Experiences in Architectural Design and Deployment of eHealth and Environmental Applications for Cloud-Edge Continuum

Atakan Aral*, Antonio Esposito*, Andrey Nagiyev, Siegfried Benkner, Beniamino Di Martino, and Mario A. Bochicchio

Abstract The Cloud-Edge continuum has lately exponentially grown, thanks to the increase in the availability of computational power in Edge Devices, and the better capabilities of communication networks. In this paper, two use cases, in eHealth and environmental domain, are presented in order to provide an application context to exemplify the approaches driving the analysis and selection of Cloud-Edge architectural solutions and patterns, the structural design, the allocation and deployment of distributed applications targeted to the Cloud Continuum. The main focus of this paper is the comparison of the architectural choices made for the two use cases, and how they have been driven by typical non-functional requirements, guiding the adoption of a Cloud Continuum solution.

1 Introduction

The Cloud-Edge continuum has lately exponentially grown, thanks to the increase in the availability of computational power in Edge Devices and the better capabilities of communication networks that, as of today, can support massive exchanges of data with high reliability.

Atakan Aral · Andrey Nagiyev · Siegfried Benkner
University of Vienna, Faculty of Computer Science, e-mail: ⟨name.surname⟩@univie.ac.at

Antonio Esposito · Beniamino Di Martino
Università degli Studi della Campania Luigi Vanvitelli Via Roma 29, 81031, Aversa(CE), Italy, e-mail: beniamino.dimartino@unicampania.it e-mail: antonio.esposito@unicampania.it

Mario A. Bochicchio
Università degli Studi di Bari Aldo Moro - Dipartimento di Informatica, via E. Orabona, 4, 70125. Bari, Italy, and CINI Consorzio Interuniversitario Nazionale per l'Informatica, via Ariosto 25, 00185, Roma, Italy, e-mail: mario.bochicchio@uniba.it, mario.bochicchio@consorzio-cini.it

* A. Aral and A. Esposito contributed equally to this work as the first authors.

There are still drawbacks to this approach, regarding the necessity to optimally manage the available computational resources, and to avoid traffic congestion that can always happen, even with modern networks. The main drive behind the use of the Cloud-Edge continuum paradigm resides in the necessity to keep data elaboration close to data sources, to avoid privacy and security issues, and to better organize resource management. However, this requires a clear decomposition of the applications' components that need to be re-thought in a completely distributed manner and having in mind that Edge Devices do not possess infinite computational power. The use of the Cloud-Edge continuum, which exploits centralized Cloud resources to satisfy the computational needs of the application, can resolve several resource scarcity problems but still requires to be accurately designed. Patterns to define Cloud-Edge architectures and to identify the optimal data workflows in such applications are currently being identified, studied, and applied, but there is still a strong need for standardization and general acceptance from programmers' communities.

Edge computing was originally proposed to satisfy four non-functional requirements, namely high responsiveness, scalability, privacy enforcement, and fault tolerance [1]. However, more recent, innovative applications of Cloud-Edge call for further requirements, including low network accessibility, deployment of personalized software configurations, computational offloading, and energy management.

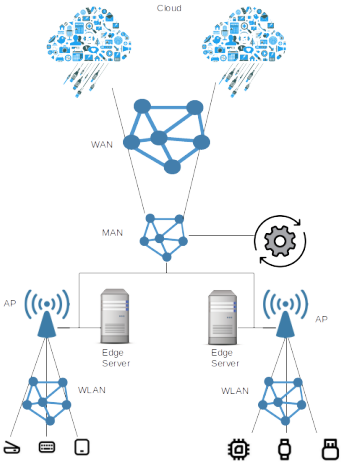
In this paper, two use cases, in the eHealth and environmental domain, are presented in order to provide an application context to exemplify the approaches driving the analysis and selection of Cloud-Edge architectural solutions and patterns, the structural design, the allocation and deployment of distributed applications targeted to the Cloud Continuum. The main focus of the paper is the comparison of the architectural choices made for the two use cases and how they have been driven by the abovementioned non-functional requirements, guiding the adoption of a Cloud Continuum solution. Moreover, we discuss how these requirements entail unique hardware designs for the two use cases.

2 Reference Architectures for the Cloud-Edge continuum

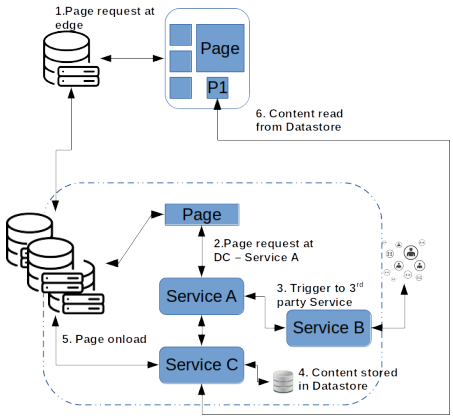
2.1 Multi-Layer Architecture

The Cloud-Edge paradigm is still in full development, especially as regards computational, energetic and privacy requirements expressed by distributed applications. Therefore, there is the need to develop techniques for the efficient definition of architectures, deployment and management methodologies for algorithms, to be run on distributed devices, and schedule the computation offload.
Several architectural solutions are being developed to support the efficient development of Cloud-Edge platforms, with a particular interest in Mobile scenarios [2]. Patterns have been proposed for Cloud-Edge, also by private, commercial organizations that apply them in their everyday activities.

Cloud-Edge Patterns can be divided into four main categories:
Architectural Cloud-Edge Patterns that support the creation of Cloud-Edge Architectures, and that are strongly influenced by multi-layered approaches already in use in several distributed paradigms. Computational capabilities of the Edge Nodes are the main drive behind the selection of a specific Architectural Pattern.
Cloud-Edge Patterns for Data are less focused on architectural design, and more interested in discussing how data should be transferred among distributed components.
Deployment Cloud-Edge Patterns define how the deployment of distributed components should be handled in a distributed environment, especially when different versions of the same application may be running on the Edge Nodes.
Among the Architectural Patterns, one of the most complex is represented by the **Multi-tier Architecture Pattern with Edge Orchestrator**[3] shown in Figure 1a. Such a Pattern describes a common situation in which the Cloud computational capabilities are separated from the Edge network, and they are both managed separately. The Edge layer is indeed organised through a Metropolitan Area Network (MAN), where and Edge Orchestrator specifically manages the computational offload among the Edge nodes.
Cloud-Edge Patterns for Data greatly differ according to the specific problem they focus on [4]. Common proposed solutions regard Synchronous and Asynchronous accesses to data produced by Edge Devices, or describe the exact workflow of the data streaming through the Edge framework. Figure 1b shows the Subsequent Data Retrieval Pattern, where the organization of the data flow is clearly presented.



(a) Two-tier Architecture Pattern with Edge Orchestrator



(b) Subsequent Data Retrieval Pattern [4]

Fig. 1: Cloud Edge Patterns examples

Patterns for Data management in complex Cloud-Edge systems are necessary, in order to describe the correct workflow of information within such systems and to optimize the energy consumption and computational loads for all the collaborating devices. To stress a privacy-by-design development of distributed algorithms, Federated Learning techniques have arisen [5, 6, 7], which guarantee the absence of personal or identifiable data in the parameters exchanged over networks to train Machine and Deep learning algorithms. Federated Learning (FL) approaches are extremely useful for the efficient exploitation of computational nodes in distributed environments, and most importantly for the positive impact they have on privacy. Since several FL-oriented approaches are possible, being guided by architectural and computational patterns becomes fundamental to reduce risks. Indeed, FL Patterns and reference architectures have already been defined [8, 9]. Applications of such patterns are available [10], but are still quite limited. Also, support for the application of such Patterns and the development of FL algorithms is still missing.

2.2 Event-Driven Architecture

Another highly connected complex of approaches and patterns for distributed applications is based on the event-driven architecture (EDA) [11]. The methods and techniques for using EDA are constantly changing and improving by adding new viewpoints and interaction mechanisms. Nevertheless, the immutable cornerstone of the architecture is the concept of the event, as an entity carrying a certain state throughout the system.

Considering the application of EDA for the Cloud-Edge paradigm, it operates with a sequence of events with a certain topic, forming a stream that transmits information from the edge devices as producers of the data, to clients consuming these events by a subscription of this topic. At the same instant, amongst the most fundamental capabilities of EDA is the availability of the data, which is achieved by the usage of mechanisms inside the brokers, acting as middleware nodes between producers and consumers. Brokers are combined into an event-driven cluster and are used as physical nodes, providing the mechanisms of replication, retention, partitioning and availability of data streams. Event-driven patterns can implement the approaches for the logical management under streams, organizing complex event processing and providing flexible procedures for the transmission of data from edge devices.

EDA has been implemented in commercial tools, e.g. [12], which are used in various organizations and activity areas, providing flexible loose coupling between different parts of systems, and supporting near-real-time interaction. Examples of the implementation of EDA to varying degrees might be such tools as Kafka or RabbitMQ. The application of the event-driven paradigm to FL and other Cloud-Edge application scenarios allows ample room for research and further improvements [13].

3 Applications to the Use Cases

3.1 E-health Application

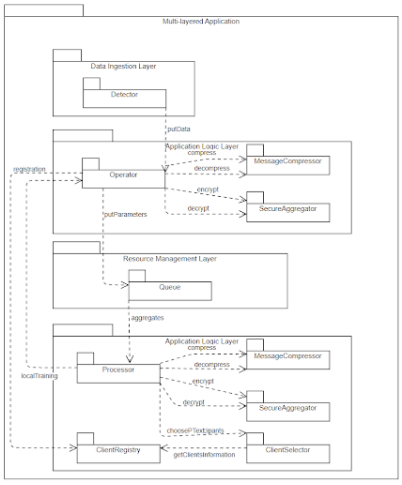
E-health applications have surely flourished thanks to the availability of advanced computational power, raw data to be analysed and, above all, distributed devices that can tackle part of the required computations locally.
The E-health use case that will be used here as an example regards the monitoring of maternal and fetal health status during pregnancy. In particular, we consider a monitoring system that includes wearable devices and series of sensors connected to it, that detect vital parameters (temperature, heart rate, blood oxygenation level, and blood pressure variances) of the mother and fetus and report any detected abnormalities to a central system. Considering that there were more than 4 million births in Europe in 2020 (about 140M worldwide) and that about one third of these are associated with health problems for the mother or the fetus, the potential user segment for such a device translates to more than one million mothers/year in Europe, and 35M mothers/years worldwide. These figures triple when considering that during the COVID-19 pandemic, remote monitoring proved essential to limit the risk of infection for pregnant women. Several companies in Europe and the U.S. are working towards the same goal, but none has yet developed clear leadership, in part because of the technical difficulties inherent in solving the problem. A first important challenge faced by this kind of Use Cases, indeed, is about the non-invasive monitoring of fetal health through sensors placed on the maternal abdomen. This requires the development and fine-tuning of sophisticated de-noising techniques and subsequent separation of signals originating from the maternal body (heartbeat, uterine contractions, muscle activity) from those originating from the fetus (mainly cardiac signal). To solve this problem, the most advanced and performing solutions make extensive use of machine learning techniques [14] that require a significant computational load and an energy expenditure that exceed the computational capacity and energy autonomy of processors currently used in wearable devices.
Figure 2a reports the envisioned target architecture for the implementation of the Use Case. Such an architecture follows the Two Tiers Pattern shown in Figure 1a, by dividing the overall framework into two main layers: the Edge tier and the Cloud Tier. In particular, the Edge Tier comprehends:

- The **Data Ingestion Layer** represents the input section of the architecture, that is responsible for the acquisition of data. In particular, this architecture foresees the use of a generic Detector component, which represents the sensors acquiring the data. For the Use Case, the Detector represents the tools used to detect the signals from the mother and the fetus during the monitoring activities.
- The **Application Logic Layer** is in charge of elaborating the incoming data to obtain the expected results. Such a layer contains three sub-components, represented by: the Operator, that is the actual computational node; the MessageCompressor, acting on exchanged messages; the SecureAggregator that focuses on

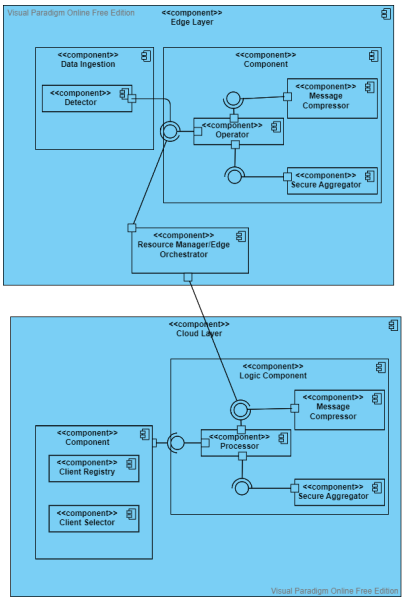
security aspects. The Operator represents here the computational capabilities of the wearable devices used to monitor the Mother and Fetus vitals.

The Cloud Tier comprehends, instead:

- The **Resource Management Layer** is in charge of the critical aspect of managing computational and data resources within the envisioned framework. Load balancing, scaling and traffic management are the main responsibilities of such a layer. This layer acts as the Edge Orchestrator, as reported in the Architectural Pattern in Figure 1a.
- The **Application Logic Layer** is in charge of elaborating the incoming data, to obtain the expected results. Such a layer contains five sub-components, represented by: the Processor, that is the computational node/cluster of nodes within the Cloud. A MessageCompressor and a SecureAggregator as in the Edge tier; a ClientRegistry and a ClientSelector that are in charge of deciding the target Cloud platform, when multiple ones are available.



(a) Target Architecture for the Use Case implementation



(b) Componentization of the Architecture

Fig. 2: Cloud Edge Patterns examples

Figure 2b shows how the Architecture can be componentised and stresses the fact that communications between the layers only happen through the Resource management component.

The communication path followed by the application is better described through the Sequence Diagram in Figure 3. All the information follow a linear flow through

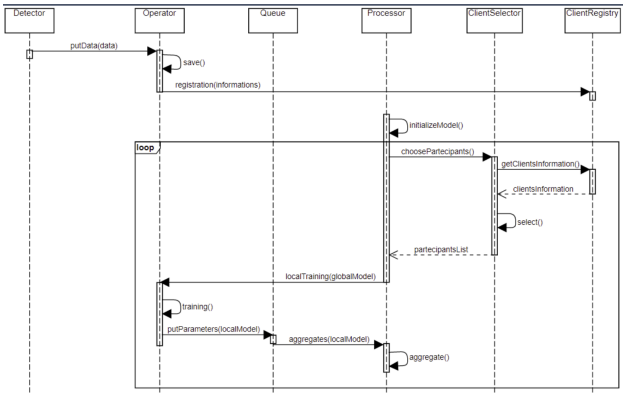


Fig. 3: Communications among components in the E-health Use Case

the components of the application, with Detector and Operator communicating the data directly. The Processor retrieves the data from a common data structure (a generic Queue in the Architecture of Figure 2a), and it operates in a loop, retrieving the data and elaborating them locally. Different Processors can participate into the elaboration of the data.

Theoretically speaking, the minimum sampling frequency for an optimal ECG recorder is equal to 50 Hz, but in order to obtain precise measurements a typical ECG recorder samples data with a frequency of more than 500 Hz [15]. In some situations even higher frequencies are required: this means that the Detector needs to feed a continuous stream of data to the local Operator, which in turn has to rapidly analyse the data and provide immediate feedback, in order to detect anomalies (*putData* method in the Sequence Diagram in Figure 3).

On the opposite side, the Operator will not send the data (*registration* method) and the updated local model (*putParameters* method) frequently. Indeed, the updates can be scheduled on a regular basis, and their rate can vary a lot, according to the overall settings of the system. It is evident that dividing the system between the Edge Tier, with Detector and Operator working at strict contact, and a remote Cloud Tier, where the Processor can analyse the data without haste, and the update of the global model must be coordinated among several Operators, seems a feasible and suitable solution.

After extracting the fetal heart rate (FHR) from the maternal abdominal signal, this information can also be used to trigger alarms when its level is outside the standard range for the specific gestational age and context (e.g., walking, sleeping, etc.). Similar alarms can be associated with the other monitored vital signs (e.g., SPO2, body temperature, blood pressure, blood glucose level) based on the specific risks associated with the patient. This scheme of customized triggers and alarms fits very well with the EDA approach mentioned above.

3.2 Environmental Monitoring Application

Environmental monitoring and real-time decision-making are essential to environmental protection. Various applications towards pollution monitoring (air, water, soil, etc.) and disaster early warning (seismic activity, avalanches, etc.) already benefit from Cloud-Edge deployment [16]. In water quality monitoring, there exist monitoring systems for marine regions and freshwater bodies (both ground and surface water). SWAIN project (https://swain-project.eu/) focuses on surface waters, particularly rivers. The project aims to detect and locate pollutant sources (e.g., industrial leaks or failed wastewater treatment plants) through an unprecedented implementation of Edge Computing and IoT for the real-time analysis of water contamination data. Timely decision-making is crucial in this use case as the river water polluted upstream might be used downstream for irrigation or for municipal water intake.

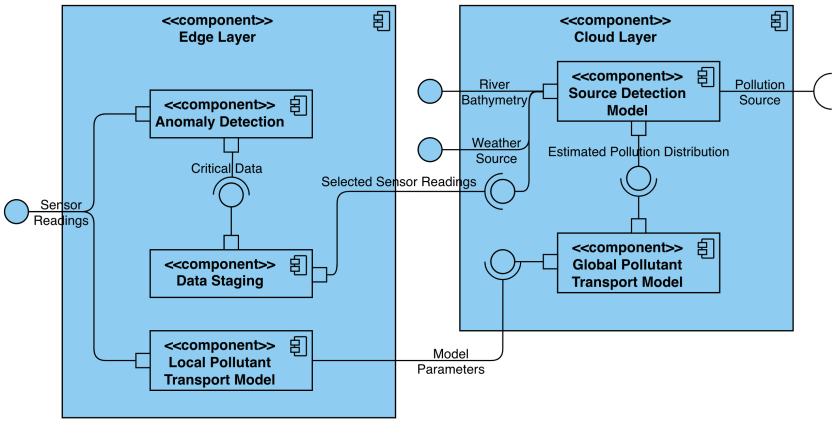


Fig. 4: Component Diagram for the Environmental Monitoring Use Case.

Data analytics components of this application are deployed on the Cloud-Edge continuum in order to benefit from its responsiveness and network access advantages. Figure 4 visualizes these components and their interfaces. Since watersheds are located in remote areas, it is not always possible (due to the lack of reliable networks) to transmit sensor readings to the cloud, where complex river models can be executed. The edge subsystem, therefore, acts as intermediate data storage and pre-processing locations, which mitigates delayed decisions or data loss. As demonstrated in Figure 4, the transport model of pollutants is decomposed into local and global components. Local components are less complex since they estimate the pollutant distribution only in their local geographical area. They transmit trained model parameters to the Cloud subsystem, where a global model incorporates the local updates. Based on the critical data readings selected by the Edge subsystem and the pollutant distribution estimated by the pollutant transport model, the source detection model is able to identify the source of the pollution.

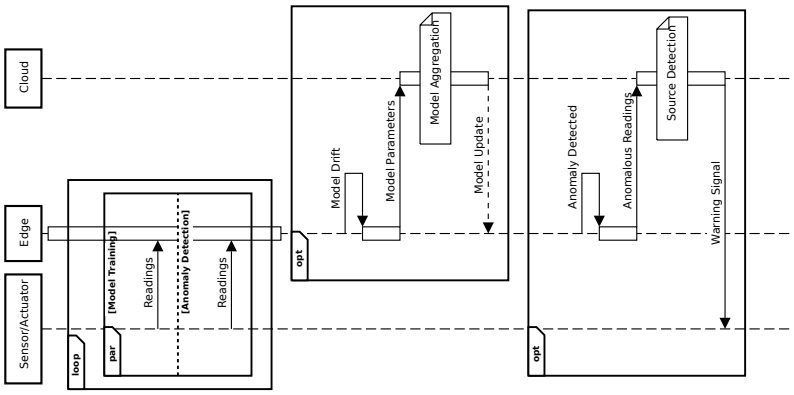


Fig. 5: Sequence Diagram for the Environmental Monitoring Use Case.

Figure 5 illustrates the event communication between IoT, edge, and cloud components as a sequence diagram. The loop fragment demonstrates the synchronous data transmission between IoT sensors and the Edge component. The rest of the messaging is event-driven, as illustrated by the optional fragments. Model parameters are sent to the cloud only when a model drift is observed during local training, whereas pollution source detection at the cloud is only triggered when anomalous data is detected at the edge.

4 Conclusions

As exemplified by the two application use cases described in the paper, Multi-Layer and Event-Driven architectural solutions can be used effectively to capture various nonfunctional requirements that characterize numerous industrially and socially relevant application domains For instance, the low network access requirement of river monitoring entails the two-tier architecture and local training at the edge layer. Local training results in substantial data reduction compared to transmitting raw sensor data to the cloud. Similarly, EDA is motivated by the energy constraints of edge resources since processing is only triggered by rare events resulting in low power consumption.

In general, EDA is a powerful approach for building IoT-based environmental monitoring systems due to their particular non-functional requirements, such as (i) loose coupling between the tiers (intermittent network connectivity), (ii) real-time processing (delays might result in irreversible environmental impact), (iii) low energy consumption (no access to the electricity grid in remote areas) and (iv) reduced bandwidth utilization (the transfer is based only on changes in data).

Acknowledgement

This work was partially funded by the Digital Europe Programme, Project DANTE EDIH, ID: 101083913, as within the activities conducted by the "CINI - Consorzio Interuniversitario Nazionale per l'Informatica".
A. Aral was supported by the CHIST-ERA grant CHIST-ERA-19-CES-005 and by the Austrian Science Fund (FWF): I 5201-N.

References

1. M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
2. L. Baresi, D. F. Mendonça, M. Garriga, S. Guinea, and G. Quattrocchi, "A unified model for the mobile-edge-cloud continuum," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–21, 2019.
3. T. Zheng, J. Wan, J. Zhang, and C. Jiang, "Deep reinforcement learning-based workload scheduling for edge computing," *Journal of Cloud Computing*, vol. 11, no. 1, pp. 1–13, 2022.
4. A. Koloth, "Data patterns for the edge: Data localization, privacy laws, and performance," *Published online at:https://www.infoq.com/articles/data-patterns-edge/*, 2022. "Last access 25/11/2022.
5. J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, no. 1, pp. 1–19, 2021.
6. N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, *et al.*, "The future of digital health with federated learning," *NPJ digital medicine*, vol. 3, no. 1, pp. 1–7, 2020.
7. C.-R. Shyu, K. T. Putra, H.-C. Chen, Y.-Y. Tsai, K. T. Hossain, W. Jiang, and Z.-Y. Shae, "A systematic review of federated learning in the healthcare area: From the perspective of data properties and applications," *Applied Sciences*, vol. 11, no. 23, p. 11191, 2021.
8. S. K. Lo, Q. Lu, L. Zhu, H.-y. Paik, X. Xu, and C. Wang, "Architectural patterns for the design of federated learning systems," *Journal of Systems and Software*, vol. 191, p. 111357, 2022.
9. S. K. Lo, Q. Lu, H.-Y. Paik, and L. Zhu, "Flra: A reference architecture for federated learning systems," in *European Conference on Software Architecture*, pp. 83–98, Springer, 2021.
10. B. Di Martino, M. Graziano, L. Colucci Cante, and D. Cascone, "Analysis of techniques for mapping convolutional neural networks onto cloud edge architectures using splitfed learning method," in *International Conference on Advanced Information Networking and Applications*, pp. 163–172, Springer, 2022.
11. A. Bellemare, *Building Event-Driven Microservices*. O'Reilly Media, Inc., 2020.
12. G. Shapira, T. Palino, R. Sivaram, and K. Petty, *Kafka: the definitive guide: real-time data and stream processing at scale*. O'Reilly Media, Inc., 2020.
13. C. Martín, P. Langendoerfer, P. S. Zarrin, M. Díaz, and B. Rubio, "Kafka-ml: Connecting the data stream with ml/ai frameworks," *Future Generation Computer Systems*, vol. 126, pp. 15–33, 2022.
14. M. R. Mohebbian, S. S. Vedaei, K. A. Wahid, A. Dinh, H. R. Marateb, and K. Tavakolian, "Fetal ecg extraction from maternal ecg using attention-based cyclegan," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 2, pp. 515–526, 2021.
15. E. Ajdaraga and M. Gusev, "Analysis of sampling frequency and resolution in ecg signals," in *2017 25th Telecommunication Forum (TELFOR)*, pp. 1–4, 2017.
16. V. De Maio, A. Aral, and I. Brandic, "A roadmap to post-moore era for distributed systems," in *Proceedings of the 2022 Workshop on Advanced tools, programming languages, and PLatforms for Implementing and Evaluating algorithms for Distributed systems*, pp. 30–34, 2022.