Estimating performances of Application Deployment on Distributed IoT-Edge-Cloud Infrastructures

Fabrizio Marozzo, Aleandro Presta
DIMES Department, *University of Calabria*
Rende, Italy
{fmarozzo, aleandro.presta}@dimes.unical.it

Rosa Varchera, Andrea Vinci†
ICAR-CNR National Research Council of Italy
Rende, Italy
{rosa.varchera, andrea.vinci}@icar.cnr.it

***Abstract*—Within the dynamic edge-cloud continuum, characterized by a lack of shared standards among platforms and vendors, application development is often dictated by the initial platform selection. Moreover, applications designed to harness the cloud for data analytics and storage, while relying on devices as data sources, face challenges in adapting to intermediate points along the edge-cloud spectrum. These issues are challenging the research community to propose novel solutions and methodologies aimed at aiding developers and integrators in realizing applications that can fully exploit the capabilities of the edge/cloud continuum, as well as choosing the most suitable edge infrastructure and cloud service ecosystem. The above research challenges are the main focus of the INSIDER research project. In this context, this paper aims to propose a formal model of the concepts and elements that compose applications in the edge/cloud continuum, and to show how this model could be used to estimate some performance indicators of possible deployment of applications in a cloud/edge computing infrastructure.**

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

I. Introduction

In recent years, the edge-cloud continuum computing paradigm has attracted significant interest from both the research and industrial communities. This interest is growing as cloud-edge integration promises to combine the high availability of computing resources in the cloud with the low latency and distributed resources available at the edge. The increasing amount of data generated by IoT devices necessitates local analysis directly on the device and at the edge to reduce latency and bandwidth usage. Simultaneously, remote analysis on the cloud is essential for more extensive processing and long-term storage, enabling advanced analytics and machine learning applications [14].

This work was supported by the research project "INSIDER: INtelligent ServIce Deployment for advanced cloud-Edge integRation" granted by the Italian Ministry of University and Research (MUR) within the PRIN 2022 program and European Union - Next Generation EU (grant n. 2022WWSCRR, CUP H53D23003670006), by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART"), and by Project "SoBigData.it – Strengthening the Italian RI for Social Mining and Big Data Analytics" founded by European Union – NextGenerationEU – National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) – Prot. IR0000013 – Avviso n. 3264 del 28/12/2021;

†Corresponding author.

Even if existing applications are generally conceived for cloud-based environments, solutions based on the edge-cloud continuum paradigm are being developed for, e.g., smart cities [5], cognitive buildings [3], and industrial environments [13], where there is the need to monitor, control, and perform real-time actions on IoT devices directly deployed in the environments while also storing a large amount of information and exploiting novel machine learning techniques.

The design of applications for an integrated edge-cloud computing infrastructure is challenging, as it should take into account not only the functionalities and requirements of an application but also the deployment infrastructure at the edge and the different services made available by different providers in the cloud, each usually having a different set of APIs [10]. Developers must consider factors such as data privacy, security, and compliance with regulatory standards, which can vary significantly between edge and cloud environments.

Choosing a suitable distributed application deployment on existing edge infrastructure and cloud services can directly impact the application's performance and execution cost. This selection process involves evaluating network latency, bandwidth availability, and the scalability of resources, making it a complex problem. Moreover, the dynamic nature of edge and cloud environments necessitates continuous monitoring and adaptation to maintain optimal performance and cost-efficiency [12].

In [9], we proposed a novel approach that uses a set of abstractions to conceptualize platform-independent applications. These applications are modeled as a composition of abstract services, which are independent of the place of distribution—whether edge or cloud—and specific platforms. This approach offers developers flexibility in the selection of services and providers during distribution, optimizing resource use and costs. This paper advances the research on deploying applications on edge/cloud continuum computing infrastructure by: (i) proposing a minimal formalism for mathematically modeling the concepts introduced in [9], focusing on abstract services, tasks, computing infrastructure, and deployment; and (ii) proposing a preliminary model, based on queue theory, to model the performance and utilization of a concrete computing node, and to approximate the expected response time of an instance of a task deployed on a node. These contributions could be useful for estimating performances of a possible ap-

plication deployment on an edge/cloud continuum computing infrastructure, and thus for selecting the best concrete solution for the realization of application in the edge/cloud continuum.

The rest of the paper is organized as follows: Section II briefly summarizes other studies in the context of assessing expected application performance in edge and cloud computing environments; Section III describe the reference framework and context where this study is originated; Section IV and V introduce a formalization for describing abstract applications, edge/cloud computing infrastructures, and deployments, according to the introduced framework and the design model presented in [9]. Section VI proposes a preliminary model, leveraging queue theory, for assessing computational nodes and tasks performances. Finally, Section VII concludes the work and draws future research avenues.

II. Related works

Estimating the performance of an application deployment requires identifying, developing, and exploiting performance metrics and standards. Addressing these challenges requires a clear understanding of the potential metrics that can be used in this optimization domain, for which developers are mostly struggling. The work in [4] presents a taxonomy of various real-world metrics to assess the performance of cloud, fog, and edge computing. In an edge infrastructure, we can identify two main categories: infrastructure parameters and QoS. In the survey in [16], a benchmarking-based methodology is presented that measures qualitative characteristics such as I/O throughput, end-to-end communication, or computation latency. These benchmarks capture performance metrics at the system infrastructure level, such as CPU, memory, network, storage. Kun Cao et al.'s survey [6] focus on optimizing QoS-related parameters in terms of service latency, energy consumption, security, privacy, and reliability of edge-cloud computing components. The possible application scenarios on edge-cloud infrastructures are many and the best solution depends on the application context. The work [15] presents an algorithm based on PSO (particle swarm optimization) to minimize time and costs in a business workflow of a manufacturing chain, through the composition of production services. In our use case, the reference infrastructure is the edge-cloud, where the services are tasks that perform computational calculation, with completely different scheduling parameters. Amadeo et al. [2] propose a novel strategy for the placement of computation tasks with reusable cache-based outputs in an SDN network domain. Their algorithm aims to minimize the utilization of network resources during the selection of edge nodes. Tasks are scheduled as independent autonomous tasks that do not communicate and coordinate with each other as in workflows. The papers in [17] and [1] exploit edge computing for analyzing real-time video streams, the former based on GPU scheduling and the latter on filtering and deep learning. Both aim to improve response time and throughput but do not handle distributed workflow-based applications.

III. Reference framework for developing applications in the cloud/edge continuum

This section introduces the overall reference framework that originated this work, and that was developed within the context of the INSIDER research project[1].

The framework aims at introducing a set of abstractions to conceptualize applications for the edge-cloud continuum, making them independent of any specific cloud platform. It allows defining applications as collections of abstract services that are not dependent on the deployment layer (e.g., cloud or edge) or particular cloud solutions, whether public (such as AWS or Microsoft Azure) or private (such as OpenStack or OpenNebula). By differentiating between services and applying optimization procedures, the framework enables the selection of a suitable set of services for implementation while ensuring adherence to Quality of Service (QoS), data locality, and security constraints. The conceptual model identifies interactions between abstract services within an application and provides clear guidelines for implementing various tasks—such as data collection, data transformation, and model training—with concrete services while respecting the functional constraints. Figure 1 illustrates the framework's main components:

- *Execution Infrastructure*: This component models cloud and edge execution infrastructures, considering technologies, hardware types, and network topology. It uses dynamic parameters like virtual machine sizes, number of devices, on-premises resources, and available execution layers (cloud, far edge, near edge, on-premise, or on-device)
- *QoS Constraints*: This defines a taxonomy of constraints, such as latency, energy consumption, and reliability, that can be used in the application model to meet performance objectives.
- *Abstract Application*: This represents applications as compositions of services with associated tasks and execution dependencies, modeling both functional and non-functional requirements, including QoS constraints (e.g., latency, energy consumption, message exchange, persistence, and reliability).
- *Cloud Service Catalog*: This includes service catalogs from various cloud providers (e.g., AWS, Microsoft Azure, and Google Cloud), enabling the exploration of services offered by each to identify those that meet application requirements.
- *Deployment Configuration Generator*: This component builds deployment configurations for an abstract application, given QoS constraints, the execution infrastructure, and cloud service catalogs.
- *Deployment Configuration Evaluator*: This defines an automated approach for evaluating application deployment configurations by analyzing performance parameters such as latency, energy consumption, network traffic, and task failure rates.

[1] https://scalab.dimes.unical.it/projects/insider/

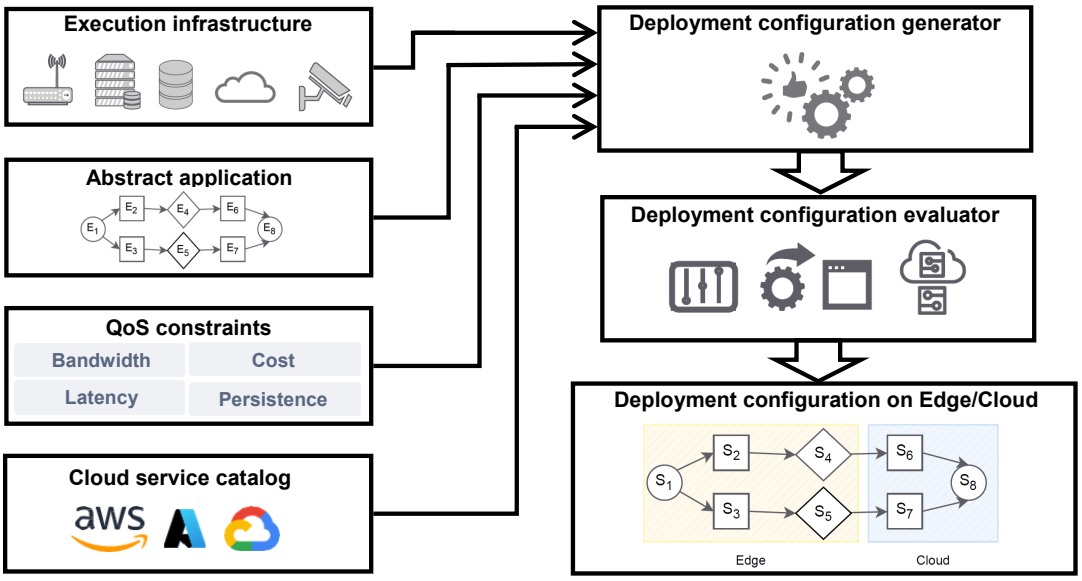

Fig. 1. The INSIDER framework.

- *Deployment Configurations on Cloud Platforms*: After identifying suitable configurations, this component aids developers in deploying the selected configuration on the chosen cloud platform.

Overall, the framework facilitates a systematic approach to modeling and deploying applications efficiently across diverse cloud and edge infrastructures, ensuring that QoS constraints are met and fully leveraging the edge-cloud continuum.

IV. Application Design Methodology

Abstract Service as a Workflow. In the context of the above-proposed framework, and according to the design methodology presented in [9], an edge-cloud distributed application is represented by abstract services. Abstract services are the template for applications that can be executed in a distributed manner on the Cloud-Edge Continuum. An abstract service is created by composing and linking tasks in the form of a workflow and is represented as an oriented graph [11]. Each abstract service and task is characterized by functional and non-functional requirements, and supplemented by quality of service (QoS) metrics that measure operational efficiency.

The realization of a concrete service starting from an abstract one consists of the selection of edge nodes and cloud services that will concretely perform each defined task, according to user defined QoS thresholds, such as total execution time and throughput.

The tasks that form an abstract service are software components with specific functionalities. Our approach defines three main task categories [9].

The first category is the *computation task*, which represents a component primarily responsible for processing input data and producing output. To this type belong various software that provides functions for data processing, training, and use of models, as well as encapsulation and integration with third-party services. The infrastructure requirements for computation tasks mainly include CPU, GPU, and RAM specifications. While quality of service (QoS) metrics mainly include reliability and fault tolerance. Infrastructural requirements and metrics make it possible to calculate the execution time for each request and the volume of data processed within a given period with good approximation.

The second category is the *communication task*, responsible for communication and information exchange between the other components. For an abstract service to run on a distributed infrastructure such as the Cloud-Edge continuum, it is essential to model and manage the communication times between the different components. Indeed, knowing the communication times between two tasks assigned to two cloud-edge nodes is necessary to calculate parameters such as the total execution time of the concrete service. A communication task can be characterized by requirements about latency, throughput, reliability, and security [7].

The third category is the *storage task*, which is the component responsible for managing data persistence. Various systems can concretize these tasks, such as relational databases, NoSQL databases, distributed file systems, network storage, etc. Requirements for a storage task include specifications related to storage capacity, read and write query execution times, persistence, consistency, and other relevant factors.

Given the depicted context, we can propose a preliminary mathematical model for abstract services, tasks, and computing infrastructure as follows.

Abstract service. An *abstract service* $\mathcal{A}$ is a directed acyclic graph $\langle \mathcal{T}, \mathcal{E} \rangle$, where $\mathcal{T}$ is the set of tasks and $\mathcal{E}$ is the set of oriented edges (t_i, t_j) between tasks, meaning that t_i is required to be executed prior t_j for a specific request.

Each *task* $t_i \in \mathcal{T}$ is characterized by its category, be it

computational (*com*), communication (*net*), or storage (*sto*). We define $\mathcal{T}^{com}$, $\mathcal{T}^{sto}$ and $\mathcal{T}^{net}$, respectively, as the set of computational tasks, storage tasks, and communication tasks of the considered abstract service, and $\mathcal{T} = \mathcal{T}^{com} \cup \mathcal{T}^{sto} \cup \mathcal{T}^{net}$. Each edge (t_i, t_j) is such that $t_i \in \mathcal{T}^{net} \oplus t_j \in \mathcal{T}^{net}$, and thus communication tasks interleave computation and storage tasks in an abstract service.

We characterize task requirements as follows:

- $op(t)$ identifies the expected number of operations that should be executed for performing an instance of a computational task $t \in \mathcal{T}^{com}$;
- $readOp(t)$ and $writeOp(t)$ identify, respectively, the expected volume of data that should be read or written to perform an instance of a storage task $t \in \mathcal{T}^{sto}$;
- $dataSize(t)$ identifies the expected volume of data that should be transferred to perform an instance of a communication task $t \in \mathcal{T}^{net}$;
- $cap(t)$ identify a set of capabilities that are required to perform a generic task $t \in \mathcal{T}$

Computing infrastructure. A computing infrastructure $\mathcal{CI}$ is defined as the set $\mathcal{N}$ of the possible edge-cloud computing nodes that can host or perform tasks. Here, a node $n \in \mathcal{N}$ can represent a physical edge device, a virtual machine hosted in the cloud, or a specific cloud service provided by public vendors. We distinguish two possible kinds of infrastructural nodes, that are, computing nodes and storage nodes, identified respectively by the sets $\mathcal{N}^{com}$ and $\mathcal{N}^{sto}$, such that $\mathcal{N}^{com} \cup \mathcal{N}^{sto} = \mathcal{N}$.

We characterize node capabilities as follows:

- $compOps(n)$ identifies the operation frequency of a node $n \in \mathcal{N}^{com}$;
- $readOps(n)$ and $writeOps(n)$ identify the speed of data read/write of a node $n \in \mathcal{N}^{sto}$;
- $bup(n)$ and $bdown(n)$ identify the upload/download bandwidth of a node $n \in \mathcal{N}$;
- $cap(n)$ identifies a set of capabilities that are available or characterize a node $n \in \mathcal{N}$

V. Application Deployment

A deployment of an abstract service on a computing infrastructure maps the abstract service's tasks to the computing infrastructure's nodes, assigning each computational or storage task to exactly one node or service.

We can now define the boolean variables $x_{ij} \in \{0, 1\}$, where $x_{ij} = 1$ means that the task $t_i \in \mathcal{T}^{com} \cup \mathcal{T}^{sto}$ is assigned to the computing infrastructure node $n_j \in \mathcal{N}$. Since communication tasks are interleaved between two computation or storage tasks, deploying a communication task requires involvement from two computing nodes: a source node and a destination node. For each communication task t_n we define two assignment variables $s_{ij}, d_{ij} \in \{0, 1\}$. s_{ij} represents that the communication task t_i starts on the node n_j, while d_{ij} means that the communication task t_i ends on the node n_j.

Thus, a deployment of an Abstract Service on a given Computing Infrastructure is represented by an assignment of the x, s, and d variables.

A valid deployment is admissible if each computation or storage task is assigned to exactly one node within the edge/cloud computing infrastructure. In contrast, a communication task must involve exactly two nodes. Thus, a valid deployment should respect the following constraints:

$$\sum_{j \in \mathcal{N}} x_{ij} = 1, \quad \forall t_i \in \mathcal{T}^{com} \cup \mathcal{T}^{sto} \tag{1}$$

$$\sum_{j \in \mathcal{N}} s_{kj} = 1, \quad \forall t_k \in \mathcal{T}^{net} \tag{2}$$

$$\sum_{j \in \mathcal{N}} d_{kj} = 1, \quad \forall t_k \in \mathcal{T}^{net} \tag{3}$$

A valid deployment should also consider the abstract service graph topology, so as to guarantee that communication tasks linking two computation or storage tasks insist on the same nodes hosting the two tasks, and thus should respect the following:

$$\sum_{j \in \mathcal{N}} x_{ij} s_{kj} = 1, \quad \forall (t_i, t_k) \in \mathcal{E}, t_k \in \mathcal{T}^{net}, t_i \in \mathcal{T}^{com} \cup \mathcal{T}^{sto} \tag{4}$$

$$\sum_{j \in \mathcal{N}} x_{ij} d_{kj} = 1, \quad \forall (t_j, t_i) \in \mathcal{E}, t_k \in \mathcal{T}^{net}, t_i \in \mathcal{T}^{com} \cup \mathcal{T}^{sto} \tag{5}$$

Finally, since each task t requires a set of capabilities for its execution, and each node can provide a set of capabilities, a valid deployment should also assign each task t to a node that provides the capabilities required for the task execution, thus:

$$cap(t_j) \subseteq cap(n_j), \quad \forall x_{ij} = 1 \tag{6}$$

VI. Estimating deployment performances

Given the concepts of abstract service, computing infrastructure, and deployment, we propose a preliminary analytic model for estimating the expected service time of task instances, by leveraging queuing theory concepts. This can be used, e.g., to compare two deployments or to evaluate possible computing infrastructures given an abstract service.

1) Computational tasks: Given a deployment $\{x_{ij}\}$, we model the computation on each node n_j as a M/H/1 queue, where the service policy is FCFS, the task instances arrivals times are distributed with an exponential probability function and the service times are distributed by a hyper-exponential function [8], depending by the probability of executing an instance of a specific task and the related expected service time.

For this purpose, we introduce the following parameters:

- $\mu_{ij} = \frac{ops(n_j)}{op(t_i)}$ is the expected service rate for an instance of the task t_i if executed on the node n_j. The service time of a single task instance on a node is modeled through an exponential distribution, and thus $\frac{1}{\mu_{ij}}$ is the expected service time of an instance of the task t_i on the node n_j;
- λ_{t_i} is the arrival rate of instances of task t_i,

- $\lambda_{n_j} = \sum_{\forall i | x_{ij}=1} \lambda_{t_i}$ is the arrival rate of task instances to the node n_j,
- $p_{ij} = \frac{\lambda_{t_i}}{\sum_{\forall t_k | x_{kj}=1} \lambda_{t_k}}$ is the probability of observing node n_j executing a task t_i in a random time.

As we model the task execution time of a computing node serving a set of tasks as distributed as an hyper-exponential random variable, when considering a steady-state system the following equations hold.

- The expected response time for an instance of a task in n_j, denoted as $W^e_{n_j}$, could be computed as:

$$W^e_{n_j} = \sum_{\forall i | x_{ij}=1} \frac{p_{ij}}{\mu_{ij}} \tag{7}$$

- The utilization factor of n_j is computed as:

$$\rho_{n_j} = \frac{\lambda^{tot}_{n_j}}{\mu_{n_j}} \tag{8}$$

  where $\mu_{n_j} = \frac{1}{W^e_{n_j}}$.
- the expected number of tasks waiting in queue of n_j, according to the Kingman's formula [8]:

$$L^q_{n_j} = (\frac{\rho_{n_j}}{1-\rho_{n_j}})(\frac{c_a^2 + c_s^2}{2}) \tag{9}$$

  where c_a and c_s are the coefficients of variations of the task interarrival times and the node service time.
- and thus, the total expected queue time for each task instance on the node n_j is:

$$W^q_{n_j} = L^q_{n_j} W^e_{n_j}, \tag{10}$$

  which is the expected execution time of a task instance multiplied by the expected number of tasks in the queue waiting for service.

The total response time of an instance of a computational task t_i deployed on a node n_j is then:

$$W^{com}_{ij} = (\frac{1}{\mu_{ij}} + W^q_{n_j}) \tag{11}$$

which is the expected service time for task t_i on node n_j plus the expected queuing time on node n_j. All of this holds if the queue is in its steady state, and if the utilization of the node n_j is less than one, i.e., $\rho_{n_j} < 1$, otherwise the service queue could grow indefinitely.

2) Storage tasks: Storage tasks model data persistence operations, which primarily involve data storage and retrieval. Optimizing data storage and retrieval in edge-cloud computing can present several challenges. Key issues include ensuring data consistency across distributed systems, minimizing latency in data retrieval, and managing bandwidth between edge devices and the cloud. Additionally, maintaining data security and privacy, especially when transferring sensitive information, is crucial. Efficient data indexing and caching strategies can help mitigate some of these challenges.

We model utilization of storage nodes and execution times of storage tasks instances following the same assumptions made above for the computational tasks, and thus with M/H/1 queues, where

- $\mu_{ij} = \frac{readOps(n_j)}{readOp(t_i)} + \frac{writeOps(n_j)}{writeOp(t_i)}$ is the expected service rate for a sequence of an instance of a storage task t_i if executed on the storage node n_j. The service time of a single storage task instance on a node is modeled through an exponential distribution, and thus $\frac{1}{\mu_{ij}}$ is the expected service time of an instance of the task t_i on the node n_j.
- λ_{t_i}, λ_{n_j}, p_{ij}, $L^q_{n_j}$, ρ_{n_j}, and $W^q_{n_j}$ are defined and computed similarly to what described in Section VI-1.
- The total response time for a storage task instance for task t_i deployed on a node n_j is thus:

$$W^{sto}_{ij} = (\frac{1}{\mu_{ij}} + W^q_{n_j}) \tag{12}$$

  which is the expected service time for storage task t_i on storage node n_j plus the expected queuing time on node n_j. All of this holds if the queue is in its steady state, and if the utilization of the node n_j is less than one, i.e., $\rho_{n_j} < 1$, otherwise the service queue could grow indefinitely.

3) Networking tasks: Communication Task (net) transfers the output data (expressed in bytes) of the computation task t_i to the next task t_k in the workflow. Thus, the deployment of a communication task involves two edge/cloud nodes, a source node and a destination node, which are the nodes hosting the previous and next non-communication tasks. From Section IV, we here recall that each computing node is characterized by upload and download bandwidth, denoted as $bup(n_j)$ and $bdown(n_j)$. Thus, for each node, we can model two queues, an inbound queue and an outbound queue, each characterized by its bandwidth. Thus, we can model each queue as M/H/1 with FIFO service policy, under the same assumption made for the queues related to the computation and storage tasks.

Given a source node n_s, we model the corresponding outbound queue with the following parameters:

- $\mu_{is} = \frac{bup(n_s)}{dataSize(t_i)}$ is the outbound communication rate considering a sequence of communication tasks t_i sourcing from n_s, and $\frac{1}{\mu_{is}}$ is the expected communication time of the task t_i when sourcing from the node n_S.
- λ_{t_i}, λ_{n_s}, p_{is},ρ_{n_s}, $L^q_{n_s}$, and $W^q_{n_s}$ are defined similarly as in Section VI-1.
- the expected delay introduced by the outbound queue of the node n_s for a communication task t_i is then:

$$W^{out}_{i,s} = \frac{1}{\mu_{i,s}} + W^q_{n_s} \tag{13}$$

The same can be modeled for the inbound queue of the destination, and thus:

- $\mu_{id} = \frac{bdown(n_d)}{dataSize(t_i)}$ is the inbound communication rate considering a sequence of communication tasks t_i incoming on node n_d, and $\frac{1}{\mu_{id}}$ is the expected communication time of the task t_i when dispatched to the node n_d.
- λ_{t_i}, λ_{n_d}, p_{id}, ρ_{n_d}, $L^q_{n_d}$, and $W^q_{n_d}$ are defined similarly as in Section VI-1.

- the expected delay introduced by the inbound queue of the node n_d for a communication task t_i is then:

$$W_{i,d}^{in} = \frac{1}{\mu_{i,d}} + W_{n_d}^{q} \tag{14}$$

Now, given an instance of a communication task t_i insisting on the source node n_s and the destination node n_d, we can approximate the communication time as the maximum of the expected delays introduced by the related outbound and inbound queue, that is, the expected communication instance delay is bounded by the slowest queue, thus:

$$W_{isd}^{net} = \max((\frac{1}{\mu_{is}} + W_{n_s}^{q}), (\frac{1}{\mu_{id}} + W_{n_d}^{q})) \tag{15}$$

VII. Discussion and conclusion

During the development of applications that use the edge/cloud continuum computing paradigm, it is crucial to design the application generically without considering a specific implementation provider from the start. This approach helps developers identify potential issues and select the most suitable cloud/edge service provider before the implementation phase.

Positioned within the framework of the INSIDER research project, this paper contributes on two main issues, that are, how to model abstract applications and edge/cloud continuum computing infrastructures, and how to estimate the expected performances of possible deployments, in terms of computing resources utilization and expected response time of task instances required for the application execution.

About application modeling, we proposed the use of workflows, and we recognize the need for categorize and classify application tasks to better characterize them (requirements and functionalities), in order to match them with the capabilities offered by the nodes and services of the edge/cloud computing infrastructure. For the performance assessment of possible application deployment, we applied to this context concepts of queueing theory, that effectively model the behavior of a deployment in terms of resources utilization and response time of tasks.

Ongoing work is devoted to: (i) extend the application, infrastructure, and deployment modeling in order to capture explicitly other requirements, e.g., expected volatile memory utilization, processors capabilities, execution costs, and carbon footprint; (ii) evaluating formalism such as Petri networks to enhance deployment modeling and deployment performance estimation; (iii) developing the other components introduced in the INSIDER framework, that is, in Figure 1; (iv) propose novel solutions leveraging machine learning, artificial intelligence, or quantum computing for finding the best suitable deployment for a given application/computing infrastructure; (v) assessing the effectiveness of the framework in real scenarios, through case studies on, e.g., smart city, cognitive buildings and industry 4.0.

References

[1] Muhammad Ali, Ashiq Anjum, Omer Rana, Ali Reza Zamani, Daniel Balouek-Thomert, and Manish Parashar. Res: Real-time video stream analytics using edge enhanced clouds. *IEEE Transactions on Cloud Computing*, 10(2):792–804, 2020.

[2] Marica Amadeo, Claudia Campolo, Gianmarco Lia, Antonella Molinaro, and Giuseppe Ruggeri. In-network placement of reusable computing tasks in an sdn-based network edge. *IEEE Transactions on Mobile Computing*, 2023.

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

[4] Mohammad S Aslanpour, Sukhpal Singh Gill, and Adel N Toosi. Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet of Things*, 12:100273, 2020.

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

[6] Kun Cao, Shiyan Hu, Yang Shi, Armando Walter Colombo, Stamatis Karnouskos, and Xin Li. A survey on edge and edge-cloud computing assisted cyber-physical systems. *IEEE Transactions on Industrial Informatics*, 17(11):7806–7819, 2021.

[7] Antonio Francesco Gentile, Davide Macrì, Domenico Luca Carnì, Emilio Greco, and Francesco Lamonaca. A performance analysis of security protocols for distributed measurement systems based on internet of things with constrained hardware and open source infrastructures. *Sensors*, 24(9), 2024.

[8] Peter G Harrison and Naresh M Patel. *Performance modelling of communication networks and computer architectures (International Computer S*. Addison-Wesley Longman Publishing Co., Inc., 1992.

[9] Fabrizio Marozzo and Vinci Andrea. Design of platform-independent iot applications in the edge-cloud continuum. In *20th International Conference on Distributed Computing in Smart Systems and the Internet of Things, DCOSS-IoT 2024, Abu Dhabi, United Arab Emirates, April 29 – May 1, 2024*. IEEE, 2024.

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

[11] Fabrizio Marozzo, Domenico Talia, and Paolo Trunfio. A workflow management system for scalable data mining on clouds. *IEEE Transactions On Services Computing*, 11(3):480–492, 2018. ISSN: 1939-1374.

[12] Carlo Mastroianni, Francesco Plastina, Jacopo Settino, and Andrea Vinci. Variational quantum algorithms for the allocation of resources in a cloud/edge architecture. *IEEE Transactions on Quantum Engineering*, 5:1–18, 2024.

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

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

[15] Fei Tao, Dongming Zhao, Yefa Hu, and Zude Zhou. Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system. *IEEE Transactions on industrial informatics*, 4(4):315–327, 2008.

[16] Blesson Varghese, Nan Wang, David Bermbach, Cheol-Ho Hong, Eyal De Lara, Weisong Shi, and Christopher Stewart. A survey on edge performance benchmarking. *ACM Computing Surveys (CSUR)*, 54(3):1–33, 2021.

[17] Zhe Yang, Klara Nahrstedt, Hongpeng Guo, and Qian Zhou. Deeprt: A soft real time scheduler for computer vision applications on the edge. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 271–284. IEEE, 2021.