Latency-Aware Placement of Microservices in the Cloud-to-Edge Continuum via Resource Scaling

Alberto Bertoncini, Alberto Ceselli, Christian Quadri
Computer Science Department, *Università degli Studi di Milano*, Milan, Italy

***Abstract*—Latency-sensitive applications, such as autonomous driving in smart cities and industries, require robust networking and computing support. The cloud-to-edge continuum offers a promising solution by bringing computation closer to edge devices. However, heterogeneity and geographic distribution of nodes make deployment challenging. We address this in a tele-operated autonomous driving scenario by modeling the orchestration as a Virtual Network Function Placement Problem (VNFPP) with multi-tier performance levels, enabling vertical scaling per microservice. Our solution, MORAL, minimizes deployment costs based on node centrality while meeting resource and latency constraints. Simulations on realistic network topologies and synthetic applications show that our approach improves deployment feasibility and latency compliance over single-tier and baseline methods.**

***Index Terms*—Service Orchestration, Cloud-to-Edge Continuum, Mathematical Optimization**

I. Introduction

Autonomous vehicle technology is rapidly advancing, with applications spanning production, logistics, smart cities, and smart factories. By reducing human involvement in repetitive tasks, it enables safer, cleaner, and more efficient last-mile logistics. These vehicles rely on numerous onboard sensors that generate large volumes of data requiring low-latency processing. Additionally, complex operating environments necessitate the integration of external data sources.

However, limited processing power on IoT devices and autonomous vehicles hampers full on-device computation, especially under strict latency constraints. To address this, a robust networking and computing infrastructure is essential [1]. The emergence of 5G and cloud-to-edge continuum computing supports such needs, but the distributed and heterogeneous nature of this infrastructure poses new challenges. These include balancing resource utilization, meeting application requirements, and enforcing security and isolation to preserve data privacy across microservices.

This paper addresses the deployment of latency-sensitive microservice applications over a cloud-edge continuum by optimizing resource usage. We frame the problem as a Virtual Network Function Placement Problem (VNFPP), extended with vertical scaling to dynamically adjust resource tiers for each microservice to meet latency constraints. Variants of the problem reflect differing application and network requirements, with objectives typically focused on latency, delay, and resource efficiency [2]–[6]. However, no formulation for the VNFPP in edge environments explicitly integrates multi-level performance scaling under end-to-end constraints.

This work has been funded by the Italian MUR in the framework of the PRIN 2022 project *CAVIA* (Grant 2022JAFATE, CUP: G53D23002850006) under MUR (Call PRIN 2022) under the MUR NRRP, funded by the European Union NextGenerationEU.

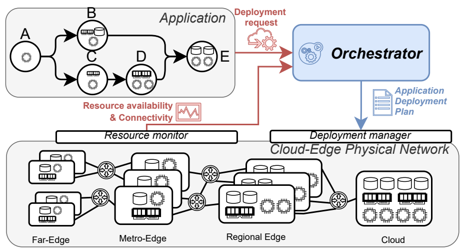

Fig. 1: System model

We introduce *Multi-level Optimization for Resource-Aware Latency constraints* (*MORAL*), a novel mathematical programming formulation that models resource requirements and service dependencies. Our contributions are: (i) native support for vertical scaling, enabling flexible resource allocation per microservice, and (ii) an objective function that minimizes reliance on topologically strategic nodes using a centrality-based deployment cost. We validate our approach through experiments on realistic urban scenarios, demonstrating improvements over models lacking vertical scaling.

II. System Model

The overall architecture is illustrated in Fig. 1. At the top left, a microservices-based application is shown. The orchestrator receives deployment requests and generates an *Application Deployment Plan*, factoring in network connectivity and resource availability.

A. Application

We model applications as directed acyclic graphs (DAGs), $M = (T, D)$, where nodes $u \in T$ represent microservices, and edges $(u, v) \in D$ denote interdependencies and data flows. Each node includes minimum hardware requirements and a processing time, defined as the maximum execution time when running with its minimum resource allocation. Processing time decreases with increased resources due to vertical scaling. Edges are annotated with the required datarate between microservices. Applications may have multiple computational flows, each with specific end-to-end latency requirements.

B. Cloud-Edge Physical Network

The physical infrastructure is modeled as a directed graph $G = (I, A)$, where I is the set of computing nodes and A the set of physical links. Nodes are annotated with available

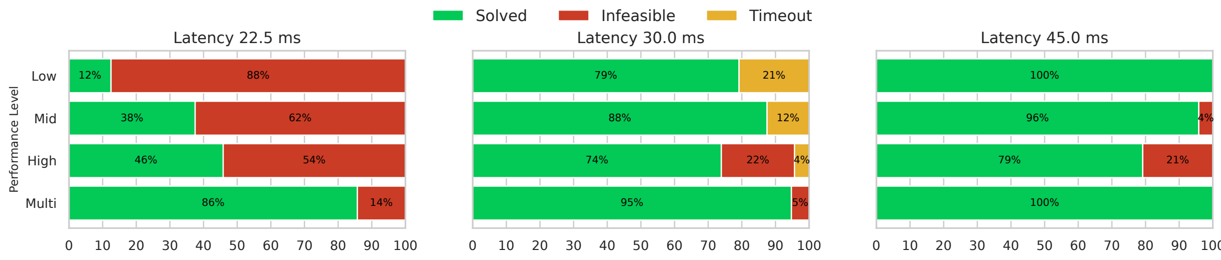

Fig. 2: Percentage of MORAL outcomes over instances, grouped by latency limit and performance level

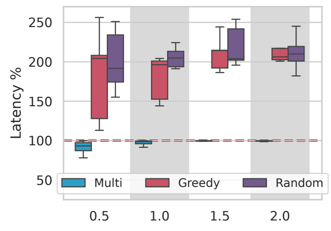

Fig. 3: Algorithm latency comparison

resources, and links with maximum datarate and minimum delay. As illustrated in Fig. 1, the infrastructure adopts a multi-tier topology: edge nodes have limited resources, while cloud nodes offer virtually unlimited capacity.

Node centrality plays a key role in meeting latency constraints. However, relying solely on central nodes can lead to resource contention. To balance deployment decisions, we assign a cost to each node proportional to its centrality. We adopt the *betweenness centrality* metric, which captures both local and global communication roles.

C. Orchestration Decision Process

The orchestrator, named *MORAL* (Multi-level Optimization for Resource-Aware Latency Constraints), processes: *(i)* the application DAG and requirements, and *(ii)* the current state of the physical infrastructure. It outputs an *assignment plan* for the deployment manager. The orchestrator maps microservices to physical nodes, ensuring compliance with resource and dependency constraints, and selects resource levels for each microservice to satisfy latency requirements across all computational flows.

We formalize the orchestration task as a VNFPP problem using Mixed Integer Linear Programming (MILP) formulation, and we solve it using a general-purpose solver to get preliminary results about the feasibility and effectiveness of the approach.

III. Results

A. Comparative Algorithms

We evaluate our model's effectiveness by comparing it with two baseline strategies that consider only node resource constraints, ignoring end-to-end latency requirements. The first baseline, *Greedy*, prioritizes vertical stacking by assigning microservices to the lowest-cost nodes at the highest feasible performance level. The second, *Random*, iteratively assigns microservices to nodes and performance levels randomly, ensuring only that resource constraints are satisfied.

B. Feasibility

The feasibility analysis, reported in Fig. 2, shows higher success rates for the multi-level performance configuration and the results are consistent across all the latency classes. On the contrary, fixed-level performance configurations struggle to find feasible solutions. Focusing on the lowest latency class (22.5 ms), we observe that the low-level configuration does not meet the latency requirements due to a long processing time. In contrast, the high-level configuration quickly saturates strategic node capacity, forcing the deployment of microservices on other nodes, and failing to meet latency requirements. By increasing the end-to-end latency threshold, the solver can find more feasible solutions. However, we can note that high-level performance configuration cannot provide feasible solutions for a non-negligible number of instances.

The results shown in Fig. 2 highlight the benefits of flexible performance level instantiation, allowing the solver to deploy the most suitable instance for each application microservice.

C. End-to-end Constraint respect

We analyze the maximum latency observed across microservice chains in each deployment. Since latency is not directly minimized – only constrained – solutions aim to respect the bound without optimizing its value. Our multi-performance level formulation not only balances resource allocation to ensure feasible deployments under strict end-to-end constraints, but also maintains latency within acceptable bounds, up to three times lower compared to baseline methods that consider only node-level constraints, as shown in Fig. 3.

References

[1] N. Aljeri and A. Boukerche, "Mobility management in 5g-enabled vehicular networks: Models, protocols, and classification," *ACM Comput. Surv.*, vol. 53, Sept. 2020.

[2] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *2015 11th International Conference on Network and Service Management (CNSM)*, pp. 50–56, 2015.

[3] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," in *IEEE INFOCOM 2018*, pp. 693–701, 2018.

[4] J. Elias, F. Martignon, S. Paris, and J. Wang, "Efficient orchestration mechanisms for congestion mitigation in nfv: Models and algorithms," *IEEE Trans. on Services Computing*, vol. 10, no. 4, pp. 534–546, 2017.

[5] J. Gil Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Trans. on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[6] J. Yusupov, A. Ksentini, G. Marchetto, and R. Sisto, "Multi-objective function splitting and placement of network slices in 5g mobile networks," in *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, pp. 1–6, 2018.