Offloading Resources to the Cloud or to the Edge? A Liqo-Powered Testbed

Offloading Resources to the Cloud or to the Edge? A Liqo-Powered Testbed

Davide Miola*, Stefano Galantino*, Ivano Cerrato†, Francesco Lucrezia†, Fulvio Risso* and Giacomo Verticale‡
*Department of Computer and Control Engineering, Politecnico di Torino, Turin, Italy
Emails: {davide.miola, stefano.galantino, fulvio.risso}@polito.it
†Tiesse, Ivrea, Italy
Emails: {i.cerrato, f.lucrezia}@tiesse.com
‡Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milan, Italy
Email: giacomo.verticale@polimi.it

***Abstract*—As cloud-native development continues to rise and the industry embraces a Kubernetes-first paradigm, interest in cloud federation and hybrid cloud architectures is rapidly growing. These models offer compelling benefits such as seamless multi-region scalability, enhanced resilience, and flexible resource management. In this paper, we introduce a state-of-the-art, Kubernetes-native multi-cluster research testbed that leverages Liqo to enable seamless offloading of workloads and services across a distributed edge-cloud continuum. This approach supports dynamic, location-transparent deployment strategies that enhance performance and availability. Our solution paves the way for future-proof infrastructure designs that meet the evolving demands of decentralized computing environments.**

***Index Terms*—Cloud Computing, Kubernetes, Cloud federation, Hybrid cloud, Liqo**

I. Introduction

The rollout of 5G and the development of 6G networks are accelerating the shift toward distributed computing paradigms, where a growing number of edge sites are deployed to satisfy stringent latency, bandwidth, and reliability requirements. This is coupled with the growing computing capacity available at the customers premises and small offices, where the number of always-on devices such intelligent edge routers is growing day after day. However, this evolution presents several challenges. On the edge side, resource scarcity – in terms of compute, memory, storage, and power – makes it difficult to support dynamic workloads or absorb traffic surges [1]. On the cloud side, the challenge lies in managing hundreds or thousands of edge sites in a consistent, reliable, and policy-driven manner [2]. Traditional centralized orchestration models struggle to cope with the geographical dispersion, connectivity variability, and contextual diversity of edge environments.

The heterogeneity of edge hardware introduces another challenge: nodes may run different CPU architectures (e.g., x86 or ARM), integrate accelerators (such as GPUs, TPUs, or NPUs), and exhibit a wide range of configurations. These variations must be abstracted and exploited efficiently by the orchestration layer to ensure optimal application placement and execution. In this context, solutions like Liqo [3], [4] emerge as enablers of a more flexible and elastic edge-cloud continuum. Liqo, through Kubernetes-native primitives, allows clusters to dynamically extend their resource pool by offloading workloads to other clusters – whether in the cloud or at the edge – treating remote resources as local. This abstraction supports cross-site elasticity, federated workload management, and fault-tolerant execution, all while accommodating hardware heterogeneity. By bridging resource gaps and enabling seamless collaboration between clusters, such approaches lay the foundation for next-generation distributed systems, capable of meeting the operational and performance demands of 5G, 6G, and beyond.

This paper presents a research testbed that leverages Liqo to establish a computing continuum spanning from centralized cloud infrastructures to a distributed layer of edge devices. This heterogeneous federation testbed enables to deploy and orchestrate workloads and services across diverse environments, thereby facilitating the evaluation of platform stability and performance under varying conditions.

The remainder of the paper is structured as follows. Section II reviews the current landscape of existing and emerging solutions for research-oriented computing federations. Section III explores the specific use cases made possible by our testbed, with a focus on its bi-directional peering architecture. Section IV details the software stack that powers both the centralized cluster and the distributed edge nodes. In Section V we demonstrate the effectiveness of the system, which was validated on two distinct edge device types: a fleet of general-purpose *small form factor* computers, and a collection of ARM-based routers supplied by Tiesse, demonstrating the platform's scalability and cross-architecture compatibility. Finally, Section VI summarizes our findings and outlines directions for future work.

II. Background

In the following, we outline the most promising cloud and edge research-oriented testbed, with a deep focus on the enabling technologies.

A. Distributed testbeds for experimentation

Distributed experimentation platforms such as Shary [5], CloudLab [6], and FABRIC [7] share a common emphasis on deep programmability, resource transparency, and reproducibility, yet each targets a different layer of the infrastruc-

ture stack. Shary [5] focuses on in-network computing by exposing programmable switches (e.g., P4-capable devices) as first-class execution platforms, enabling researchers to push data-flow processing and lightweight computation directly into the network fabric, while preserving usability and performance visibility for rapid prototyping of novel networked systems. CloudLab [6], by contrast, offers raw, low-level access to heterogeneous servers, storage, and networking gear across multiple U.S. campuses, prioritizing strict performance isolation and full-stack repeatability so that new IaaS designs can be built, instrumented, and benchmarked end-to-end under realistic conditions without the opacity of commercial clouds. FABRIC [7] bridges these approaches at the national scale by integrating compute, storage, and a fully controllable data-and-control plane into a single programmable substrate: experimenters can define arbitrary topologies, inject custom routing or protocol logic, and orchestrate resources across geographically dispersed sites to explore next-generation Internet architectures and services at scale. Together, these testbeds illustrate a unified trajectory toward converging computation and communication, providing researchers with layered infrastructure abstractions that are both powerful and transparent. However, none of the proposed testbeds extends beyond the data center premise, limiting interoperability with resource-constrained devices available at the edge.

B. Cloud federation initiatives

Several initiatives across Europe are converging toward the creation of federated, interoperable, and distributed cloud-edge infrastructures to support research, innovation, and industrial applications. *Dynamo* [1] offers an open marketplace targeting federated and multi-cloud environments, potentially enabling real deployments and performance evaluations of cloud-native applications across heterogeneous providers. Complementing this, the *Fulcrum Project* [2] introduces a marketplace-based federation model through two core components: the Inter-Cloud Exchange Foundation (ICX), which defines open-source consensus frameworks, and the Computing Exchange Market (CEM), which implements those frameworks to enable onboarding, interoperability, observability, and billing across multiple providers. Although promising, their marketplace-based approach limits the research space for practitioners, who can experiment only on the application level, but not on the infrastructure level (e.g., orchestration and resource management). Finally, *Lab8ra* represents a cutting-edge, cross-organization testing environment established within the IPCEI-CIS initiative, interconnecting cloud and edge infrastructures into a federated system across multiple European countries. It brings together industrial and research actors to pilot real-world distributed scenarios—such as smart mobility—while fostering workload mobility, interoperability, and the industrial adoption of next-generation federated systems. However, the testbed is currently in its early stages, and it does not include edge resources.

[1]https://www.dynamo.cloud/

[2]https://www.fulcrumproject.org/

C. Solutions to control a distributed cloud infrastructure

Rancher Fleet [8] leverages a GitOps model to reconcile Kubernetes manifests at scale: it ingests Git repositories, renders resources as Helm charts, and uses custom resource definitions (CRDs) and controllers to roll out and monitor applications across many clusters—yet leaves each cluster as an isolated logical entity with no shared network or storage fabric. Karmada [9] builds on a master-worker federation approach to offer centralized multi-cluster scheduling, automated failover, and traffic routing without altering application code, but it too treats clusters as peers under a single control plane and depends on external solutions (e.g., Submariner[3]) for inter-cluster connectivity. Liqo [4] pushes further toward mesh-style federation by introducing dynamic peering, workload offloading via virtual nodes, and integrated network and storage fabrics that allow pods and persistent volumes to move seamlessly across on-prem, cloud, and edge clusters—all while preserving native Kubernetes API semantics. Finally, Cilium Cluster Mesh[4] focuses narrowly on the data plane: it weaves a L3/L4 network overlay among clusters using the Cilium CNI, enabling pod-to-pod communication across environments, though it requires IP coordination or NAT workarounds and does not propagate Kubernetes Service objects natively. Together, these solutions chart a path from declarative, GitOps-based rollout through control-plane federation to fully meshed multi-cluster topologies, each trading off ease of integration, network fabric support, and logical cluster unification.

III. Use-cases

Our computing continuum testbed supports a range of distinctive use cases, enabled by a Liqo-based peering mesh that spans across an heterogeneous fleet of edge devices. Liqo was selected for its transparency (it operates on standard Kubernetes objects) and its capability to create a service, network and storage fabric across the continuum, hence providing a better matching for our objectives compared to the other solutions listed in Section II. Each peering relationship is bidirectional, allowing for flexible and generic offloading scenarios. The design thus natively accommodates two principal modes of operation: *Cloud-to-Edge* and *Edge-to-Cloud*, as shown in Figure 1 and detailed in the following.

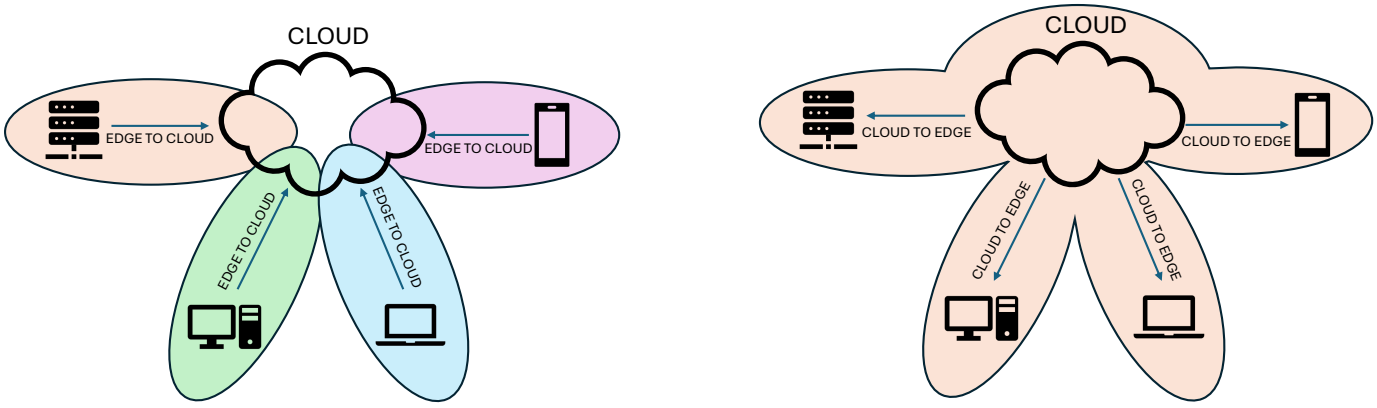
A. Edge to Cloud

In this class of use-cases, edge devices leverage cloud capabilities to compensate for local limitations, be it compute, storage, or durability. The directionality here emphasizes the edge as the initiator of interactions with centralized resources. A typical use case encompasses a SOHO user leveraging both its local resources, and cloud offloading.

1) Resource extension/compute burst management: Edge nodes typically operate with constrained resources, especially in terms of compute and memory. To accommodate peak loads or computationally intensive tasks, they can offload processing

[3]https://submariner.io.

[4]https://docs.cilium.io/en/stable/network/clustermesh/intro.



(a) Edge-to-Cloud.

(b) Cloud-to-Edge.

Fig. 1: Schematic overview of the Liqo cluster mesh extending between Cloud and Edge clusters.

to the cloud. This model, often referred to as "cloud bursting", enables dynamic scaling beyond the physical capacity of edge devices. Key challenges include minimizing offloading latency, maintaining data locality, and synchronizing intermediate results between edge and cloud.

2) Hardware constraints: Constrained hardware environments at the edge – such as limited thermal budgets, minimal memory, or lack of specialized computing accelerators – necessitate the selective execution of services locally. The cloud serves as an overflow buffer where heavier tasks can be redirected. Inference workloads, large-scale simulations, and training pipelines are prime candidates for such delegation. Hybrid execution models are thus required to partition logic across nodes depending on current resource states.

3) Cloud for reliability/persistency: Persistency guarantees are harder to meet at the edge, where data loss risks are elevated due to storage wear-out, power instability, or harsh environments. To preserve critical data and maintain long-term records, the cloud acts as a durable backend. Applications can replicate data asynchronously to cloud storage, periodically checkpointing computing states, and rely on cloud-based coordination services to maintain global consistency in distributed deployments.

B. Cloud to Edge

This category encompasses scenarios where centralized cloud systems are extended toward the edge, with the objective of deploying workloads or services closer to the data source or users. Such use-cases typically arise in industrial IoT, smart cities, and remote infrastructure management, such as a company with multiple branch offices (e.g., bank) over a geographical infrastructure.

1) Orchestration: In distributed environments stretching from cloud to edge, orchestration plays a crucial role in managing the lifecycle of workloads across heterogeneous resources. It includes deploying applications to edge nodes, ensuring configuration consistency, and dynamically adapting to changes in network topology or resource availability. Emerging orchestration platforms support declarative specifications, policy-based deployment, and intent-driven placement to meet latency, bandwidth, or locality requirements [10].

2) Scalability: Scaling across thousands of edge locations – often with similar configurations but independent operations – requires a management plane capable of templating, replication, and low-touch operations. Federated orchestration models and GitOps practices are increasingly adopted to enable consistent deployments while retaining the autonomy of individual sites, but they alone cannot provide a transparent computing continuum spanning across edge and cloud. Furthermore, efficient propagation of updates and context-aware scaling policies are essential to preserve service continuity across the fleet.

3) Disaster recovery: Edge nodes are often deployed in environments where physical access is limited and network connectivity is intermittent. Disaster recovery mechanisms must ensure that edge services can recover from power failures, software faults, or hardware degradation with minimal cloud dependency. This may involve local snapshots, transactional state replication, and periodic synchronization with the cloud to rehydrate state or re-provision services upon failure.

4) Observability: Maintaining observability across distributed sites is essential for debugging, auditing, and optimizing operations. Edge observability must function under constrained uplink conditions and often relies on local data pre-aggregation, selective telemetry, and asynchronous log forwarding. Integration with centralized systems must allow for time-correlated analysis of metrics, traces, and logs to detect anomalies and enforce compliance.

IV. Testbed Design

The system has been designed with an emphasis on portability and extensibility, and released as open-source[5]. To this end, all architectural components are compatible with multiple CPU architectures, ensuring broad hardware support. Additionally, the deployment process is streamlined through a set of Ansible

[5]https://github.com/netgroup-polito/edge-infrastructure-ansible.

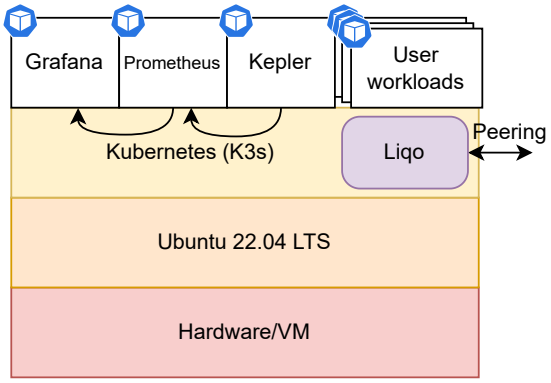


Fig. 2: Software stack powering all components of the distributed testbed.

[11] playbooks, which automate the provisioning and configuration of each testbed component. This approach significantly reduces the need for manual intervention, facilitating the seamless integration of new nodes into the infrastructure.

The open-source software stack deployed on each node within the federation is shown in Figure 2. The testbed architecture supports both bare-metal deployments and virtualized environments: the former is preferable for resource-constrained edge devices, while the latter enables integration with managed cloud platforms such as OpenStack.

The Kubernetes container orchestrator serves and the primary user interface for workload scheduling and service exposure within our testbed. We specifically utilize the lightweight **K3s** [12] distribution – an official Cloud Native Computing Foundation (CNCF) Sandbox project – optimized for resource-constrained environments, making it especially suitable for edge-layer deployment. Designed as a single binary, K3s consolidates multiple Kubernetes control-plane components – such as the API server, scheduler, and controller manager – into one process. It also uses a lightweight SQLite-backed datastore by default, hence reducing memory and inter-process overhead while maintaining full Kubernetes API compatibility.

Kubernetes installations distributed across the heterogeneous set of cloud and edge devices are federated with **Liqo**, which enables the dynamic creation of *extended clusters* spanning across multiple participating sites. These federations allow for seamless offloading of Pods and Services across the mesh. By default, Liqo establishes unidirectional peering relationships, assigning each cluster the role of either *producer* or *consumer*, and permitting offloading only from producer to consumer. To accommodate all the use-cases discussed earlier, our deployment configures Liqo to establish bidirectional peering relationships instead. This configuration ensures that each cluster can both consume and offer resources within predefined quota limits, hence enabling both *edge-to-cloud* and *cloud-to-edge* scenarios. Within the computing continuum enabled by Liqo, Pods and Services retain cross-cluster reachability via a shared network fabric. Key features include *Resource Reflection* for automatic replication of offloaded Kubernetes resources, *CIDR remapping* to properly support conflicting IP ranges, and encrypted inter-cluster communication. Once the Liqo mesh is established, Pods can be offloaded to remote clusters by simply scheduling them onto the cluster's associated virtual node, which Liqo provisions on a one-per-remote-cluster basis as a representor of the entirety of the provider's exposed resources through the *Virtual Kubelet* [13] paradigm.

The Ansible playbooks further configure the mesh with additional services to enhance monitoring of the testbed. **Prometheus** [14] is used as the aggregator of metrics produced by both Kubernetes and Liqo, while **Grafana** [15] provides the dashboards to consume such data. Power consumption tracking and monitoring of each site is handled by **Kepler** [16], which uses eBPF and hardware counters to collect energy usage statistics at the container, Pod, and node levels while also integrating into our Prometheus/Grafana setup. For example, this enables smart energy-based orchestration policies, e.g., to privilege sites that are energy-friendly, as proposed in the EU FLUIDOS project [17].

V. Validation

The system was validated to ensure correct behavior and expected functionality. To this end, we deployed our fluid computing platform on two distinct edge infrastructures to assess compatibility and scalability across diverse device types: a fleet of Intel-based small form factor computers and a set of ARM-based routers provided by Tiesse[6]. The following subsections go into additional details regarding both setups. In each configuration, we provisioned a central Kubernetes cluster to serve as the shared cloud tier of the testbed. This cluster runs on a single server hosted in Politecnico di Torino's datacenter, featuring dual Intel Xeon Gold 6442Y CPUs and 512 GiB of RAM, configured with the same software stack used at the edge. Additionally, the central server includes an NVIDIA A30 GPU, enabling advanced scenarios involving remote accelerator sharing across edge nodes.

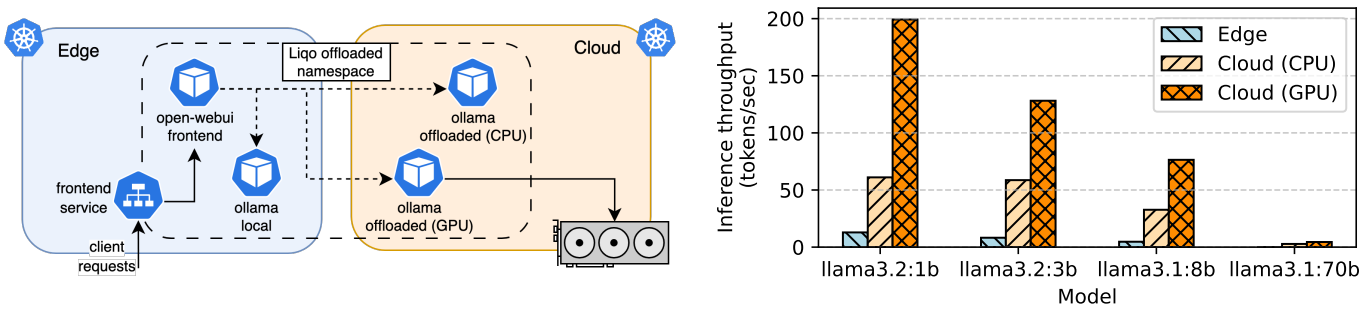
A. MiniPC edge hardware

Our primary target edge device is the *Trigkey G5* MiniPC, a small-form-factor x86-based all-in-one computer equipped with a four-core 12th generation Intel N100 processor running at 3.40 GHz, 16 GiB of DDR5 RAM, and a pair of 2.5 Gbps network interfaces based on the Intel I225 chipset. We designed the testbed around this category of edge devices to mimic the typical home gateway appliance, thus fully supporting edge deployments at home or office locations.

B. Tiesse's ARM-based router edge hardware

We also evaluated the compatibility of our software stack on Tiesse's **Imola 7** network appliance, based on an NXP Layerscape LS1046A system-on-chip, which includes an ARM A72 4-cores CPU, 4GB DDR4 RAM, and 8GB eMMC flash. The appliance is equipped with 2x10 Gbps SFP+ uplinks, 4x1 Gbps Base-T/SFP ports, WiFi, and support for 4G/5G

[6]https://www.tiesse.com/en



(a) (b)

Fig. 3: Setup (Figure 3a) and measured performance (Figure 3b) for the experimental validation of our testbed's Edge-to-Cloud use-case.

WAN interfaces, enabling dual-WAN operations and cellular fallback functionality.

C. Edge-to-Cloud

To assess the offloading capabilities of the testbed in the Edge-to-Cloud scenario, we devised a test that would validate the effectiveness of the system in handling and scheduling resource-intensive tasks throughout the computing continuum. To this end, we arranged an LLM inferencing workload as the primary benchmark, citing its high requirements in terms of computing resources and memory usage as the main motivators for justifying the offloading to a cloud environment. Moreover, this workload can easily take advantage of the GPU accelerator that our central cloud server can provide, thus demonstrating the *resource sharing* features of the platform.

As shown in Figure 3, we used a combination of the *Ollama* [18] inference engine backend with the *Open-webui* [19] frontend to deploy local LLM functionality throughout the testbed's computing mesh. All Kubernetes Pods and Services were deployed on each edge node command line, although two replicas of the Ollama runners were explicitly scheduled to be offloaded to the cloud server; one of them was additionally setup to run under the `nvidia` container runtime – which had previously been installed on the central cluster – to make use of GPU acceleration.

We tested the tokens/s inference throughput of the system under multiple Llama [20] model sizes and backend host configurations, and confirmed that users of edge nodes can effectively exploit the increased CPU and GPU performance when offloading workloads to the central cloud cluster.

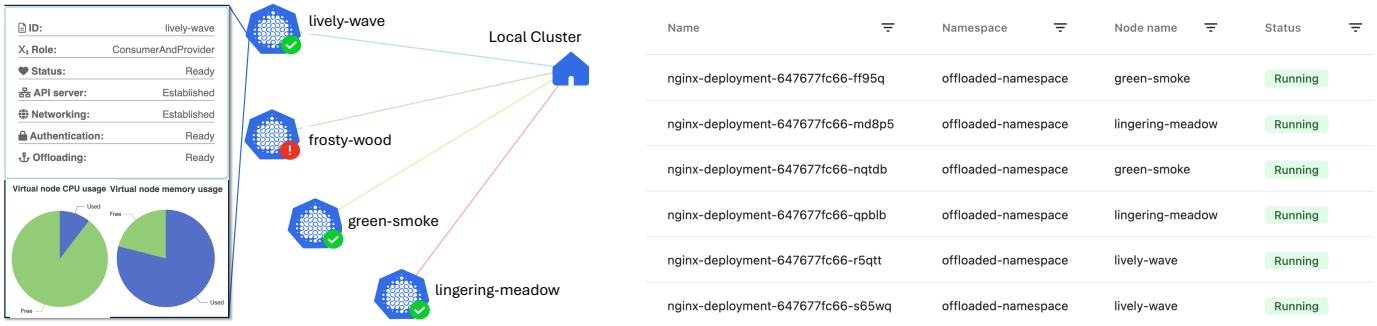
D. Cloud-to-Edge

The integration of Cloud and Edge infrastructures enables centralized observability, allowing operators to maintain a comprehensive view of the status and availability of geographically distributed edge sites. Leveraging Liqo, edge clusters are seamlessly integrated into the central control plane and are exposed as virtualized compute nodes within the central cluster. This abstraction, illustrated in Figure 4a, allows edge resources to be managed as if they were part of the cloud infrastructure, despite their physical separation and potentially limited connectivity. As a result, the central cluster gains the capability to inspect both the availability and reachability of the federated edge clusters. In the example shown in the figure, three clusters are visible and operational, accepting workload deployments from the central orchestrator. Conversely, one cluster—anonymized as `frosty-wood`—is flagged as unreachable, indicating a temporary disruption in the communication channel. This status is automatically detected and reflected in the global dashboard, enabling prompt detection and response to failures or disconnections at the edge.

Beyond binary availability indicators, the monitoring framework is enriched with edge-specific metrics. Each edge cluster continuously shares telemetry data with the central cluster, exposing internal resource metrics such as CPU, memory, and GPU utilization (when available), as well as the state of scheduled workloads (e.g., running, pending, failed). These metrics are collected using a unified observability stack based both on Kubernetes reporting information and on Prometheus, which ensures compatibility with Prometheus-based querying and alerting mechanisms. Figure 4b reports the pod distribution across the federated edge sites as a consequence of a possible orchestration policy, providing deeper insight into the functionality of the application. Additionally, some of the edge sites currently share power consumption metrics through Prometheus, including also the energy mix (e.g., solar vs carbon) at each each edge site, enabling enhanced energy-aware allocation policies from the centralized cluster.

VI. Conclusions

In this paper, we presented a distributed research testbed implementing a hybrid cloud architecture through a federation of disaggregated Kubernetes clusters interconnected via a Liqo mesh. Through the bidirectional workload offloading capabilities offered by Liqo, our testbed enables diverse Cloud-to-Edge and Edge-to-Cloud use-cases, such as seamless multi-cluster scalability across cloud and edge deployments, resource sharing, improved reliability, and more. We validated the system using a centralized server node – capable of offering large computing resources and equipped with GPU accelerators –,



(a) (b)

Fig. 4: Testbed visualization as seen by the local cluster (Figure 4a), and workload distribution across the federation (Figure 4b).

and a fleet of distributed edge devices – unlocking low-latency, low-power computing –, and demonstrated its effectiveness by deploying a sample LLM inferencing workload on the computing mesh.

Acknowledgements

This work was partially supported by the European Union - Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3, CUP E83C22004640001, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

It was also supported by European Union's Horizon Europe research and innovation programme under grant agreement No 101070473, project FLUIDOS (Flexible, scaLable, secUre, and decentralIseD Operating System).

Finally, this publication is part of the project PNRR-NGEU, which has received funding from the MUR - DM 117/2023.

References

[1] L. F. Bittencourt, R. Rodrigues-Filho, J. Spillner, F. De Turck, J. Santos, N. L. da Fonseca, O. Rana, M. Parashar, and I. Foster, "The computing continuum: Past, present, and future," *Computer Science Review*, vol. 58, p. 100782, 2025.

[2] S. Galantino, F. Risso, A. Cazzaniga, F. Garrone, R. Terruggia, and R. Lazzari, "An edge-based architecture for phasor measurements in smart grids," in *2022 AEIT International Annual Conference (AEIT)*. IEEE, 2022, pp. 1–6.

[3] The Liqo Contributors. (2025) Liqo: Enable dynamic and seamless kubernetes multi-cluster topologies. An open-source solution for Kubernetes cluster peering, workload offloading, and multi-cluster networking and storage fabric. [Online]. Available: https://liqo.io/

[4] M. Iorio, F. Risso, A. Palesandro, L. Camiciotti, and A. Manzalini, "Computing without borders: The way towards liquid computing," *IEEE Transactions on Cloud Computing*, vol. 11, no. 3, pp. 2820–2838, 2023.

[5] S. Salsano, A. Mayer, P. Lungaroni, P. Loreti, L. Bracciale, A. Detti, M. Orazi, P. Giaccone, F. Risso, A. Cornacchia *et al.*, "Sharing gpus and programmable switches in a federated testbed with shary," *arXiv preprint arXiv:2501.18840*, 2025.

[6] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, K. Webb *et al.*, "The design and operation of {CloudLab}," in *2019 USENIX annual technical conference (USENIX ATC 19)*, 2019, pp. 1–14.

[7] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "Fabric: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2020.

[8] Rancher. (2025) Rancher fleet: manage gitops for large scale deployment of kubernetes clusters. [Online]. Available: https://fleet.rancher.io/

[9] Karmada Authors. (2025) Karmada: Open, multi-cloud, multi-cluster kubernetes orchestration. [Online]. Available: https://karmada.io/

[10] S. Braghin and L. Nedoshivina, "MIMO: a framework for extensible and flexible intent-based workload meta-orchestration," in *Proceedings of the 2nd International Workshop on MetaOS for the Cloud-Edge-IoT Continuum*, 2025, pp. 1–6.

[11] The Ansible authors. (2025) ansible/ansible: Ansible — radically simple it automation. Open-source, agentless IT automation platform handling configuration, deployment, cloud provisioning, networking, and orchestration across systems and clusters. [Online]. Available: https://github.com/ansible/ansible

[12] K3s Project. (2025) K3s: Lightweight, certified kubernetes for edge & iot. A lightweight, CNCF-certified Kubernetes distribution packaged in a single ¡70MB binary; designed for resource-constrained, remote, or IoT environments. [Online]. Available: https://k3s.io/

[13] The Virtual Kubelet authors. (2025) Virtual kubelet: Kubernetes node abstraction for serverless and multi-cluster providers. [Online]. Available: https://virtual-kubelet.io/

[14] Prometheus Authors. (2025) prometheus/prometheus: The prometheus monitoring system and time series database. A graduated CNCF project; open-source monitoring toolkit with a multi-dimensional data model, PromQL, local storage, and powerful alerting. [Online]. Available: https://github.com/prometheus/prometheus

[15] Grafana Authors. (2025) grafana/grafana: The open and composable observability and data visualization platform. A CNCF-related, open source observability and data visualization tool supporting metrics, logs, and traces. [Online]. Available: https://github.com/grafana/grafana

[16] Sustainable Computing Project. (2025) sustainable-computing-io/kepler: Kepler — efficient power level exporter for kubernetes. A CNCF Sandbox project and Prometheus exporter using eBPF to expose container, pod, and node energy consumption metrics. [Online]. Available: https://github.com/sustainable-computing-io/kepler

[17] N. Asadov, V. C. Coroamă, M. Franzil, S. Galantino, and M. Finkbeiner, "Carbon-aware spatio-temporal workload shifting in edge–cloud environments: A review and novel algorithm," *Sustainability*, vol. 17, no. 14, 2025.

[18] Ollama Authors. (2025) ollama/ollama: Get up and running with large language models. Open-source framework for running and managing LLMs locally across macOS, Linux, and Windows. [Online]. Available: https://github.com/ollama/ollama

[19] Open WebUI Authors. (2025) open-webui/open-webui: A self-hosted, extensible ai interface. Feature-rich, offline-capable AI platform supporting Ollama/OpenAI-compatible runners, RAG, plugins, pipelines, RBAC, vision, and web browsing integration. [Online]. Available: https://github.com/open-webui/open-webui

[20] A. Grattafiori, et al., "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783