EDITOR: Schahram Dustdar, dustdar@dsg.tuwien.ac.at

DEPARTMENT: INTERNET OF THINGS, PEOPLE, AND PROCESSES

Active Inference for Sustainable Computing Continuum Systems

Juan Luis Herrera, Boris Sedlak, and Schahram Dustdar, *TU Wien, 1040, Vienna, Austria*

The computing continuum (CC) is expected to improve the quality of service of distributed applications. However, adding more devices to current cloud datacenters to implement the CC can increase the energy-related CO_2 emissions of the paradigm. Sustainability requires intelligently managing applications through the CC, considering both the energy consumption and carbon intensity of energy sources. Moreover, the CC provider should perform this management, given their control of devices and networks. This work provides an overview on achieving sustainability in the CC and the associated challenges, proposing an architecture that leverages active inference to learn, manage, and reconfigure the system to achieve a low carbon footprint while meeting the service level objectives of applications.

The interest in the computing continuum (CC) as a computing paradigm has been growing intensely in recent years, with market projections of US$380 billion in spending by 2028.[1] The distributed nature of the CC enables application services to be executed where they best fit, be it closer to users or in more powerful data centers, promising to improve their quality of service (QoS).[2] As a result of this expected increase in QoS, companies are increasingly interested in bringing key services, such as web services,[3] media streaming,[4] or artificial intelligence (AI),[5] traditionally seen as cloud services, to the CC.

However, with the enhancement of QoS in the CC comes also the challenge of energy consumption and, with it, greenhouse gas emissions. Cloud computing is responsible for 0.9% of the energy-related greenhouse gas emissions.[6] The cloud is generally deployed as a set of servers expected to be available ubiquitously and on demand,[7] requiring constant energy. If the CC followed suit, it would be deployed as a set of additional devices that are permanently powered on. Such a deployment could further increase this figure,[8] as the cloud would keep its current deployment, and the CC would increase it. Traditional energy optimization leveraged in the cloud is hard to apply on the CC,[8] on the one hand, due to its distributed nature, and on the other hand, as services requiring hardware acceleration become more widespread and harder to migrate.

This increase in emissions of CO_2-equivalent gases, however, is directly in opposition to the ongoing interest in sustainable computing, such as green computing,[9] as well as incentivized by governments, as in the EU Green Deal or the Sustainable Development Agenda. Thus, reducing the carbon footprint of CC is a crucial challenge to be tackled.

To address this challenge, the factors that influence the carbon footprint of the CC are the energy consumption of the infrastructure and the carbon intensity of the energy sources. The CC provider can control the energy consumption of the system by performing reconfigurations to the software and hardware running in the infrastructure (e.g., limiting resources or disabling hardware acceleration). Carbon intensity is governed by the specific energy source (e.g., coal, solar, wind) or the *energy mix* in case multiple sources were leveraged, which are not directly in control of the provider. Nonetheless, as the CC is distributed, different nodes can have different carbon intensities and

Digital Object Identifier 10.1109/MIC.2025.3590621
Date of current version 17 March 2026.

hence reduce the overall carbon footprint by moving services toward devices with a lower carbon intensity.

Managing the infrastructure to reduce the carbon footprint, however, is not a simple endeavor. Developers leverage the CC to get high QoS, meeting their service-level objectives (SLOs). There is, however, an important gap between the requirements of developers, with QoS-oriented SLOs, and those of CC providers, such as SLOs on the carbon footprint of their infrastructure. Performing reconfigurations is especially challenging, as CC providers have limited visibility to the services running. The reconfigurations to save energy taken by the CC provider can decrease the QoS, thus requiring careful balance to meet both developers' and CC provider's SLOs. Moreover, the reconfigurations available can vary across machines in the CC (e.g., not all nodes integrate variable thermal design power (TDP), powering off a device can be risky if no lights-out management exists), and the effects of such reconfigurations in both SLOs and energy consumption can also vary across machines. These effects depend on how the services use the resources provided and are complex to determine a priori, requiring *learning* about them.

As shown by Sedlak et al.[10] and Sedlak et al.,[11] active inference (AIF) can be used for developing structural knowledge about a system's behavior given certain actions, e.g., providing the expected energy consumption after moving or reconfiguring services. For this, AIF employs a continuous action–perception cycle, in which it balances the exploration of unknown but insightful configurations, with such that promise to fulfill the configured SLOs.

This work includes its study in five realistic use cases, and the architecture of a framework to leverage AIF from the CC provider's perspective. AIF is expected to benefit developers as well as reducing the carbon footprint of the CC.

BACKGROUND AND CHALLENGES

Cloud computing has been the reference computing paradigm for the last decade,[2] in part, due to its model in which computing capacities are provided *as a service* and paid for based on their usage.[7] This model removes the need for an initial investment from developers, while also automatically scaling the computing capacity.[1,2,7] However, as cloud computing became more dominant, the limitations imposed by its centralization also became more apparent: high network latencies, privacy concerns as data passes through borders, and lack of locality. While acceptable for some applications, these limitations made cloud computing infeasible for others.[2] In recent years, the CC emerged as a paradigm aimed at keeping the model of cloud computing, while leveraging distribution to address its shortcomings.

In this context, one of the most interesting movements in the cloud computing space is the *green cloud,*[9] which tries to ensure the sustainability of cloud computing. On the one hand, if the CC is the evolution of cloud computing, it is natural for it to also pursue sustainability. On the other hand, the interest in bringing sustainability to both computing and networking systems also stems from governmental incentives. The main metric for the sustainability of a computing system is its *carbon footprint*, i.e., the mass of CO_2-equivalent gases generated per unit of time by the system due to its energy consumption.[12] Traditionally, sustainable computing was synonym with energy-efficient computing,[13] as lower energy consumption correlates to lower carbon footprints. Nonetheless, cloud providers recently noticed that the carbon intensity (i.e., mass of carbon produced by unit of energy consumed) of the energy source, or the mix of energy sources used to power the computing devices is equally important.[12] Google Cloud schedules energy-consuming tasks to moments when the carbon intensity of the energy mix is predicted to be low.[12] The CC provides an additional dimension as a distributed paradigm: Services can be moved to a different point in the CC where the energy mix is less carbon intense. However, this dimension comes with the challenge of the high complexity in managing and orchestrating distributed services.

There are initial proposals to address this challenge that frame it from the point of view of application developers or operators,[14] treating sustainability or energy consumption as another aspect to optimize. However, application developers only have control and knowledge over their application, not the whole infrastructure, which can lead to different developers making decisions that are optimal individually but not globally. For example, 10 developers may decide to deploy a service into 10 very energy-efficient fog nodes, but it would be preferrable to deploy all 10 services in a single node. Moreover, solving these issues from the point of view of developers is complex: Not only would they have to share information with other developers, potentially their business competitors, about their services, their solutions would need to be ready to consider other applications they have no control over and that may be managed in unpredictable ways.

Instead, we believe this problem should be framed from the perspective of CC providers, who have knowledge and control of the services deployed. A CC

provider is expected to have full control over the cloud and fog layers, as modern cloud providers do. Devices closer to users, such as edge devices, may not be under the direct control of the CC provider, but are expected to execute an integration agent from the CC provider, like the current Google Distributed Cloud Edge or AWS Greengrass. This agent allows the CC provider to control these nodes as well, while it allows the owner of the device to be integrated with the CC provider's ecosystem. If the owner of the device is the developer, the agent provides them with automated and intelligent management of their application, while owners such as edge providers may benefit from partnerships with CC providers.

However, this perspective brings a key challenge: CC providers lack information about the services running in their infrastructure. While developers know what the services of their application are, for the CC provider, they are all seen as opaque services. Like sharing details across service vendors, sharing details about services with the CC provider can be problematic in terms of privacy. Hence, the CC provider cannot know what exactly is running and must operate based on the limited information the developers provide.

In summary, to address sustainability in the CC, it is necessary to consider the following challenges:

C1) *Consideration of the carbon footprint*, including both energy consumption and carbon intensity.
C2) *Knowledge and control over the CC infrastructure*, understood as the ability to know the status of and reconfigure technical characteristics of devices of the CC, such as powering them off or disabling their GPU.
C3) *Knowledge and control over the services*, including their SLOs, characteristics, reconfiguration, scheduling, and migration.
C4) *Management of conflicting SLOs*, both between developers and CC providers (e.g., high QoS versus low carbon footprint) and between different developers (e.g., one service requests CPU load above 90% and another one requests to stay below 50%).
C5) *Privacy respecting*, meaning that the parties involved must not need to disclose any information they do not wish to disclose to the parties in control.
C6) *Consideration of the peculiarities of the CC*, including distribution or heterogeneity.
C7) *Adaptive behavior* to schedule, configure, and migrate services depending on environmental variables such as the carbon intensity of the energy mix.

EXAMPLE USE CASES

To better illustrate the benefits of this approach, we present a set of five representative use cases that would benefit from it. Noteworthily, they are not aimed at being an exhaustive list, but a representative sample of a wider variety of use cases.

Content Management System (CMS)

A CMS is a software service to simplify the administration of websites. Although some CMSs specialize in one type of site, such as MyBB for forums, CMSs such as WordPress can be leveraged for different types of websites. Distributing and replicating a CMS throughout the CC can enhance the availability of the website and its performance,[3] and thus, CMS may be interested in leveraging the CC.

CMSs are mainly storage-intensive and easy to scale down. However, they are sensitive to increases in traffic, which require replicating and scaling up the application to avoid performance loss. Managing traffic increases requires anticipation, which can be achieved by learning the patterns behind demand. Moreover, different types of websites have different SLOs, regardless of the CMSs that are leveraged: Forums host user-created content, while the content of a news website is generally managed by a single party, and shopping websites require careful synchronization during the buying process.

The need to learn the behavior of CMSs in the CC makes AIF an adequate technique to improve their sustainability. As websites are very flexible, CC providers have no knowledge of what type of website is being hosted. Learning how the website operates and how actions on the hardware or container configurations may affect them are crucial to decreasing the carbon footprint of the system. Spatiotemporal demand peaks can also be learned and adjusted for ahead of time. Interestingly, if the interactions among services, hard to determine a priori, are learned, adjustments can be made only to the services most affected by peaks. The sustainability of highly energy-consuming periodic maintenance tasks, such as backups, can be improved by scheduling them during off-peak times for the website, not affecting its QoS, and at points where the energy mix is least carbon-intense.

Multimedia Streaming Services

Services that allow users to stream multimedia content to their devices on demand are nowadays commonplace. Netflix, Spotify, or YouTube are examples of such services. These services have traditionally

been an example of distributed systems due to their use of content delivery networks (CDNs)[4]: distributed networks of devices closer to users that stored copies of their multimedia content to improve the QoS of the service. As such, the developers of multimedia streaming services are greatly interested in leveraging the potential of the CC to further expand their CDNs.[4]

Multimedia streaming services are very intensive in terms of storage, as they store high-quality video and audio, and in network bandwidth to stream the content. Many of these characteristics are dependent on the specific type of content offered: Different multimedia types have different requirements (e.g., audio requires less bandwidth than video), and even different types of content do (e.g., 2-hour-long films require vastly more storage space than 30-second-long short videos). Moreover, these services can trade off storage space and processing power: Multimedia content can be stored in multiple qualities or transcoded ad hoc to a lower quality when requested. Depending on the source and target qualities, transcoding may be performed directly on the CPU or require GPU acceleration. All these characteristics are also greatly affected by content popularity: Transcoding popular videos for millions of users requires vastly more power than transcoding videos for hundreds or dozens of users. Similarly, requiring dozens of users to stream their content from a central server rather than the CDN for their content is more acceptable than doing so for millions.

Therefore, learning about the behavioral characteristics of these services, such as storage, bandwidth, CPU and GPU intensity, or the demand, is also crucial for sustainable multimedia streaming. Developers already learn about the popularity of content to populate their CDNs with content, but they may not share them with the CC provider for privacy reasons. Nonetheless, learning about their SLOs at the CC provider level could ensure hardware acceleration and CPU resources can be adequately managed to reduce the energy consumption of CDNs. Due to the inherent role of networks in multimedia streaming, learning about the QoS offered to different users in different locations of the CC can also be used to move CDN nodes to less carbon-intense locations. As the CC offers full support for service execution and not simply storage, if the network is powered by a significantly more carbon-intensive energy mix than the CC devices, it is also possible to only migrate the high-quality multimedia files and populate lower quality versions by transcoding them in situ. Learning about the energy mix patterns would allow predicting periods with a low-carbon energy mix to schedule these tasks to them. Overall, the need to learn in multimedia streaming is patent, making AIF an interesting approach to these challenges.

Remote Gaming

Remote gaming is a service in which videogames are installed in remote computing devices, such as cloud instances, fog or edge nodes, and livestreamed to a user device, generally assumed to be less powerful, and who can control it remotely.[15] Among others, Xbox Cloud currently offers this service. Remote gaming is expected to bring high-quality gaming capabilities to virtually every device with a display, from mobile phones to TV screens, without the need for specialized hardware.

In this use case, high latency can result in perceived unresponsiveness, making the game unplayable in practice. As videogames often run at 30, 60, or even 120 frames per second, streaming the image to the user, as well as receiving the inputs from the user, in as little as 8 ms is challenging. This has brought forth increased interest in deploying remote gaming services to the CC, but it has also made it necessary to learn how the management of these services affects their latency. Moreover, the resource-intensity of videogames can vary greatly: Some require significant memory, CPU and GPU, while others can easily run in underpowered devices. Similarly, some games are played in short sessions, while others are played for a longer time.

Learning the latency SLOs, resource intensity, or session length patterns, all of which are difficult to know a priori, can be of great importance for a greener remote gaming ecosystem. Latency can be ensured, while also aiming for a lower carbon footprint within possibility. Videogames that require more resources, and hence, have higher energy consumption, can be placed in less carbon-intensive, or more energy-efficient devices. Learning from the patterns in which these games are played is crucial to set a preference for those involving longer sessions to be placed in nodes with a smaller carbon footprint. It is also interesting to learn how limits to the resources these games can access, such as lowering the TDP if the device allows it, affect their performance. AIF can be used to learn and perform such intelligent management of remote gaming services in the CC.

Large AI Systems

The use of AI systems is constantly becoming more widespread, especially with large AI models such as large language models (LLMs).[5] Generative AI in general is also being offered for a variety of services, such

as image and video generation, editing, reinterpretation of images in given styles, or voice imitation. Some services even combine these generative AI services with other AI-based functionalities like image recognition or optical character recognition. Nonetheless, large AI systems are also famous for their large energy consumption and carbon footprint.[16] Sustainability is thus an important focus for large AI systems,[16] and the CC is key for reducing the carbon footprint of these systems.

From the perspective of a CC provider, large AI systems are some of the most intensive services in CPU, GPU, and storage. Nonetheless, not all the models are equal, and even modern LLMs have multiple editions with different numbers of parameters and hardware requirements. Moreover, some large AI models can work with limited resources, even without GPU acceleration, although at a much slower pace. Learning the effects that resource limits have over large AI models allows CC providers to set reasonable limits that preserve energy without greatly impacting the SLOs of the system. In this respect, the type of model is also important: If images or video are generated, rather than text, which the developer may not explain to the CC provider, it is key to learn if, e.g., bandwidth limits affect SLOs. Finally, the patterns on how the model is used can be learned and leveraged to adjust the limits to the expected load, as well as to move the services to less carbon-intensive devices.

Remote Workspaces

Tools such as GitHub Codespaces allow developers to have flexibility in the environment they use. If services are moving toward the CC, such as in the previous four examples, so are the spaces in which they are developed.[15] Collaborative programming, such as pair coding, as well as previews of development environments, can all be provided more easily using remote workspaces, becoming another important service in the CC.[11]

Remote workspaces are critical learning points for CC providers, as the confidentiality of the software developed in them is a crucial requirement. On the one hand, it is key to learn about how the technological stacks must be provisioned: Storing a set of prebuilt workspaces in the CC requires a separate storage system from which they can be downloaded on demand, while building them ad hoc requires time in each workspace, which may be more energy consuming and carbon intensive if not managed correctly. Use patterns and predictions on them are critical pieces of information to ensure sustainable provisioning. Moreover, the software being developed and tested in these workspaces can be of many kinds, and their characteristics are not known a priori, leaving learning as the only possibility to ensure their sustainable management.

FRAMEWORK ARCHITECTURE

To achieve this vision of a sustainable CC, it is necessary to intelligently manage, adapt, and reconfigure services and the CC infrastructure.

Example Scenario

Figure 1 shows a simplified and small example CC infrastructure, with three edge nodes, two fog nodes, and a cloud data center, hosting services from the previous five use cases. To illustrate the view of the CC provider, the color of each service represents the developer who deployed it, but there is no information about which color represents which use case. Edge nodes A and B are powered by coal, while C leverages wind. Each node in the infrastructure also has a different energy efficiency rating, ranging from A (best) to F (worst). In the case of edge nodes, they have A, F, and D efficiencies, respectively. Node A can partially offset the high carbon intensity of coal with its high energy efficiency, thus having a moderate carbon footprint. In contrast, B has both high carbon intensity and energy consumption, resulting in a high carbon footprint. C, despite its low energy efficiency, achieves a small carbon footprint due to the low carbon footprint of wind power. On top of the edge nodes, we find fog nodes D and E, with B and C energy efficiencies, respectively, and powered by the energy mixes of Austria and Hungary, with a moderate carbon footprint each. Finally, the cloud takes energy from an energy mix akin to the average one in the European Union, and has an A-level energy efficiency, resulting in an overall low carbon footprint.

Given this information, the reconfiguration agent acts accordingly to reduce the overall carbon footprint while respecting the SLOs of the hosted services. In A, a service from the yellow application that was momentarily scaled up is removed. Together with this removal, the CPU resources of the node are limited, bringing down the QoS. Nonetheless, as now only a purple service is in the device, the lower QoS is still above the SLO. In B, a service from the blue application is moved to the rightmost edge node. This allows the agent to power off B, not needing to meet any SLOs as no services are executed there. C, in contrast, starts hosting a blue and a yellow service, migrating from other devices. These new services set a high SLO, which requires the node to raise its CPU's TDP. Reconfigurations also

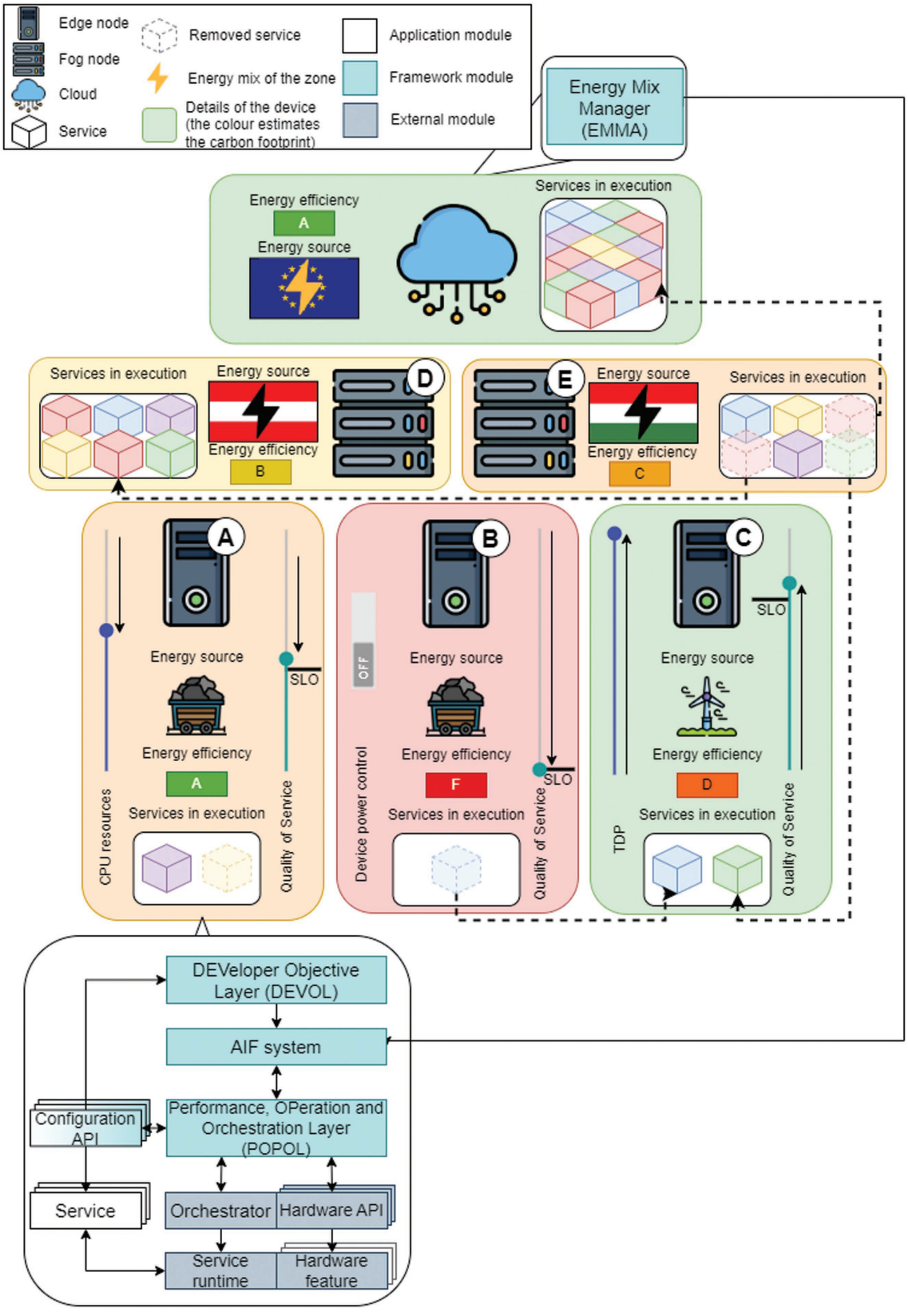

FIGURE 1. Overview of the proposed framework managing an example CC infrastructure.

exist in the fog: As the energy mix in Hungary is predicted to increase, services in E are migrated to nodes with a smaller predicted carbon footprint. Namely, a red service is moved to D, a second red service migrates to the cloud, and the third one is now hosted by C. Three services, which cannot be migrated without violating their SLO, are not moved.

Architecture Description

To perform this task, we propose a framework based on AIF, whose architecture is depicted in Figure 1. The framework considers that each device has a set of hardware features that can be adjusted, such as CPU clock speed, TDP, or whether the device is powered on. These features can be programmatically adjusted through an application programming interface (API) (e.g., simple network management protocol [SNMP], intelligent platform management interface [IPMI]). In parallel, the device is running a service runtime, such as Docker, in which services are being executed. On top of the runtime, an orchestrator such as Kubernetes performs high-level service management (e.g., migration).

The first framework proposal is that, if hardware offers APIs to be configured, services should also offer a configuration API. This API should allow for four tasks: discovery of available configuration features and QoS metrics, query of current values of each feature and metric, reconfiguration of features, and evaluation of SLOs. These APIs could easily be auto generated from a configuration files and would allow service developers to keep the nature of their service private, as parameters do not need to be humanly interpretable (e.g., the "quality" feature from the media streaming service, which can be set to 720p, 1080p, or 4K, could be expressed as feature "a," admitting integer values from 1 to 3). The framework unifies all these status queries and reconfiguration APIs, from hardware, services, and the orchestrator, into the performance, operation, and orchestration layer (POPOL). POPOL eases all tasks related to monitoring the performance of services through service-specific, orchestrator-level, and hardware-level metrics, as well as operation, e.g., reconfiguration of hardware and services, or migrations.

The AIF system continuously collects and monitors the available information, including the currently available CC devices, their (predicted) energy mixes, and the service demand. Using this information, the AIF system develops a structured generative model of the probability under which the system can fulfill its SLOs under certain states, or when taking an action from a state. This generative model allows interpreting and predicting the conditional impact of taking actions; to that extent, the AIF system pursues two goals: taking actions that improve the agent's understanding of the environment (i.e., the generative model), for example, by taking actions with less certain outcome, and taking actions that are highly likely to ensure the configured SLOs. Thus, the AIF system identifies parameters that directly impact SLO fulfillment, e.g., how to minimize emissions by proactively moving and reconfiguring services to specific nodes.

The AIF system is directly integrated with the rest of the framework by leveraging the rest of the modules. POPOL allows AIF to learn about the current configuration of hardware, individual services, and their orchestration, as well as about the performance of the device in terms of energy consumption. AIF can also invoke POPOL to reconfigure them. Carbon intensity is considered by using the energy mix manager (EMMA). EMMA provides information about the energy mix of the nodes, combining information sources such as internal information or real-time datasets.[16] EMMA also makes it possible to predict future energy mixes. The last piece of information required by AIF is the management of SLOs, handled by the DEVeloper objective layer (DEVOL) module. DEVOLA gathers information about the SLO-related performance metrics and SLO fulfillment from the APIs of the different services and unifies them for AIF, resolving conflicts between SLOs through different techniques, such as prioritization. By querying DEVOL, AIF can get information on how reconfigurations affect the overall SLO fulfillment as well as separate SLO metrics.

In terms of deployment, as depicted in Figure 1, DEVOL, the AIF system, and POPOL, as well as the configuration APIs, are designed for a distributed deployment. Every device in the CC is expected to have an instance of DEVOL, POPOL, and the AIF system, while each service deploys its configuration API. In contrast, EMMA is expected to be a centralized module, deployed to a device with a low carbon footprint to ensure a single source of truth for energy mix information. Overall, the framework manages the CC to ensure SLO fulfillment and sustainability.

DISCUSSION

The proposed framework has the potential to transform how CC providers manage their systems and how developers leverage them. Providers can keep the privacy of developers' applications while fulfilling their SLOs and automating the configuration of their applications and their infrastructure, all with a focus on sustainability. Developers see their own application management and configuration automated, declaratively stating

their objective metrics and parameters and allowing the framework to adjust them as necessary.

The framework addresses the challenges identified to fulfill this need, be it with individual modules or with emergent properties of the system itself. Table 1 reviews how the framework addresses these challenges, which modules are involved, and related proposals or artifacts that could serve to implement the framework. The proposed framework is compared with Q-NGINE,[14] which is used to represent systems that try to address the same problem from the developer's perspective. Moreover, the framework contributes to sustainable development goals 7.3 (energy efficiency), 9.2 (sustainable industrialization), 11.2 (sustainable management

TABLE 1. Discussion of the challenges addressed by the proposed framework, comparison to how they are addressed in other works, and potential implementation for the framework modules.

ID	Challenge addressed	Involved modules	Addressing in Q-NGINE[14]	Related proposals and artifacts
C1	The proposed framework is fully aware of the carbon footprint, both in terms of energy consumption and carbon intensity of energy sources. Moreover, it is able to predict the carbon intensity of sources, as well as to learn about the implications of other variables for them (e.g., system load, location) and act accordingly.	EMMA, POPOL, AIF system	Partial: Energy consumption is considered; carbon footprint is not.	ElectricityMaps,[18] power measurement systems, AIF framework
C2	AIF fundamentally addresses this challenge by learning how high-level and low-level metrics relate to and affect each other. POPOL offers visibility and control over the details of hardware and service management.	POPOL, AIF system	Partial: Q-NGINE operates exclusively from the developer perspective. Services can be moved, but nothing else.	OpenAPI, SNMP, IPMI, Kubernetes API, AIF framework
C3	AIF performs the same task as in C2 with services as well. The services' configuration APIs offer information and let AIF modify the service configuration. SLO fulfillment can be evaluated leveraging DEVOL.	DEVOL, Configuration API, AIF system	Not considered: static service configurations are assumed.	OpenAPI, JSON, YAML, AIF framework
C4	DEVOL manages conflicts across different SLOs automatically for the rest of the framework. DEVOL can be configured to decide the method to resolve conflicts, and which SLOs to prioritize.	DEVOL	Not considered: A single party is assumed	AIF conflict resolution algorithm (cite diffusing high-level SLO in microservice pipelines)
C5	The framework as a whole is designed assuming that developers do not need to provide any specific information to the service providers. Configuration APIs and DEVOL facilitate private information sharing from developers, while POPOL and the AIF system work with it.	All (emergent property)	Addressed: As the problem is considered from the developer's perspective, no information sharing is necessary.	N/A (emergent property)
C6	DEVOL, POPOL, and the AIF system are meant to be deployed in each specific device, while each service has its own configuration API, and EMMA acts globally, exploiting the distribution and heterogeneity of the CC.	All (emergent property)	Partial: Designed for the CC, but fully centralized.	N/A (emergent property)
C7	The AIF system learns from the effects of different metrics in the carbon footprint of elements, including EMMA predictions. Then, POPOL and the configuration APIs allow AIF to take appropriate actions, such as service migration or reconfiguration.	EMMA, DEVOL, Configuration API, AIF system, POPOL	Not addressed: Q-NGINE does not have an adaptive behavior.	ElectricityMaps[16], power measuring systems, OpenAPI, SNMP, IPMI, Kubernetes API, AIF framework.

and efficient use of natural resources), and 13.2 (integration of climate change measures).

CONCLUSION

As the CC becomes the successor paradigm to cloud computing, the sustainability concerns become more relevant and complex to handle. At the same time, the issue of carbon footprint is more pressing than ever, as shown by both governmental incentives and initiatives in industry and academia. To contribute toward a sustainable CC, this work proposes a framework to intelligently manage the infrastructure by combining AIF, observability and management of the hardware, software, and service aspects, as well as SLO and energy mix information.

The framework is expected to help CC providers in achieving their objective of reducing CO_2-equivalent emissions, all while respecting the privacy and SLOs of developers. Furthermore, developers will benefit from the automated management of services through the CC while their SLOs are fulfilled, all without providing potentially private information to the CC providers.

ACKNOWLEDGMENTS

This work was supported by the European Union under the MSCA project RENOS (Contract 101205037). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. The authors acknowledge TU Wien Bibliothek for financial support through its open access funding program.

REFERENCES

1. "IDC estimates global spending on edge computing to grow at 13.8% reaching nearly $380 billion by 2028," *IDC Research*, Mar. 17, 2025. Accessed: Jun. 16, 2025. [Online]. Available: https://my.idc.com/getdoc.jsp?containerId=prUS53261225
2. S. Dustdar, V. C. Pujol, and P. K. Donta, "On distributed computing continuum systems," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 4092–4105, Apr. 2023, doi: 10.1109/TKDE.2022.3142856.
3. G. Serra, P. Fara, and D. Casini, "Enhancing the availability of web services in the IoT-to-edge-to-cloud compute continuum: A WordPress case study," in *Proc. 26th Euromicro Conf. Digit. Syst. Des. (DSD)*, 2023, pp. 602–609, doi: 10.1109/DSD60849.2023.00088.
4. K. Bilal and A. Erbad, "Edge computing for interactive media and video streaming," in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, 2017, pp. 68–73, doi: 10.1109/FMEC.2017.7946410.
5. M. Zhang, X. Shen, J. Cao, Z. Cui, and S. Jiang, "EdgeShard: Efficient LLM inference via collaborative edge computing," *IEEE Internet Things J.*, vol. 12, no. 10, pp. 13,119–13,131, May 2025, doi: 10.1109/JIOT.2024.3524255.
6. "Data centres and data transmission networks." IEA. Accessed: Jun. 16, 2025. [Online]. Available: https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks
7. F. Liu et al., "NIST cloud computing reference architecture," Sep. 2011. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-292.pdf
8. 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 *High Performance Computing*, A. Bienz, M. Weiland, M. Baboulin, and C. Kruse, Eds. Cham, Switzerland: Springer-Verlag, 2023, pp. 234–249, doi: 10.1007/978-3-031-40843-4_18.
9. U. Demirbaga, "EcoCloud: Green computing through energy and carbon efficient task scheduling in industrial IoT-enabled cloud environments," *IEEE Internet Things J.*, vol. 12, no. 17, pp. 34,644–34,652, Sep. 2025, doi: 10.1109/JIOT.2025.3537111.
10. B. Sedlak, V. C. Pujol, P. K. Donta, and S. Dustdar, "Designing reconfigurable intelligent systems with Markov blankets," in *Service-Oriented Computing*, F. Monti, S. Rinderle-Ma, A. Ruiz Cortés, Z. Zheng, and M. Mecella, Eds. Cham, Switzerland: Springer-Verlag, 2023, pp. 42–50, doi: 10.1007/978-3-031-48421-6_4.
11. B. Sedlak, V. C. Pujol, P. K. Donta, and S. Dustdar, "Equilibrium in the computing continuum through active inference," *Future Gener. Comput. Syst.*, vol. 160, pp. 92–108, Nov. 2024, doi: 10.1016/j.future.2024.05.056.
12. A. Radovanović et al., "Carbon-aware computing for datacenters," *IEEE Trans. Power Syst.*, vol. 38, no. 2, pp. 1270–1280, Mar. 2023, doi: 10.1109/TPWRS.2022.3173250.
13. P. Sharma, P. Pegus II, D. Irwin, P. Shenoy, J. Goodhue, and J. Culbert, "Design and operational analysis of a green data center," *IEEE Internet Comput.*, vol. 21, no. 4, pp. 16–24, 2017, doi: 10.1109/MIC.2017.2911421.
14. J. L. Herrera, J. Galán-Jiménez, J. Berrocal, P. Bellavista, and L. Foschini, "Energy-efficient QoS-aware application and network configuration for next-gen IoT," in *Proc. IEEE 28th Int. Workshop Comput. Aided Model. Des. Commun. Links Netw. (CAMAD)*, 2023, pp. 105–110, doi: 10.1109/CAMAD59638.2023.10478393.

15. F. Spinelli, A. Bazco-Nogueras, and V. Mancuso, "Edge gaming: A greening perspective," *Comput. Commun.*, vol. 192, pp. 89–105, Aug. 2022, doi: 10.1016/j.comcom.2022.05.022.
16. Y. Ding and T. Shi, "Sustainable LLM serving: Environmental implications, challenges, and opportunities: Invited paper," in *Proc. IEEE 15th Int. Green Sustain. Comput. Conf. (IGSC)*, 2024, pp. 37–38, doi: 10.1109/IGSC64514.2024.00016.
17. M. Wu, L. Zhou, and F. Huang, "EVCS: An edge-assisted virtual computing and storage approach for heterogeneous desktop deployment," in *Proc. IEEE 8th Int. Conf. Big Data Secur. Cloud (BigDataSecurity), IEEE Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, 2022, pp. 107–112, doi: 10.1109/BigDataSecurityHPSCIDS54978.2022.00029.
18. "Electricity Maps | The world's most comprehensive electricity data platform." Electricity Maps. Accessed: Jul. 4, 2025. [Online]. Available: https://www.electricitymaps.com

JUAN LUIS HERRERA is a postdoc researcher at TU Wien, 1040, Vienna, Austria. He is the corresponding author of this article. Contact him at j.gonzalez@dsg.tuwien.ac.at.

BORIS SEDLAK is a postdoc researcher at TU Wien, 1040, Vienna, Austria. Contact him at b.sedlak@dsg.tuwien.ac.at.

SCHAHRAM DUSTDAR is a full professor at TU Wien, 1040, Vienna, Austria. Contact him at dustdar@dsg.tuwien.ac.at.