The Slices Cloud Continuum Blueprint: a Resilient Infrastructure to Support Large-scale Cloud Continuum Experiments

Andrea Sabbioni†, Armir Bujari*, Paolo Bellavista*
Department of Computer Science and Engineering, University of Bologna, Italy
*{name.surname}@unibo.it,
†{name.surname}5@unibo.it

***Abstract*—Cloud computing has undergone a profound transformation in recent years, evolving to a comprehensive computing platform catering to the specific needs of verticals, with an increasing number of deployment, ownership, and service models continuously emerging in the market. The simultaneous exploitation of multiple cloud and computing models, commonly referred to as the Cloud Continuum (CC), is gaining increasing interest among researchers and practitioners. In this context, the Slices Research Infrastructure (Slices-RI) aims to provide a community infrastructure that supports researchers in their experiments and the development of new technologies in the CC. In this work, we present the CC Blueprint (CCBP), a work-in-progress effort that focuses on providing the community with a replicable set of software, hardware, and methodologies to conduct experimental research in cutting-edge distributed environments. The CCBP is built with composability at its core, offering a clear abstraction over services and resources across multiple sites, while centralizing user authentication and experiment configuration. The blueprint is rigorously tested to ensure its functionalities and assess its adaptability to host and control resources and services, from large-scale deployments to constrained ones, made available by Slices-RI partners.**

***Index Terms*—Cloud Continuum, Edge-Cloud, Research Infrastructure, Platform Engineering**

I. Introduction

The Cloud Continuum (CC) refers to the seamless integration of distributed cloud technologies and environments across various deployment models, locations, and services. It embodies the evolving landscape of cloud computing, where businesses can effectively leverage various resources to address their specific needs. This concept spans public, private, multi, and hybrid clouds, edge computing, and on-premises data centers, employing virtual or physical resources [1].

This computing paradigm provides unprecedented flexibility in terms of costs, performance, and reliability, satisfying the diverse needs of verticals, such as Smart Manufacturing, Smart Agriculture, and Smart Cities. CC support frameworks should offer several critical features. They should ensure interoperability across diverse environments, enabling smooth operations between clouds, on-premises infrastructure, and edge-IoT systems. They should provide scalability, allowing resources to be dynamically allocated as needs evolve. Security should be maintained consistently across all employed environments through unified policies and controls. Automation and orchestration should simplify the management of workloads across the CC with minimal manual intervention, while integrated data flows should enable powerful AI and analytics capabilities.

In this work, we present the Slices Cloud Continuum Blueprint (CCBP), a project within the SLICES ecosystem [2], which aims to provide a flexible and modular architecture that can support the execution of experiments over CC resources. The adaptability of the CCBP lies in the capability of Slices partners to offer different types of resources and services through high-level interfaces managed by specialized controllers. The aim is to provide the research community with a clear set of abstractions, methodologies, and technological building blocks that can be used to create complex environments spanning heterogeneous resources, all managed programmatically in a uniform and seamless way. To this end, the architecture of the blueprint has been designed to minimize the control points for improved scalability, while allowing to onboard an increasing number of geographically distributed resources, ensuring overall reliability of the infrastructure, mitigating the effects of failures or network partitions.

The CC Blueprint is distributed as an open-source solution to enable scientists and researchers to experiment with it and Slices partners to contribute to its development [3].

II. Cloud Continuum Experiments

Since the early stages of cloud computing, researchers have been very interested in developing and providing tools and supporting infrastructure to experiment with different cloud computing technologies and models [4]. A notable example is CloudLab [5], a flexible scientific infrastructure designed for research on the future of cloud computing. It allows researchers to build and experiment using bare metal hardware and various software stacks. CloudLab provides control, visibility, and interoperability across multiple sites and federated facilities in the US and beyond, supporting hundreds of simultaneous experiments in isolated environments. Recent literature has explored the concept of the Cloud Continuum, highlighting various approaches to managing and deploying resources in a heterogeneous environment. One such approach exploits modern DevOps technologies to manage the distributed deployment of services and applications across multiple cloud platforms, such as AWS, Azure, and Google Cloud. For example, [6] proposes and evaluates IoTDeploy, a solution for streamlining and scaling static and dynamic IoT service deployment over

ISBN 978-3-903176-72-0 ©2025 IFIP

the continuum. IoTDeploy implements a CI/CD tool plugin for deploying applications running on the continuum and supports dynamic service migration. Although this approach provides great flexibility, potentially enabling the integration of any cloud computing model and technology, the associated overhead and boilerplate code needed to support the experiment can be overwhelming. Moreover, this approach is mainly based on static configuration, offering limited support for dynamic reconfiguration and failure tolerance, thus reducing the resiliency of the proposed infrastructure.

A second approach that can be considered is the integration of middleware to create an overlay infrastructure on top of existing cloud solutions. An example of this approach is the Service Mesh technology [7], [8], which builds an infrastructure support targeting multi-site observability, security, and network management, enabling the transparent deployment of distributed workloads. The approach offers greater dynamism and automation support for CC DevOps, but its intrinsic nature limits its support to service-oriented CC experiments and does not support other types of experiments, such as infrastructural ones.

To overcome the limitations of DevOps and Overlay approaches, both researchers and practitioners have proposed initial solutions for the creation of federated CC infrastructures. On the business side, the cloud native computing foundation and, in particular, the Special Interest Group on Multicluster, have been developing platform extensions and tools to support the management of multiple Kubernetes clusters. As a notable example on this front is Karmada, which aims to create a centralized control plane that enables the definition and distribution of workloads in geographically distributed Kubernetes instances [9], [10].

On the research front, different works have proposed the extension of existing infrastructure solutions by creating centralized controllers capable of provisioning and managing the lifecycle of resources at different sites [11]–[14]. For example, the authors in [12] propose *Liqo*, a framework that enables the provision of federated deployments and control over different Kubernetes sites, supporting them with a federated network and storage solution. All of the aforementioned proposals focus on a limited range of use cases, providing specialized infrastructure that supports only certain technologies and cloud models.

Our proposed CC architecture and accompanying technological stack address these limitations by introducing a set of pluggable decentralized controllers. These controllers enable partners within a federated cloud continuum infrastructure to offer various cloud models, technologies, and managed services. In addition, the decentralized nature of our control plane improves the reliability of the federated infrastructure, making it resilient to failures and network partitions. The overall infrastructure is then tailored to support the execution of CC experiments through the integration of automation and best practices, simplifying researchers' experiences with the platform.

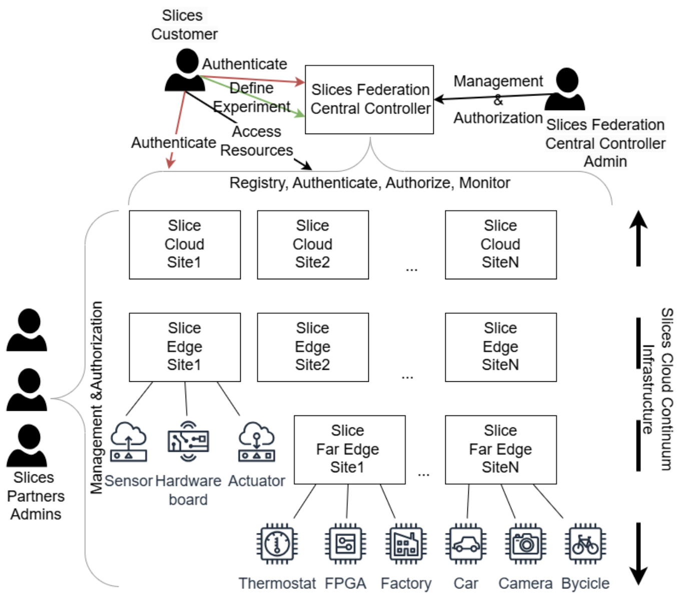

Fig. 1: High level view of the Slices Ecosystem featuring distributed and geo-delocalized sites participating in the federated infrastructure and offering Services and Resources to support Cloud Continuum experiments

III. The Cloud Continuum Blueprint Architecture

The core objective of the CC Blueprint is to provide a distributed and decentralized solution that allows researchers to experiment with new technologies, algorithmic approaches, and new resource distribution & management models. The CCBP embodies a hierarchical and programmatic control plane built atop of heterogeneous technologies. Slices Customers, after defining the necessary resources of their Experiment, can deploy it transparently over the CC resources, and this task is automatically handled via the cooperation of the logically centralized control plane and local control plane components available at each Slices site. This hierarchical design alleviates the criticalities of a single centralized control plane, providing local sites with the autonomy of tailored management & control strategies over their resources. It is worth noting that while a purely decentralized management approach may be viable in certain scenarios, it is currently not fully supported by mainstream management frameworks, which rely on strong consistency for cluster state management.

Specifically, the blueprint defines two entities: the Slices Federation Central Controller (SFCC) and the *Site*, as depicted in Fig. 1. The SFCC represents a logically centralized endpoint, maintained by the *Slices Federation Central Admin*, where *Experimenters* can authenticate and define experiments. A *Slices Site* represents a physical/virtual site, exposing its resources and services to the Slice community, maintained by Slices Partners Admins.

In modern CC deployments, the complexity of managing heterogeneous and distributed resources can result in an overhead that is not acceptable for many experimental use cases, which would greatly benefit from ready-to-use and managed services. For this reason, each Slice Site can decide to provide both basic

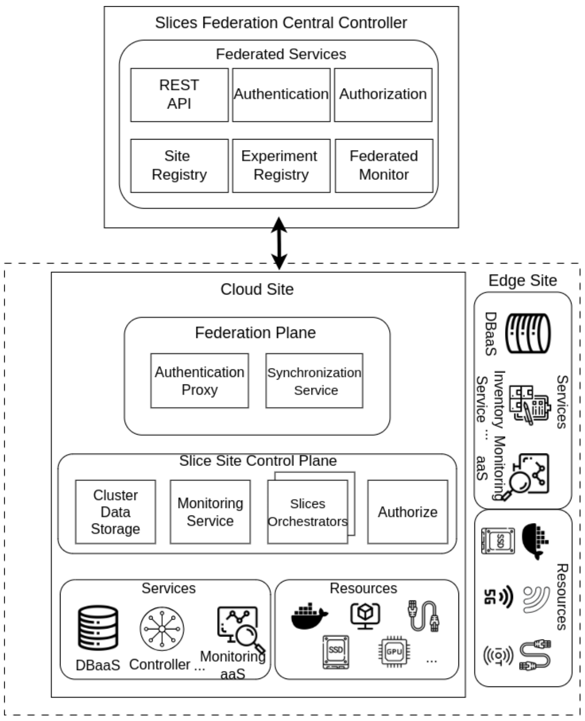

Fig. 2: High level architecture of the CC blueprint showing distribution of services in the Slices Federation Central Controller and Slices Sites

computational resources, such as containers, virtual machines, and physical networks, as well as ready-to-use services, such as databases, brokers, or application-specific controllers. Zooming in on the architecture of the SFCC (Fig. 2), the site primarily serves as the endpoint for the submission of new Experiment definitions through a REST API interface. The definitions are then permanently stored in the Experiment Registry, which can be retrieved later through the REST API by authenticated customers and the infrastructure controller.

Indeed, to submit a request to the REST API, each user and site must be authenticated against the Authentication server of the SFCC. These requests are then verified by the REST API service, checking their authorization to access SFCC service through the *Authorization Server*. To list available resources, customers submit availability queries to the *Site Registry*, which serves as an endpoint for the Cloud Site resources registration and an advanced query engine.

Services of the SFCC are executed on top of a federated infrastructure that, thanks to additional services such as Disaster Mitigation, Data Replication, and High Availability Hosting, can ensure the resiliency of the infrastructure even in the case of failures.

On the Slice Site, each Slices partner admin must deploy the Federation Plane and Slice Site Control Plane on top of their existing infrastructure. The Federation Plane implements the essential logic for reliable synchronization to achieve an eventually consistent state with the SFCC. This state includes both the definitions and configurations of resources requested for each experiment and user authentication. The Slice Site Control Plane, on the other hand, implements local orchestration logic to provision and manage the lifecycle of requested services and resources, regulating their authorized access.

Each Federation Plane installed at a site must include at least the Authentication Proxy, which enables transparent authentication of requests to site resources against the federation controller Authorization service, and the Synchronization service, which acts as a secondary copy of the SFCC Experiment Registry, creating a local copy of experiments relevant to the site where the Federation Plane is installed. This service is responsible for achieving a consistent state with the Experiment Registry despite potential failures and network partitions. Additionally, the *Synchronization Service* creates a utility copy of the experiment definitions stored in the *Cluster Data Storage* of the Slice Site Control Plane, serving as a cache to improve performance and reduce network traffic overhead. The Cluster Data Storage is the unique source of truth of the cluster, enabling the consistent storage of the desired state of resources and then of resources requested for Slices experiments.

Slices Orchestrators are resources and services tailored services that manage the provision, configuration, and lifecycle of Slices resources. These orchestrators compare the slice experiment definitions stored in the Cluster Data Storage with the actual state of the cluster obtained through the *Monitoring Service*. When the two states differ, the Orchestrators enact specific procedures to reach the desired state. Orchestrators also expose high-level APIs, which provide abstractions over resources and managed services.

IV. Slice Cloud Continuum Blueprint First Release Implementation

The objective of the CCBP is to provide a reference implementation that the broader community can test, install, validate, and integrate with. To this end, we developed the first implementation of the blueprint, guided by principles tailored to the project's specific needs. The main pillars of this implementation are the distributed (hierarchical) orchestration, modularity, composability, high resiliency to failure, and support for heterogeneous resources.

Figure 3 shows the architecture, along with some technological choices, of the current implementation. In the SFCC, the *REST API and Web Interface* serve as the endpoint for the Slices Experimenter. Requests to this interface are authenticated and authorized through the Keycloak SDK, which connects to the Keycloak authentication and authorization server. Keycloak is an open-source identity and access management solution designed to secure applications and services. It provides features like single sign-on (SSO), user federation, identity brokering, and social login, making it easier to manage user authentication and authorization across multiple applications. Keycloak supports various protocols such as OAuth2, OpenID Connect, and SAML, and offers a customizable user interface for authentication and managing accounts. In the blueprint, Keycloak serves as the endpoint for the authentication of both Slices experimenters and Slices site partners. In Keycloak, each user can belong to one or more groups. The ID of this group

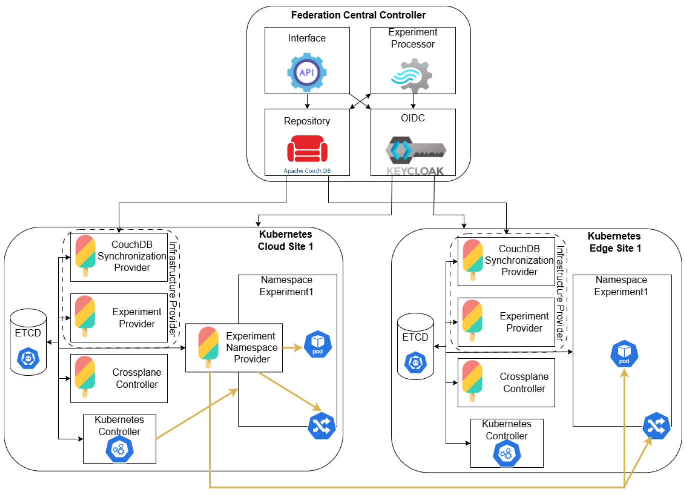

Fig. 3: Architecture of the first released implementation of the Slices Cloud Continuum Blueprint showing CouchDB and Keycloak as implementation of the Experiment Registry and Federated Authentication servers in the SFCC and the hierarchy of Crossplane controllers enabling resource management and abstractions in the Slices Sites

is then evaluated at each request by each Slices Site to check if a user has the right to access a particular resource.

In the SFCC, the Site Registry and Experiment Registry are implemented as different tables in Apache CouchDB. *Apache CouchDB* is an open-source NoSQL database that uses a schema-free, document-oriented data model. It stores data in JSON format and allows for powerful querying using JavaScript. CouchDB is designed for high availability and partition tolerance, making it ideal for distributed systems. It features a RESTful HTTP API for easy interaction with the database and supports multi-master replication, enabling seamless data synchronization across multiple nodes. CouchDB's unique append-only storage model ensures data integrity and reliability, even in the event of crashes or offload copies. To each CouchDB document are automatically associated an *ID* and a *Revision*. While the first guarantees the uniqueness of a document, enabling a key-value interaction with the database, the latter regulates the updates to the document. To modify the document, indeed, the client has to submit a document containing the modified field, but the current revision is memorized in the database. If the submitted revision and the actual revision do not match, the document is not modified. This approach enables an easy replication of passive copies of CouchDB that can be installed across the cloud continuum infrastructure to provide cache functionality and reduce stress on the SFCC. Moreover, this field is exploited by each single site that fetches only the ID and revision for each resource, and only in the case of a change in the document, pulling the entire document and minimizing the network traffic in this way.

Experiments are defined through the interface proposed by the Slices Data Management Registry and stored inside CouchDB [15] . Here, the *Experiment Processor* is responsible for converting experiment definitions in a format that is easier to manage for Slices sites. In particular, the Experiment Processor is a periodic batch process that extracts single defined resources for each experiment and stores them in a partitionable CouchDB table, as shown in Listing 1.

Listing 1: Example of a Kubernetes Namespace resource belonging to experiment1 targeting the site bologna1 and accessible by experiment1 group.

```
"ExperimentName": "exp1a",
"ExperimentGroup": "experiment1",
"EndTime": "1/1/1",
"SiteID": "bologna1",
"ResourceKind": "xslicenamespaces",
"ResourceObject": {
    "virtualMemory": "4Gb",
    "virtualCpu": "2",
    "experimentGroup": "experiment1"
    }
```

This format enables each site to retrieve assigned resources that are provisioned with a simple column filter over a partitioned and indexed table. On the Slices Site side, the site controller, tasked with fetching requested resources targeting their site and provisioning and configuring them to provide access, is implemented following the Operator Pattern.

The Operator Pattern [16] follows a declarative model in which the desired state of the cluster is memorized and controlled resources are stored in a consistent database, performed by *etcd* in Kubernetes. A stateless process, called *operator*, continuously observes the state of managed resources, and when this differs from the one stored as the desired state, it executes necessary actions to reconcile. To implement the Operator Pattern, different frameworks and ecosystems exist on the market, and we decided to adopt the *Crossplane* framework for its native support to multiple cloud providers and already developed Operator available in the store, enabling us to manage different cloud continuum resources. In particular, *Crossplane* is an open-source framework designed for creating cloud-native Control planes integrated with the Kubernetes platform. It provides tools that facilitate the definition of Custom Resources within Kubernetes and the control plane that implements the control loop for managing these resources according to Kubernetes' declarative configuration model. Moreover, Crossplane offers powerful resource organization tools, enabling the composition of interdependent resources to create higher-level logical abstractions called *composite resources*, which are accessed and managed as Kubernetes resources.

Through the Crossplane, we implemented and integrated four controllers, embracing the separation of concerns principle and implementing the synchronization and creation of Slices resources in sites. In particular, the *CouchDB Synchronization Operator* implements the process that watches CouchDB for new definitions of resources assigned to the site and replicates those definitions in a specific table "Slices" stored in the local *etcd* instance (steps 1 and 2 in Fig. 4).

The Experiment Provider transforms definitions of Slices into composite resources, augmenting them with contextual information when needed (step 3 in Fig. 4). The composite resource represents a customizable template that associates fine-grained resources and permissions. For example, the provided

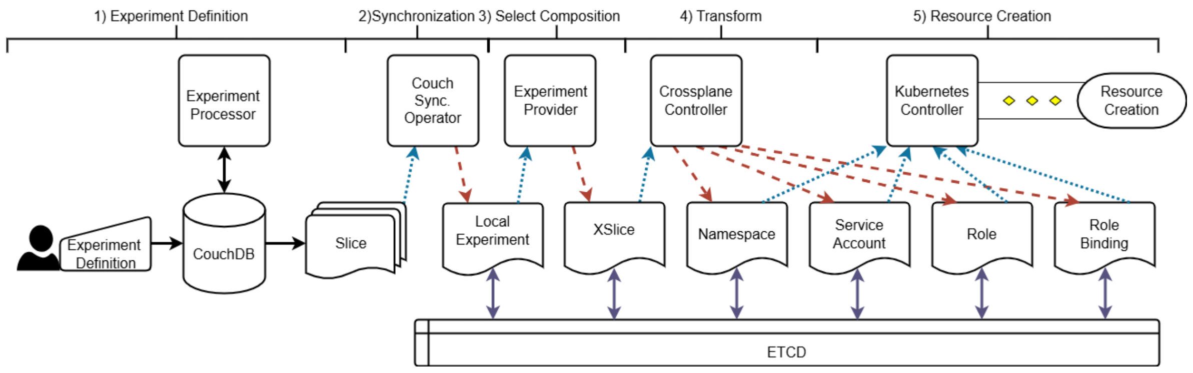

Fig. 4: The resource creation flow spans various controllers, starting with the initial definition in the Experiment Registry, followed by the creation of requested resources by the infrastructure controller. Intermediary representations of resources are saved on *etcd* to ensure the infrastructure's resiliency.

XSlice composite resource definition enables the remapping of each slice definition to a Kubernetes Namespace, a Namespaced Service account, a Kubernetes Role, and a Kubernetes Role Binding, as shown in step 4 of Fig. 4. The resources defined in this way are then stored in *etcd* to be provisioned by the Kubernetes Controller.

Optionally, sites' slices can offer the *Cross-Namespace Controller* as a service. This service simplifies the management of cloud continuum resources by providing a unique endpoint for the definition of cloud continuum resources and their lifecycle management. This controller is accessible by end-users and enables them to define multi-site deployments inside controller namespaces and a list of accessible Kubernetes clusters. The Cross-Namespace Controller watches these definitions and implements the provisioning and lifecycle management of deployments over local and remote clusters.

V. Experimental Results

Thanks to continuous automation tools, the blueprint is automatically tested at each release. These tests ensure the release's stability, highlight bottlenecks and bugs, and provide an informative overview of the computational resources needed to run its various components. We discuss a synthetic experimental analysis to provide an informative view of the effectiveness of our solution and a guide on how to tailor resources to provision the blueprint. In particular, the test reported here assesses the CCBP when deployed on a single virtual machine equipped with 8 VCPUs, 32 GB of RAM, and 80 GB of disk, and a complete Kubernetes cluster deployment characterized by 3 nodes equipped with 12 VCPUs, 24 GB of RAM, and 250 GB each.

In the first experiment, our aim is to validate the capability of the blueprint to synchronize with the SFCC and manage the creation of Namespace resource batches, with 1, 10, and 100 resources created simultaneously. This experiment shows how a node with limited capacity can handle the concurrent creation of resources belonging to different experiments, submitted simultaneously. The result depicted in Fig. 5 shows that despite the limited resources available in a single VM deployment, the

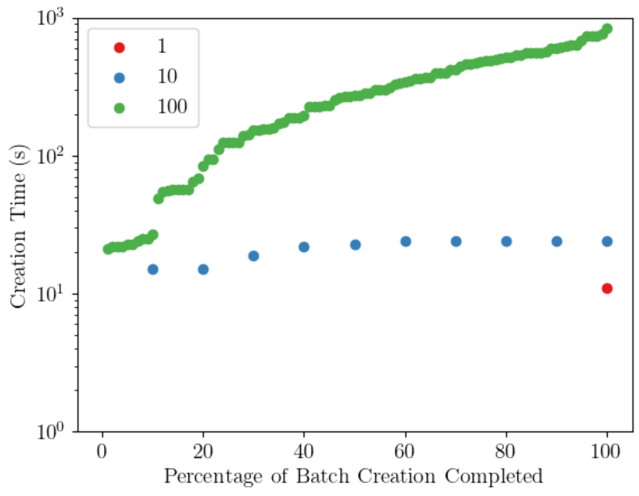

Fig. 5: Time taken by a Slice Site to create a batch of 1,10, and 100 Kubernetes Namespaces with associated User Accounts and Roles

Slice Site Control Plane reaches consistency with the desired saved state in SFCC.

In the second experiment, which focuses on deploying the blueprint in a cluster environment, we assess the resource consumption and identify the minimum requirements necessary to host the Slice Site control plane. To this end, we create an increasing number of resources ranging from 10 to 500 and collect CPU, Memory, and Network consumption of the deployed controller. It should be noted that a single experiment can request multiple resources or services from each site, and 500 resources is a reasonable number to be evaluated as a maximum for the first version of the infrastructure. As shown in Tab. I, the consumption of resource of the control plane in a scenario-like deployment is limited. In particular, CPU usage remains relatively low even as the number of resources increases, with a maximum usage of 0.183 at 500 resources. Memory usage shows a slight increase, reaching 232 MB at the highest resource count, which is still within acceptable limits for most deployment environments. Network consumption, both inbound and outbound, scales predictably with the number of resources, peaking at 266.04 kB/s inbound and 210.39 kB/s outbound at 500 resources.

N. Resources	CPU Usage	Memory Usage	Network In/Out
10	0.012	195 MB	11.7 / 6.3 kB/s
100	0.051	198 MB	50.5 / 46 kB/s
250	0.098	212 MB	134 / 104.3 kB/s
500	0.183	232 MB	266.04 / 210.39 kB/s

TABLE I: Slices Cloud Continuum Blueprint control plane CPU, memory, and network usage when 10, 100, 250, and 500 resources are created and managed.

These results demonstrate the efficiency and scalability of our solution, indicating that it can handle a significant number of resources without excessive consumption. This adaptability opens up the possibility for deployment in more constrained sites where resource availability may be limited. The ability of the control plane to gracefully scale under varying load suggests robustness and reliability, making it suitable for various operational edge/cloud environments.

VI. Conclusion

The adoption of Cloud Continuum infrastructures, which exploit heterogeneous resources from various cloud and edge services, ownership models, and distribution frameworks, is considered a hot topic within the research community. This requires a communal infrastructure that enables researchers from diverse backgrounds in ICT to experiment with CC resources. In this work, we presented the CC Blueprint, an innovative architecture that facilitates the provisioning and management of continuum resources and services for Slices experimenters. This is achieved by leveraging a geographically distributed network of partners that offer diverse resources, enabling various experiments ranging from architectural type studies to vertical specific ones. The hierarchical and compositional approach of the CCBP provides Slice sites with the autonomy to implement their access policies and provisioning strategies, with minimal signalling overhead of a centralized coordination during initialization time. Resilience to failures and tolerance to network partitions are key features of the CCBP design, ensuring that experiments can continue even if the central node becomes unavailable or disconnected. The capabilities of Cloud Continuum Blueprint (CCBP) are validated in a real deployment environment, demonstrating its ability to adapt to different hosting conditions, highlighting its adaptability to large-scale deployments and resource-constrained ones.

Acknowledgements

This work is supported by the European Union - NextGenerationEU - National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) - Project: ”SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics” - Prot. IR0000013.

References

[1] L. Bittencourt *et al.*, “The Internet of Things, Fog and Cloud continuum: Integration and challenges,” *Internet of Things*, vol. 3-4, pp. 134–155, 2018.

[2] S. Fdida *et al.*, “Slices, a scientific instrument for the networking community,” *Computer Communications*, vol. 193, pp. 189–203, 2022.

[3] Slices-Blueprint. Mobile Middleware Research Group. Accessed: Apr. 16, 2025. [Online]. Available: https://gitlab.com/MMw_Unibo/platformeng/slices-blueprint

[4] D. Rosendo, P. Silva, M. Simonin, A. Costan, and G. Antoniu, “E2Clab: Exploring the Computing Continuum through Repeatable, Replicable and Reproducible Edge-to-Cloud Experiments,” in *Proc. of IEEE International Conference on Cluster Computing*, 2020, pp. 176–186.

[5] D. Duplyakin *et al.*, “The Design and Operation of CloudLab,” in *Proc. of USENIX Annual Technical Conference (USENIX ATC 19)*. Renton, WA: USENIX Association, Jul. 2019, pp. 1–14.

[6] F. B. Oliveira, M. Di Felice, and C. Kamienski, “IoTDeploy: Deployment of IoT Smart Applications over the Computing Continuum,” *Internet of Things*, vol. 28, p. 101348, 2024.

[7] L. Osmani, T. Kauppinen, M. Komu, and S. Tarkoma, “Multi-Cloud Connectivity for Kubernetes in 5G Networks,” *IEEE Communications Magazine*, vol. 59, no. 10, pp. 42–47, 2021.

[8] E. Song *et al.*, “Canal Mesh: A Cloud-Scale Sidecar-Free Multi-Tenant Service Mesh Architecture,” in *Proc. of the ACM SIGCOMM 2024 Conference*. New York, NY, USA: Association for Computing Machinery, 2024, p. 860–875.

[9] J. Santos *et al.*, “HephaestusForge: Optimal Microservice Deployment Across the Compute Continuum via Reinforcement Learning,” *Future Generation Computer Systems*, vol. 166, p. 107680, 2025.

[10] M. Patiño Martinez and A. Azqueta-Alzúaz, “Computing Continuum in Heterogeneous Multi-clusters with Resource-Constrained Devices,” in *Proc. of the Workshop on Middleware for the Computing Continuum*, ser. Mid4CC ’24. New York, NY, USA: Association for Computing Machinery, 2025, p. 7–12.

[11] T. Goethals, F. De Turck, and B. Volckaert, “Extending kubernetes clusters to low-resource edge devices using virtual kubelets,” *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2623–2636, 2020.

[12] 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.

[13] F. Faticanti, D. Santoro, S. Cretti, and D. Siracusa, “An Application of Kubernetes Cluster Federation in Fog Computing,” in *Proc. of Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2021, pp. 89–91.

[14] L. Larsson, H. Gustafsson, C. Klein, and E. Elmroth, “Decentralized kubernetes federation control plane,” in *Proc. of IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2020, pp. 354–359.

[15] Y. Demchenko, S. Gallenmüller, S. Fdida, T. Rausch, P. Andreou, and D. Saucez, “SLICES Data Management Infrastructure for Reproducible Experimental Research on Digital Technologies,” in *Prof. of IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 1–6.

[16] J. Dobies and J. Wood, *Kubernetes operators: Automating the container orchestration platform*. O’Reilly Media, 2020.