbps)	Bandwidth between edge servers and the cloud.	150

TABLE I: EdgeCloudSim simulation parameters.

Parameter	Description	Task		
		Data Collection	Model Training	Prediction
Poisson interarrival (s)	Mean interarrival time between two tasks.	1k	10k	600
Active period (s)	The active period of the task.	10	500	5
Idle period (s)	The idle period of the task.	10	10	10
Upload data size (KB)	Mean input file sizes to upload.	200	1	200
Download data size (KB)	Mean output file sizes to download.	1	1	200
Task length (MI)	Mean number of instructions to execute the incoming task.	25k	60M	50k

TABLE II: Parameter values of the three tasks.

- *Model training*: it consists in training a machine learning model, which is periodically updated with new mobility patterns.
- *Prediction*: it exploits the trained model for suggesting the next destination where a taxi should move to find new passengers.

Table I reports the main simulation parameters used for configuring EdgeCloudSim, while Table II details the parameter values used for configuring the three tasks described in the previous. In particular, the three tasks are generated using a Poisson distribution with different active/idle task generation patterns and interarrival times. For what concerns the simulation parameters, the training and inference times together with the information on the hardware characteristics reported by Rathore et al. [15] have been used to determine the type of tasks and their average length. For each task, we modeled a set of information, including the Poisson interarrival, the active/idle period time of tasks, and the amount of data that is downloaded and uploaded. In particular, for data collection and prediction tasks modeling, a low Poisson interarrival value was set to reflect high device activity in the urban area. Conversely, a high value was chosen for the training task to represent its low frequency. Instead, the active and idle periods control the amount of time a device spends actively generating or not generating a specific task. For example, a high active period represents that the task is frequently generated by the device (i.e., a taxi frequently requests the next location). Finally, the upload and download data size controls the amount of data that is generated and transmitted by devices in the simulation. For example, large upload and download data sizes indicate a task with high data transmission requirements, such as the prediction and data collection tasks that involve significant data exchange.

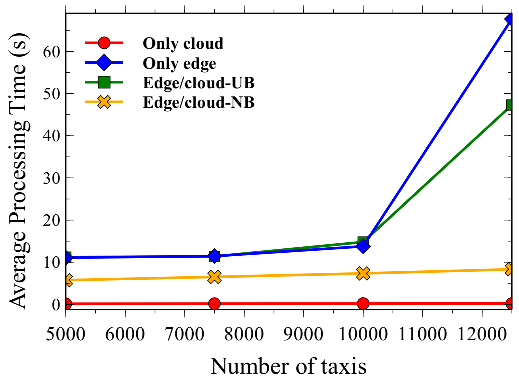
A large number of experiments have been carried out using an architecture composed of a cloud and 100 edge servers. Each edge server is assigned to a specific cell of the urban area and handles data generated by taxis that are located in that cell. Specifically, the cloud has been configured as a virtual machine (VM) equipped with 8 CPU cores, 32 GB of RAM and 1 TB of storage memory. Instead, each edge server has been configured as a VM having 4 cores, 4 GB of RAM and 64 GB of storage memory.

As explained above, the described architecture uses two orchestration policies, which are network-based (edge/cloud-NB) and utilization-based (edge/cloud-UB). In particular, for both orchestration policies the threshold values of Algorithm 1 were set at 80%, which means that the computing/network resources are preserved from being used no more than 80% of their capacity in order to avoid their saturation. Moreover, to evaluate the effectiveness of these orchestration policies, we compared them with a configuration in which data are processed entirely by the cloud (i.e., cloud-only) and another in which data are processed entirely by edge servers (i.e., edge-only). Four metrics were used to evaluate and compare the different configurations, i.e. the average processing time, percentage of failed tasks (i.e., tasks that are unable to be executed), network delay, and VM utilization.

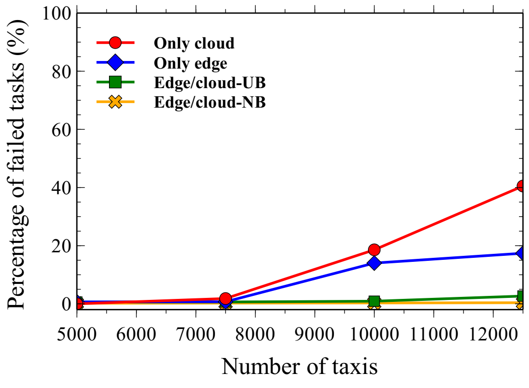
A. Performance evaluation

In this section, we present the main results we obtained. Figure 2 reports the performance metrics for each of the four configurations (cloud-only, edge-only, edge/cloud-NB, and edge/cloud-UB). In particular, the application is modeled to simulate the behavior of a taxi fleet in a city like Rome, which according to official data has about $10k$ taxi licenses [31]. However, we considered a variable number of taxis, ranging from $5k$ to $12.5k$, to investigate how a different number of taxis can impact the performance of the proposed architecture.

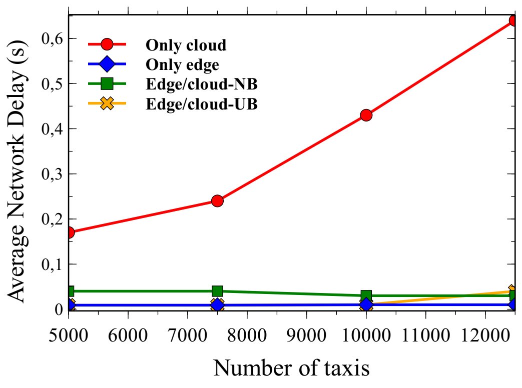
Figure 2(a) shows the average processing time obtained by the different configurations. In particular, the edge-only configuration showed the worst results, with a significant drop in performance as the number of vehicles increased (the



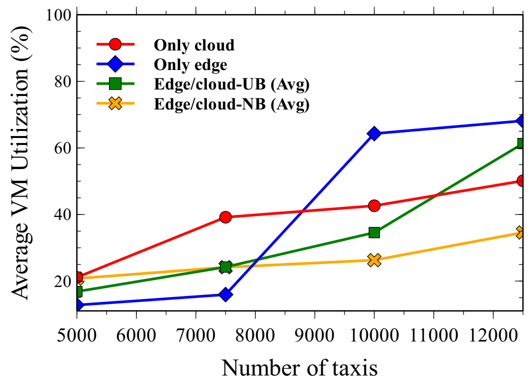
(a) Processing time.



(b) Failed tasks.



(c) Network delay.



(d) VM utilization.

Fig. 2: Performance results for the different configurations (cloud-only, edge-only, edge/cloud-UB and edge/cloud-NB).

processing time has increased from about 10 seconds with $5k$ vehicles up to about 70 seconds with $12.5k$ vehicles). In contrast, the cloud-only configuration achieved a very low average processing time but with a high number of failed tasks, which grows significantly as the number of vehicles increases. In fact, as shown in Figure 2(b), the percentage of failed tasks for the cloud- and edge-only configurations increases rapidly as the number of vehicles increases. In particular, a steep increase can be observed when using more than $7.5k$ vehicles: this means that, as long as there are few vehicles, the cloud-only configuration is able to handle the incoming workload better than other solutions, but as the number of devices increases it leads to a higher percentage of failed tasks (up to 56%). On the other hand, the edge-cloud solutions lead to a lower task failure rate (on average 6.8% for edge/cloud-NB and 1.3% for edge/cloud-UB). This is a crucial aspect to be considered, since in many contexts having a high number of failed tasks can compromise the usability of the IoT application. It is worth noting that a task can fail for three reasons: VM capacity, low network bandwidth or due to mobility. In particular, if the VM utilization is too high, incoming tasks may be rejected by any VM. Likewise, if too many vehicles connect to the same edge server, network congestion may occur and tasks may fail. Finally, a task may fail due to the vehicle movement from one cell to another. As an example, if we analyze the percentage of failed tasks in the cloud-only configuration, we find that only the 0.02% fails due to low computation capacity, while almost all failed tasks are due to network congestion. Among all the considered configurations, the edge/cloud-NB is able to balance data traffic between edge and cloud, avoiding sending traffic over the WAN when it is congested.

Figure 2(c) shows the average network delay. In particular, it emerges how the cloud-only configuration generates a very high network delay. In fact, data transfer from the edge to the cloud results in a significant increase of communication delay (up to around 98% higher than edge-only), while processing data locally at the edge does not produce significant effects.

Figure 2(d) illustrates the average VM utilization obtained by the different simulated configurations. The edge/cloud-NB achieved the best result showing a low utilization of resources, while keeping a low processing time and a low percentage of failed tasks. Reducing the use of VMs is a crucial aspect in large-scale applications that involve large computing resources, because it allows for optimizing costs and energy consumption. Additionally, reducing the risk of saturating computational resources allows to efficiently manage unexpected workload spikes that may occur. It should be also noted that the edge-only configuration produces a significant increase in the VM utilization for a high number of vehicles, but it still achieves a lower task failure rate than the cloud-only one (see Figure 2(b)). If we analyze in detail the percentage of VM utilization for the two edge-cloud continuum configurations, we can get more details about the behavior of the edge orchestrator. In particular, Figure 3 shows the percentage of VM utilization on both cloud and edge when we consider $12.5k$ taxis. The utilization-based policy results in a higher utilization of the edge resources (73% compared to 49% of cloud), while the network-based policy produces a higher utilization of cloud resources (57% compared to 12% of edge).

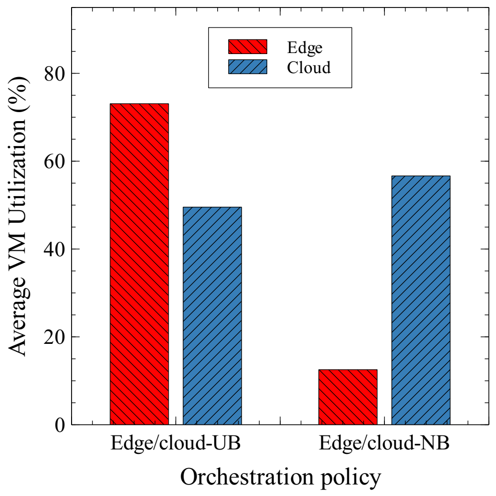


Fig. 3: Average VM utilization on both cloud and edge with the two orchestration policies for $12.5K$ taxis.

Overall, the edge/cloud-UB and edge/cloud-NB outperformed the conventional cloud- or edge-only configurations. Compared to the edge-only configuration, the edge orchestrator leads to a significant reduction in processing time, which ranges from 30% for edge/cloud-UB to 87% for edge/cloud-NB. In addition, compared to both cloud- and edge-only configurations, it permits to reduce the number of failed tasks (up to 38% for edge/cloud-UB and 40% for edge/cloud-NB) and the VM utilization (up to 29% for edge/cloud-UB and 38% for edge/cloud-NB).

V. Conclusions

With the widespread diffusion of IoT devices, edge-cloud continuum solutions have been proposed to combine the advantages of edge computing in processing data closer to where they are generated with those of the cloud in supporting compute-intensive tasks.

In this paper, we investigated the use of the edge-cloud continuum for supporting urban mobility applications in a large-scale environment. In particular, we focused on an application scenario that exploits geotagged data for predicting the next destination where taxis can find passengers. To assess the benefits of edge-cloud continuum over edge- and cloud-only, we used a simulation-based approach and two orchestration policies, based on network (edge/cloud-NB) and computational resources (edge/cloud-UB) utilization.

Through an experimental campaign, we demonstrated that the edge-cloud continuum architecture, coupled with the defined orchestration policies, outperforms traditional cloud- or edge-only architectures, obtaining a significant reduction in processing time, task failure rate, and resource utilization.

Future research efforts should be devoted to defining novel and more complex orchestration policies, which can exploit machine and deep reinforcement learning for improving task scheduling in the edge-cloud continuum. Such policies can be further evaluated using emulators instead of simulators to test how software interacts with underlying hardware.

References

[1] S. Madakam, V. Lake, V. Lake, V. Lake *et al.*, "Internet of things (iot): A literature review," *Journal of Computer and Communications*, vol. 3, no. 05, p. 164, 2015.

[2] L. Belcastro, R. Cantini, F. Marozzo, A. Orsino, D. Talia, and P. Trunfio, "Programming big data analysis: Principles and solutions," *Journal of Big Data*, vol. 9, no. 4, 2022.

[3] A. Barbieri, F. Marozzo, and C. Savaglio, "Iot platforms and services configuration through parameter sweep: a simulation-based approach," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 17-20 October 2021, pp. 1803–1808.

[4] F. Marozzo, A. Orsino, D. Talia, and P. Trunfio, "Edge computing solutions for distributed machine learning," in *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, 2022, pp. 1–8.

[5] M. Salama, Y. Elkhatib, and G. Blair, "Iotnetsim: A modelling and simulation platform for end-to-end iot services and networking," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, 2019, pp. 251–261.

[6] R. M. Fujimoto, A. W. Malik, A. Park *et al.*, "Parallel and distributed simulation in the cloud," *SCS M&S Magazine*, vol. 3, pp. 1–10, 2010.

[7] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, "Big iot data analytics: architecture, opportunities, and open research challenges," *ieee access*, vol. 5, pp. 5247–5261, 2017.

[8] E. Ahmed, I. Yaqoob, I. A. T. Hashem, I. Khan, A. I. A. Ahmed, M. Imran, and A. V. Vasilakos, "The role of big data analytics in internet of things," *Computer Networks*, vol. 129, pp. 459–471, 2017.

[9] S. Zahoor and R. N. Mir, "Resource management in pervasive internet of things: A survey," *Journal of King Saud University-Computer and Information Sciences*, 2018.

[10] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. DaSilva, C. Lee, and O. Rana, "The internet of things, fog and cloud continuum: Integration and challenges," *Internet of Things*, vol. 3, pp. 134–155, 2018.

[11] G. D'Angelo, S. Ferretti, and V. Ghini, "Simulation of the internet of things," in *2016 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2016, pp. 1–8.

[12] L. Belcastro, A. Falcone, A. Garro, and F. Marozzo, "Evaluation of large scale roi mining applications in edge computing environments," in *25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2021, pp. 1–8.

[13] E. Cesario, F. Marozzo, D. Talia, and P. Trunfio, "Sma4td: A social media analysis methodology for trajectory discovery in large-scale events," *Online Social Networks and Media*, vol. 3-4, pp. 49–62, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468696417300861

[14] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, "Where to find my next passenger," in *Proceedings of the 13th International Conference on Ubiquitous Computing*, ser. UbiComp '11. New York, NY, USA: ACM, 2011, pp. 109–118.

[15] P. Rathore, D. Kumar, S. Rajasegarar, M. Palaniswami, and J. C. Bezdek, "A scalable framework for trajectory prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3860–3874, 2019.

[16] A. Rossi, G. Barlacchi, M. Bianchini, and B. Lepri, "Modelling taxi drivers' behaviour for the next destination prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 7, pp. 2980–2989, 2019.

[17] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "Wherenext: a location predictor on trajectory pattern mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 637–646.

[18] E. Toch, B. Lerner, E. Ben-Zion, and I. Ben-Gal, "Analyzing large-scale human mobility data: a survey of machine learning methods and applications," *Knowledge and Information Systems*, vol. 58, no. 3, pp. 501–523, 2019.

[19] H. Lin, G. Liu, F. Li, and Y. Zuo, "Where to go? predicting next location in iot environment," *Frontiers of Computer Science*, vol. 15, no. 1, pp. 1–13, 2021.

[20] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[21] X. Zeng, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos, and R. Ranjan, "Iotsim: A simulator for analysing iot applications," *Journal of Systems Architecture*, vol. 72, pp. 93–107, 2017.

[22] T. Qayyum, A. W. Malik, M. A. K. Khattak, O. Khalid, and S. U. Khan, "Fognetsim++: A toolkit for modeling and simulation of distributed fog environment," *IEEE Access*, vol. 6, pp. 63 570–63 583, 2018.

[23] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, 2018.

[24] N. Khanghahi and R. Ravanmehr, "Cloud computing performance evaluation: issues and challenges," *Comput*, vol. 5, no. 1, pp. 29–41, 2013.

[25] V. K. Reddy, B. T. Rao, and L. Reddy, "Research issues in cloud computing," *Global Journal of Computer Science and Technology*, 2011.

[26] D. Rosendo, A. Costan, P. Valduriez, and G. Antoniu, "Distributed intelligence on the edge-to-cloud continuum: A systematic literature review," *Journal of Parallel and Distributed Computing*, 2022.

[27] L. Belcastro, F. Marozzo, D. Talia, and P. Trunfio, "Big data analysis on clouds," in *Handbook of Big Data Technologies*, A. Zomaya and S. Sakr, Eds. Springer, December 2017, pp. 101–142.

[28] M. K. Mohan Murthy, H. A. Sanjay, and J. Anand, "Threshold based auto scaling of virtual machines in cloud environment," in *Network and Parallel Computing*, C.-H. Hsu, X. Shi, and V. Salapura, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 247–256.

[29] "Amazon EC2 Auto Scaling FAQs," https://aws.amazon.com/ec2/autoscaling /faqs/, accessed January 2023.

[30] "Best practices for Autoscale - Microsoft Azure," https://learn.microsoft.com/en-us/azure/azure-monitor/autoscale/autosc-ale-best-practices, accessed January 2023.

[31] "Taxis in Rome," https://www.agenzia.roma.it/, accessed January 2023.