Best Practices for Resource Provisioning Declaration Within the Cognitive Cloud Continuum

Sergio Moreschini
Tampere University, University of Oulu
Tampere, Finland
sergio.moreschini@tuni.fi

Michele Albano
Aalborg University
Aalborg, Denmark
mialb@cs.aau.dk

David Hästbacka
Tampere University
Tampere, Finland
david.hastbacka@tuni.fi

***Abstract*—The evolution of cloud computing, driven by advances in mobile, edge technologies, and AI, has led to the development of the Cognitive Cloud Continuum (COCLCON). However, this paradigm introduces new challenges in managing and optimizing computing resources across a heterogeneous environment. This paper explores best practices for declaring resources within COCLCON, with a focus on efficient resource allocation and transparently declaring available resources by devices. In this study, we undertook a non-holistic literature review to identify current technologies used to specify requirements and to determine current gaps in best practices. The main outcome of our work is a proposed schema for Resource Provisioning Declaration, which will allow for increased knowledge related to the available devices and resources within the COCLCON.**

***Index Terms*—Kubernetes, Ontology, Serverless, Resources**

I. Introduction

Since its inception, cloud computing has been a valuable option for those in need of large amounts of computing resources and flexibility. Nevertheless, recent developments in mobile and edge device development, as well as advances in artificial intelligence (AI), have led to a shift to a more decentralized model, resulting in reduced latency in accessing information. Consequently the concept of cloud computing has evolved into the Cloud Continuum [1]. Ideally, we would like the devices to be capable *"of sensing the environment, learning from it, and opportunistically and dynamically adapt the computational load as well as their outcome"* [2]. A cloud system satisfying these requirements is defined as Cognitive Cloud, and by extension applying this concept within the Cloud Continuum results in the Cognitive Cloud Continuum (COCLCON). In the COCLCON, it is crucial to have *knowledge* of the available resources of each constituent device at all times, therefore, these devices should openly declare the resources available for sharing.

This study aims to identify and refine current best practices for declaring the available resources within the Cognitive Cloud Continuum. Due to page constraints, this review provides a concise and focused examination of the relevant literature, rather than a comprehensive analysis. The contribution of the work is the identification of different approaches for the definition of resource requirements for the main tools used to build the environment, and the gap created by them. As a result, we also proposed a scheme for the declaration of resource provisioning.

The rest of this work introduces background information regarding the cognitive cloud continuum (Section II), then presents the research questions targeted by the paper and our research methodology (Section III). Later on, existing specification schema discovered during the research process are presented (Section IV), our proposed schema for resource declaration is presented and discussed (Section V), and conclusions are drawn on the topic at hand (Section VI).

II. Background and Related Works

A. Challenges in the COCLCON

The COCLCON paradigm offers new ways to manage computing resources by allowing for computational load sharing among devices, both horizontally (edge-to-edge) and vertically (edge-to-fog or edge-to-cloud). It is challenging to optimize resource use in these diverse environments, especially with the rise of serverless computing [3].

Efficiency for both infrastructure use and energy consumption are two critical points for systems which need to be always ready for computation [4]. An example of critical system is stream reasoning where devices are continuously processing and analyzing data streams in real-time at the edge of the network [5]. In this scenario we can and must apply resource declarations to virtual devices in simulations, which will enhance computational efficiency when applied to real devices in the COCLCON by considering factors like energy availability and transmission rates. Resource declaration methods must be fully defined, independently of sniffers. This is essential for exploiting the capabilities of the COCLCON and enabling real-time conditioning offloading, based on external factors such as available energy, average transmission rate and load trends [6].

B. Similarities with the Semantic web

The definition of resource requirements in a complex edge-cloud continuum faces significant challenges in terms of expressiveness and computational complexity in order to match performance requirements with available resources. This problem is a reflection of historical challenges in systems engineering, in particular those that were encountered in the development of the semantic web [7], whose goal was to enable distributed, interoperable data by associating metadata with data and creating a general, machine-readable ontology.

Similarly, COCLCON requires a a strategic methodology for resource description to achieve interoperable computing resource sharing across vendors and industries [8].

A key difference is that COCLCON resources can provide computational power to facilitate their own matching processes, whereas the Semantic Web relied on third-party computational power to process metadata [7]. Thus, modern approaches must move from a centralized orchestration schema to distributed intelligence or, to use a modern term, to advance towards a COCLCON.

C. Limitations of a Centralized Orchestration

While centralized orchestration brings control and simplicity to the management of distributed computing environments, it comes with a number of drawbacks. Specifically, when comparing this approach to offloading, there are 4 issues that have a negative impact on the performance of centralized orchestration: Flexibility, Latency, Resource Efficiency, and Scalability [9].

In terms of flexibility, offloading is the clear winner over centralized orchestration within the COCLCON. Its dynamic and adaptive nature makes it the superior choice. About latency, being able to adapt to changing conditions becomes meaningless without the ability to respond in real time. As the use of resources has a significant impact on QoS. The existing literature on resource optimization for orchestration proposes several algorithms [10]. However, even with this method, resources may remain unoptimized because some devices may still remain available while others are overloaded. At last, as the orchestrator is the main responsible for communication management and resource allocation. Overloading the orchestrator will lead to latency, poor resource allocation, and a reduction in QoS.

D. The dichotomy between Open Source and Proprietary Solutions

In the development of a computing environment, the decision on the type of resources to be used is of crucial importance. Because of its flexibility, interoperability, and accessibility, open source software (OSS) is the preferred choice for real-world resources. Driven by community innovation, OSS offers a high degree of customization and hybrid solution capabilities while reducing costs, especially for smaller organizations.

Conversely, due to their advantages, virtual environments often rely on proprietary solutions. Cloud providers offer robust customer support, reliability, advanced functionality and enhanced security, which makes them ideal for projects requiring industry standards and regulations. These benefits ensure a comprehensive and supported solution throughout the project lifecycle, justifying the costs associated with cloud services [11].

III. Research Method

The goal of this work is to identify and extend current best practices for establishing resource declaration within the Cognitive Cloud Continuum. We defined the following Research Questions:

- **RQ1:** *What are the different technologies used to specify requirements within the Cognitive Cloud Continuum?*
  In this RQ we aim at understanding what are the current resource provision declaration methods for different tools independently of their nature of being an open source or a proprietary solution.
- **RQ2:** *Which gaps emerge in regards to current best practices?*
  The purpose of this RQ is to investigate any uncovered areas after understanding the current state of the resource provisioning statement.

To answer the research questions, due to the limited amount of valuable results, we performed a non-holistic literature review. The main aim was to impartially summarize and classify the information collected to catalogue the different best practices for resource declaration within the COCLCON and in particular for edge devices. The original research string ***"resource specific" languages methods*** led to a plethora of results using Google (i.e. 5.300.000 results within 0.37 seconds on February 20th 2024). A similar search on Medium and Stack Overflow yielded similar results. We did an initial convergence check, which failed, so the search string was refined to get more relevant results, achieving different keywords. At first with the inclusion of the word ***yaml***, results related to Azure and Kubernetes began to appear.

For this reason a new search with the keywords ***yaml edge resources*** has been performed. This specific search has pointed to at least one page for all the main Proprietary Solutions (Azure, Google Cloud Platform and AWS CloudFormation) but also to pages related to Kubernetes. The results are presented and discussed in Section IV-A

IV. Analysis of the Results

The performed research showed that different platforms rely on similar templates, but the specific declarations differ for each.

A. RQ1. What are the different technologies used to specify requirements within the Cognitive Cloud Continuum?

When focusing on the different technologies used to specify requirements it is important to understand how different tools accept specific resource provisioning declaration.

AWS CloudFormation templates, written in YAML or JSON, consist of 10 parts [12]. These parts range from version and description to resources and outputs. The resource section includes an ID, type, and properties. The key advantage of this template approach is its customizability. This allows users to add custom resources if the default types are insufficient. This flexibility makes CloudFormation highly adaptable to specific needs [13].

Microsoft Azure uses the management service, Azure Resource Manager (ARM), which implements Infrastructure as Code (IaC) through ARM templates [14]. These JSON-based templates define infrastructure and configuration and convert

to REST API operations during deployment. ARM templates ensure repeatable results with declarative syntax and manage resource orchestration, including parallel deployments where possible. Azure's new language, *Bicep*, simplifies writing resource declarations, but ultimately, Bicep code is converted to ARM templates for deployment.

Google Cloud Platform (GCP) organises resources in a clear and logical hierarchy [15]. This allows the user to manage ownership, attach points, and inheritance. The top of the hierarchy usually consists of the organisation, followed by projects and folders. To automate hierarchical infrastructure of GCP, the external tool Cloudify is the best option [16]. Cloudify provisions and manages cloud resources across different orchestration domains. It is possible to create custom resources by utilising workarounds such as the Kubernetes Operator Config Connector, which facilitates resource management through Kubernetes [17].

Kubernetes is an open-source platform that automates the operational processes associated with containerized applications, including the deployment, scaling, and orchestration of these applications across clusters of containers. Operations are mainly executed via a command-line interface using APIs [18]. The OpenAPI specification defines the resources required by containers, allowing for the specification of the CPU and memory needs of individual containers. Custom resources can be declared using the Kubernetes device plugin framework, such as specifying an NVIDIA GPU requirement with *nvidia.com/gpu.*

It is also important to mention that well-known Infrastructure as Code frameworks like **TOSCA** and **DOML**, which are primarily designed for managing cloud infrastructure and virtualized resources, are now focusing on IoT/Edge/Fog environments to explore new opportunities [1].

Within this group of tools Kubernetes is the one showing the highest level of customization and due to its open source nature it is also the one that is mostly used for what concerns offloading and orchestration. For this reason our focus will be placed on this tool.

B. RQ2. Which gaps emerge in regards to current best practices?

The non-holistic literature review reveals a significant gap: while it is possible to define and limit the resources available at a given time, there is no unified schema for devices to define their available resources at a given point in time universally, particularly when involving energy utilization within the COCLCON.

As an example, Kubernetes does not inherently support energy consumption management, necessitating indirect methods to achieve this goal. One viable approach is to use device plugins, similar to those used for GPU resource declarations [19], to provide information on their current energy levels and usage patterns, including the available battery capacity and current consumption. Such data may then be employed by the device to conduct a comprehensive analysis of completed tasks. This analysis may provide a framework within which to contextualize future tasks and to make informed predictions regarding the potential outcomes of these future tasks.

To this aim, it should be possible to connect directly to devices and receive a statement with all the necessary information, including the computational power, available support, available energy, and an estimation of when it will become available if it is currently engaged in another process.

V. Discussion

To our knowledge, no other work has focused on the importance of declaring resources in the edge layer of the COCLCON. In environments where edge devices are independent and willing to join networks, it is important to protect their privacy, as well as reduce network traffic, by reducing the use of sniffer tools.

The importance of defining an appropriate resource declaration system within COCLCON is highlighted by the results presented in the previous section. According to this, the main goal of such a declaration system is to optimize the use of computing resources and to allow the partitioning of tasks in a *trigger- and action-oriented way*.

A similar concept is provided by serverless computing which is defined as a software architecture where *the applications are decomposed into events and functions* and the platform seamlessly hosts and executes the environment [20]. The exposure of available resources allows applications to scale efficiently and to predict costs in a pay-as-you-go model. The application of this approach to COCLCON can optimize the use of real-world resources by understanding the minimum or recommended number of devices needed, providing not only resource efficiency (i.e. Resources-as-a-continuum) but also devices efficiency (i.e. Devices-as-a-continuum) [21].

An example is a set of devices running Federated Learning algorithms [22]. In this collaborative approach, multiple clients work together on machine learning tasks, coordinated by a central server, while maintaining data privacy through decentralization. Devices train locally on their data and send models to a central server for aggregation. The aggregated model is then sent back to the devices to continue learning. Such framework is therefore possible only by making use of an architecture which could be supported by all the devices within the environment. All the available devices within this framework would deploy a similar yet unique version of a system previously trained on fog devices within the same COCLCON.

Following, we should expect that different devices within a specific environment should provide, when requested, a declarative log with some specific information related to their available resources.

A. The Proposed Schema

We therefore propose that different elements within the COCLCON should be able of generating a **Resource Provisioning Declaration.** This resource declaration should be

[1]OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC

generated both as a *YAML* file or *JSON* file and composed of specific and well known fields. An example of the proposed structure is described in Listing 1 as a YAML file.

```yaml
apiVersion: v1
kind: Edge
metadata:
  name: EdgeDeviceName
spec:
  containers:
  - name: my-container
    image: myimage:latest
    resources:
      specs:
        cpu: "x"
        memory: "x"
        energy: "x"
        gpu-vendor.example/example-gpu: "x"
      available:
        cpu: "x"
        memory: "x"
        energy: "x"
        gpu-vendor.example/example-gpu: "x"
        nextUpdate: "x" #in seconds
```

Listing 1: YAML file for Resource Provisioning Declaration

The ontology is a fundamental component of the COCLCON framework, serving as a conduit for the provision of information about the devices that comprise it. This information is made available without the need for the use of specialized software, such as sniffer tools. The proposed ontology focuses on resource provision declaration on two different perspectives: what it can be available and what is available right now. This viewpoint is also reinforced by the ability to determine the timing of the next significant update in the resource declaration.

VI. Conclusion

In this paper, we propose a schema for the declaration of available resources for all devices in the Cognitive Cloud Continuum, from edge devices to cloud platforms. We started with a holistic literature review to identify the current resource declaration, which led to answering the two main research questions identified: What are the different technologies in use for requirements specification, and what are the current best practices?

In the first case, we found that various platforms use templates that are similar but not identical. Moreover, open source tools such as Kubernetes allow to customize missing resource through device plugin frameworks. In the second case, we realized that there are still some gaps which needs to be address. The most important of these is, in the context of the Cognitive Cloud Continuum, energy declaration. For these reasons, the Cognitive Cloud Continuum has been compared to serverless architectures and a schema for resource provisioning declaration has been proposed.

Acknowledgment

This work is funded by the IndustryX and 6GSoft projects (Business Finland), and by the Villum Investigator Project "S4OS: Scalable analysis and Synthesis of Safe, Small, Secure and Optimal Strategies for Cyber-Physical Systems".

References

[1] S. Moreschini, F. Pecorelli, X. Li, S. Naz, D. Hästbacka, and D. Taibi, "Cloud continuum: The definition," *IEEE Access*, vol. 10, pp. 131 876–131 886, 2022.

[2] S. Moreschini, F. Pecorelli, X. Li, S. Naz, M. Albano, D. Hästbacka, and D. Taibi, "Cognitive cloud: the definition," in *International Symposium on Distributed Computing and Artificial Intelligence*. Springer, 2022, pp. 219–229.

[3] K. Govindarajan and A. D. Tienne, "Resource management in serverless computing - review, research challenges, and prospects," in *2023 12th International Conference on Advanced Computing (ICoAC)*, 2023, pp. 1–5.

[4] S. Baneshi, A.-L. Varbanescu, A. Pathania, B. Akesson, and A. Pimentel, "Estimating the energy consumption of applications in the computing continuum with ifogsim," in *International Conference on High Performance Computing*. Springer, 2023, pp. 234–249.

[5] D. Dell'Aglio, E. Della Valle, F. van Harmelen, and A. Bernstein, "Stream reasoning: A survey and outlook," *Data Science*, vol. 1, no. 1-2, pp. 59–83, 2017.

[6] K. Sheshadri and J. Lakshmi, "Qos aware faas for heterogeneous edge-cloud continuum," in *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*. IEEE, 2022, pp. 70–80.

[7] A. Rhayem, M. B. A. Mhiri, and F. Gargouri, "Semantic web technologies for the internet of things: Systematic literature review," *Internet of Things*, vol. 11, p. 100206, 2020.

[8] TerminusDB, "The semantic web is dead – long live the semantic web!" https://terminusdb.com/blog/the-semantic-web-is-dead/, 2022.

[9] A. Megargel, C. M. Poskitt, and V. Shankararaman, "Microservices orchestration vs. choreography: A decision framework," in *2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE, 2021, pp. 134–141.

[10] A. Droob, D. Morratz, F. L. Jakobsen, J. Carstensen, M. Mathiesen, R. Bohnstedt, M. Albano, S. Moreschini, and D. Taibi, "Fault tolerant horizontal computation offloading," in *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*, 2023, pp. 177–182.

[11] E. Truyen, D. Van Landuyt, D. Preuveneers, B. Lagaisse, and W. Joosen, "A comprehensive feature comparison study of open-source container orchestration frameworks," *Applied Sciences*, vol. 9, no. 5, p. 931, 2019.

[12] A. CloudFormation, "Template anatomy," https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html, 2024.

[13] ——, "Custom resources," https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-custom-resources.html, 2024.

[14] M. Learn, "What are arm templates?" https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/overview, 2023.

[15] G. C. Platform, "Resource hierarchy," https://cloud.google.com/resource-manager/docs/cloud-platform-resource-hierarchy, 2023.

[16] C. Platformm, "Cloudify," https://cloudify.co, 2022.

[17] G. C. Platform, "Config connector overview," https://cloud.google.com/config-connector/docs/overview, 2024.

[18] T. K. Authors, "Kubernetes api," https://kubernetes.io/docs/concepts/overview/kubernetes-api/#openapi-v3, 2024.

[19] ——, "Schedule gpus," https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/, January 2024.

[20] A. P. Rajan, "A review on serverless architectures-function as a service (faas) in cloud computing," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 1, pp. 530–537, 2020.

[21] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for internet of things: a primer," *Digital Communications and Networks*, vol. 4, no. 2, pp. 77–86, 2018.

[22] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.