ICOS: An Intelligent MetaOS for the Continuum

Jordi Garcia
Xavi Masip-Bruin
CRAAX Lab
UPC BarcelonaTECH
Vilanova, Spain

Anastasios Giannopoulos
Panagiotis Trakadas
DPMS, National and Kapodistrian
University of Athens (NKUA)
Psachna, Greece

Sebastián A. Cajas Ordoñez
Jaydeep Samanta
Andrés L. Suárez-Cetrulo
Ricardo Simón Carbajo
CeADAR
Dublin, Ireland

Marc Michalke
Admela Jukan
ICNE, Technische Universität
Braunschweig (TUBS)
Braunschweig, Germany

Artur Jaworski
Marcin Kotliński
Poznan Supercomputing and
Networking Center
Poznan, Poland

Gabriele Giammatteo
Engineering Ingegenria Informatica
S.p.A.
Rome, Italy

Francesco D'Andria
ATOS Research and Innovation
Barcelona, Spain

Abstract

This paper presents ICOS, a meta-operating system designed for the cloud continuum. The paper provides insight into the ICOS architecture, focusing on the Intelligence Layer and highlighting the benefits and functionalities it provides to administrators and users of the edge-to-cloud continuum. It also describes in detail some experimental results to predict the CPU utilization of the nodes that build up the ICOS system. The purpose of this paper is to show the benefits of using ICOS with AI-subsystem and illustrate them through real experiments.

ACM Reference Format:
Jordi Garcia, Xavi Masip-Bruin, Anastasios Giannopoulos, Panagiotis Trakadas, Sebastián A. Cajas Ordoñez, Jaydeep Samanta, Andrés L. Suárez-Cetrulo, Ricardo Simón Carbajo, Marc Michalke, Admela Jukan, Artur Jaworski, Marcin Kotliński, Gabriele Giammatteo, and Francesco D'Andria. 2025. ICOS: An Intelligent MetaOS for the Continuum. In *2nd International Workshop on MetaOS for the Cloud-Edge-IoT Continuum (MECC '25), March 30–April 3, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3721889.3721929

1 Introduction

In light of the advent of 6G and related technologies like the proliferation of AI and ML applications that are being used in all kinds of areas, computing has to become omnipresent to drive a diverse range of use cases. In modern environments, this can be realized by utilizing cloud computing together with edge computing to provide flexible processing and IoT to gather data or instruct actuators [10]. Due to the sheer number of devices, managing such a variety of resources individually becomes virtually impossible, so infrastructure providers try to merge them together into the so-called IoT-Edge Cloud continuum.

The continuum allows providers to treat the whole environment as one, abstracting away the difficulties of individual resource scheduling and allocation while simultaneously allowing for great flexibility in workload distribution [9]. While this moves complexity away from the user, the management has to be taken care of by a system that is able to make effective use of the computational capabilities of all devices, assign workloads dynamically, and interact with many technologies at once [8]. Especially the latter can become quite complicated as the continuum might leverage different platforms like Kubernetes, OpenShift, Docker, or other solutions for different domains and devices, depending on their individual capabilities, roles, and integrated environments. This coordination and orchestration effort demands a Meta Operating System for the continuum; a role that has captured the attention of researchers [13, 14] and EU projects [1, 3, 5–7], and that ICOS [4], the intelligent MetaOS for the continuum, fills.

ICOS allows the user to interact with a single system to manage their applications, regardless of where the individual workloads are being run. The user merely states what properties the components of each application need, e.g., a certain amount of compute resources, a minimum security level, or a connection to specific IoT devices like cameras, while ICOS takes care of the placement of each component as well as their inter-connection even across cluster boundaries if necessary. Placing these components not only requires knowledge of the node's connected devices, their compute resource utilization, and other potentially dynamic properties but also demands the ability to foresee the changes in these dimensions after the components have been deployed. This requires ICOS to employ sophisticated matchmaking and prediction capabilities to anticipate the foreseeable future whenever components have to be placed on compute nodes. In addition, advanced features for sophisticated decision-making like user-defined policies, forecast

MECC'25, Rotterdam, Netherlands

ACM ISBN 979-8-4007-1560-0/25/03
https://doi.org/10.1145/3721889.3721929

metrics, and ML model creation are needed for the MetaOS to be able to fulfill the demanding task of application placement across such a vast number and range of hardware.

In this paper, we lay out the architecture of ICOS, explain its components and how they interact with each other. We then provide deeper insight into the intelligence layer, the AI-subsystem, highlighting the benefits and functionalities it provides to administrators and users of the ICOS system. We then describe the architecture of an experimental setup, focusing on the end-to-end learning framework which we then use for CPU utilization forecasting with data corresponding to three ICOS nodes and present the results, showing the accuracy of this prediction.

Our main contributions can be summarized as follows:

- present and describe the ICOS MetaOS architecture,
- lay out the details of the ICOS AI subsystem,
- conduct an experiment to show the benefits of this research.

The remainder of the paper is organized as follows: Section 2 presents the ICOS architecture. Section 3 describes the Intelligence Layer of the system. Section 4 then provides the details and results of our experimentation, and section 5 concludes the paper.

2 ICOS Architecture

The ICOS MetaOS for the continuum has been designed with two main roles: Controllers and Agents. ICOS Controllers have two main responsibilities: managing the resources along the continuum and providing an efficient and effective execution environment. ICOS Agents represent an ICOS delegate that runs along the continuum and becomes a single point of management on the computing devices (hereafter referred to as nodes) attached to them.

An ICOS instance requires at least one Controller to be responsible for managing the continuum and making the runtime decisions. However, ICOS has been designed as a multi-controller system. This means that more than one Controller could cooperate in the management of ICOS. The reason for having more than one Controller could be twofold: either because one Controller has too many Agents attached to it and it is becoming a bottleneck (scalability), or because some Agents are too far from the Controller and latency is becoming an issue (locality). Agents are the ICOS point of management on the computing nodes (either in the cloud or at the edge). One or more nodes are attached to an Agent, and one or more Agents are connected to a Controller. Fig. 1 shows an illustration of a typical ICOS instance. In this figure, a set of more or less powerful computing devices (nodes) can be seen at the edge, which are attached to different Agents (one Agent can be managing several nodes). In addition, one cluster in the cloud is also attached to an Agent. Finally, all agents are connected to a controller, who will be responsible for managing the continuum.

The architecture of an ICOS Controller has been conceived as a three-layer design: the Meta-kernel Layer, which provides all functionalities related to resources and runtime management; the Intelligence Layer, which implements all functionalities related to intelligence training and model management; and the Security Layer, that implements all the authentication, authorization, and trust mechanisms.

The Meta-kernel Layer is organized into three main blocks: the Continuum Management, the Runtime Management, and the Telemetry Controller, as can be seen in Fig. 2. The Continuum Management block is responsible for keeping track of the devices in the continuum along with information about their status (e.g., availability) and includes the Resource Onboarding component, which is responsible for managing the dynamic node onboarding process, and the Aggregator component, responsible for retrieving all static and dynamic resource-related information from the system database. The Runtime Management block is responsible for managing all requests related to the execution of an application along the continuum. This block includes the Job Manager, responsible for managing the execution lifecycle of jobs; the Matchmaker, responsible for finding the appropriate nodes to execute each application component according to some user-defined SLA, and the Policy Manager, responsible for monitoring the execution of applications and detecting and remediating any potential violations in the expected execution performance. And finally, the Telemetry Controller builds and manages a sophisticated telemetry system that collects detailed telemetry information from the underlying nodes and stores it in a time-series database. This information will later be retrieved by the Aggregator (presented above) to create a comprehensive snapshot of the system topology and its instantaneous state. The collected data contains information about the resources' capacities, their status, their consumption, and their security score (SCA, generated as part of the Security Layer), among others. This detailed information allows an efficient and effective selection of the nodes that should execute a job in the continuum.

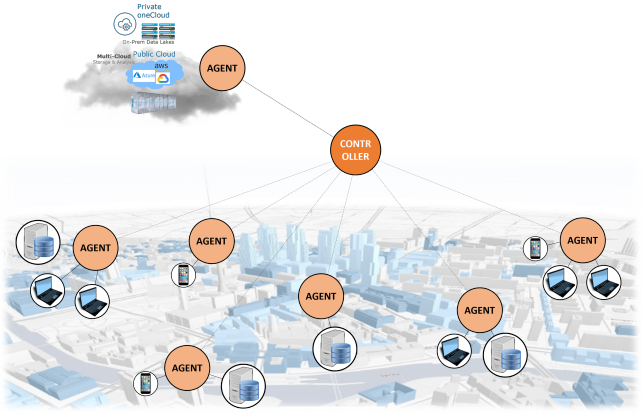

Figure 1: Typical ICOS instance.

On the other hand, the architecture of an ICOS Agent consists of four main components, as shown in Fig. 3. The Onboarding Manager is the component responsible for managing the onboarding of the Agent itself, as well as of the resources (nodes) attached to it, and coordinates the onboarding process with the Controller. The App Setup Manager is responsible for configuring the nodes prior to the execution of an application (in particular, when cross-cluster operation is required), and the Deployment Manager is the component responsible for the actual deployment of the container on the target node (according to the Controller's placement instructions). Finally, the Telemetry Gateway is responsible for collecting all telemetry data from the underlying nodes and transferring them to the Telemetry Controller.

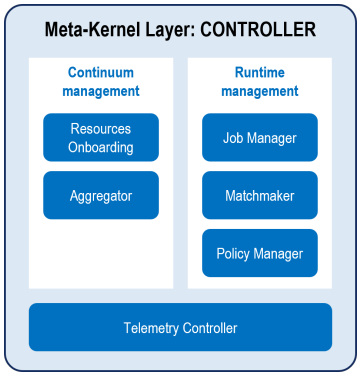

Figure 2: Architecture of an ICOS Controller.

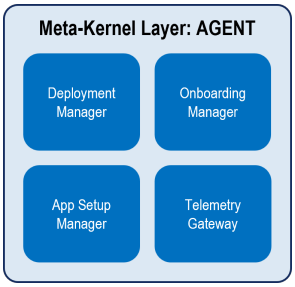

Figure 3: Architecture of an ICOS Agent.

The basic process for running an application is carried out as follows. At the Controller level, the Job Manager receives an execution request. This requests an optimal mapping from the Matchmaker, which considers not only the characteristics and topology of the available nodes but also their current and predicted availability (provided through the Intelligence Layer) and other user-defined preferences. All this resource-related information is extracted by the Aggregator from the database fed by telemetry. Once the Job Manager knows the target nodes that should run the application, it communicates with the appropriate Agents to offload the execution of the corresponding container. Now, the App Setup Manager configures any initialization settings, and the Deployment Manager launches the container for execution on the target nodes.

Meanwhile, the Telemetry Gateway collects complete telemetry information from nodes that will feed the Telemetry Controller time-series database. This process, together with the rich data collected in ICOS, shapes a comprehensive mechanism that becomes the root of a sophisticated intelligence-based subsystem and an opportunity to leverage ML-based technology. In the next section, the Intelligence Layer is introduced. The Security Layer is out of the scope of this paper; however, a complete description can be found in [11].

3 The Intelligence Layer

The Intelligence Layer is a core component in the ICOS ecosystem that enables optimizing AI operations across the edge-to-cloud continuum. It orchestrates model coordination, optimization, and monitoring, ensuring robust support for predictive analytics and real-time decision-making. The Intelligence Layer is made of the following four modules within the ICOS Controller: i) Intelligence Layer Coordination module, ii) Data Processing module, iii) AI Analytics module, and iv) Trustworthy AI module. On top of this, it integrates with an online repository of AI models and data generated using ICOS [1]. The next subsections explain the main modules and functionalities within the Intelligence Layer that contributed to the experimental results in Section 4.

3.1 Intelligence Layer Coordination Module

The Intelligence Layer Coordination Module coordinates AI model training, inference, and deployment for models that forecast future usage metrics while ensuring seamless sharing, updating, and integration within the ICOS framework. It provides APIs to facilitate efficient model coordination and maintains a model registry with essential metadata, including descriptions, algorithms, and performance metrics, while it operates across the cloud continuum. Key performance indicators, such as training and validation curves, are tracked using MLOps monitoring tools to ensure continuous optimization. This module is divided into frontend and backend APIs, which are described in the next subsections.

3.1.1 AI coordination frontend (Export metrics API). The Export Metrics API, as part of the AI coordination frontend, serves as the gateway for the ICOS MetaOs to the Intelligence API. It connects with other *ICOS* components and gives ICOS the capability of metric forecasting. It automates the metric collection, preprocessing, and model updates, using Grafana and Prometheus to retrieve and structure telemetry data. Enabling time-series preprocessing for both univariate and multivariate models ensures efficient real-time monitoring, predictive analytics, and AI-driven decision-making.

It allows ICOS developers to request new models to the Intelligence layer through the */create_model_metric* endpoint. Additionally, it queries the ICOS Aggregator through Telemetry in the background to proactively create new AI models and forecasts for key utilization metrics in new ICOS nodes.

3.1.2 AI coordination backend (Intelligence API). The Intelligence Layer Coordination API serves as the backend of the Intelligence Layer, interacting with its model registry and managing the layer's modules to coordinate AI model training, prediction, and monitoring. It provides AI experiment tracking functionality, including model performance indicators, training loss curves, and leveraging tools like MLFlow [2] to enable AI experiments across the cloud continuum. It supports model versioning through a model registry, asynchronous model serving, and micro-batching using BentoML [3].

3.2 Data Processing Module

This module focuses on managing the reading and writing operations as well as data transformation and preparation. It tackles raw data retrieved from the data management component (dataClay[4]) and converts it using libraries such as NumPy[5] and pandas[6] into structures optimized for training machine learning models.

[1] ICOS AI Models and Data Repository: https://huggingface.co/ICOS-AI
[2] MLFlow platform: https://mlflow.org
[3] BentoML model-serving framework: https://github.com/bentoml/BentoML
[4] dataClay distributed data storage: https://github.com/bsc-dom/dataclay
[5] NumPy: https://numpy.org
[6] pandas: https://pandas.pydata.org

This module can distribute the processing workload to improve efficiency. When a request comes into the Intelligence Layer, computationally heavy operations, particularly model training, are transferred using dataClay to other ICOS nodes for processing.

3.3 AI Analytics Module

The AI Analytics module is a machine-learning toolkit designed to optimize workload distribution by aggregating energy-efficient task offloading strategies, security, load balancing time series forecasting (including anomaly detection), and CPU or RAM utilization forecasting. It supports improved resource management through multivariate metrics forecasting, model compression techniques for efficient AI training and inference, and integration with data pipelines through the Data Processing module for optimized data preprocessing.

(1) **Anomaly detection**: identifying irregularities in system behavior.
(2) **Metrics forecasting**: predicting load balancing, CPU utilization, RAM, or energy to be consumed by an ICOS node [12].

3.4 Trustworthy AI Module

This module aids the Intelligence Layer in meeting with the Trustworthy AI pillars [16]. It incorporates:

(1) **Model explainability**: SHAP (Shapley Additive Explanations) [7] is integrated within the Intelligence Layer to interpret predictions and provide explainability and transparency, helping to reduce model bias.
(2) **Confidence estimation**: confidence scores and intervals to quantify prediction reliability for every prediction.
(3) **Drift detection and model performance estimation**: to monitor data distribution changes and concept drifts [12, 15] , helping to increase the overall robustness of the models monitored. This is achieved by integrating NannyML [8] into the Intelligence Layer.
(4) **Federated learning**: for privacy-aware mode training. This helps further reduce data movement from edge devices.

This paper makes use of the three Intelligence Layer modules covered, placing special emphasis on the federated learning component introduced in this section. Federated learning in ICOS leverages the Flower framework [9] that enables collaborative model training across multiple decentralized devices or organizations while preserving data privacy. When local models, trained throughout the ICOS infrastructure, are combined into a single model, this aggregated (global) model becomes part of the AI coordination backend model registry and is available for inference (like any other model trained via the Intelligence Layer.

4 Experimental Results

In this section, we present the architecture of the experimental setup, focusing on the end-to-end learning framework of both standalone and federated learning pipelines [2]. Then, based on the proposed architectural sequence, we provide numerical results for CPU utilization forecasting with data corresponding to three ICOS nodes, and we compare the prediction accuracy of multiple single-model algorithms (GRU, Transformer, RNN, LSTM) and federated learning.

[7]SHAP library: https://shap.readthedocs.io
[8]NannyML framework: https://www.nannyml.com
[9]Flower library: https://flower.ai

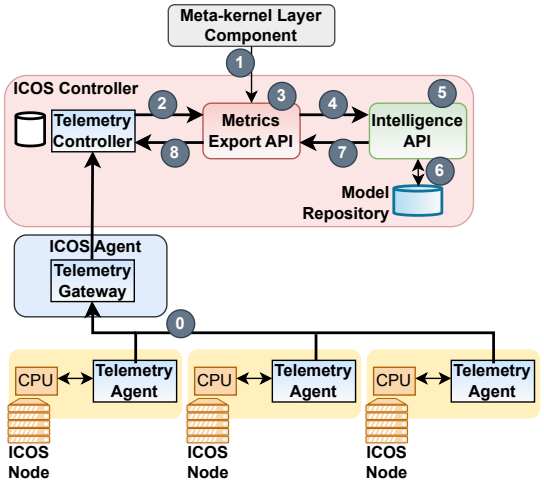

Figure 4: Experimental Setup for ICOS-based CPU Monitoring and Pattern Learning.

4.1 Architectural Learning Framework

The architectural setup of ICOS MetaOS has been designed to facilitate a complete learning sequence, encompassing both standalone (i.e., single-model) and Federated Learning (FL) paradigms. Figures 4 and 5 illustrate the flow of operations within the ICOS framework for these learning processes.

4.1.1 Standalone Learning Sequence. Figure 4 depicts the experimental setup for ICOS-based CPU monitoring and pattern learning. This sequence can be generalized to any other metric, such as memory usage, disk capacity, or power consumption of ICOS nodes. The sequence begins at the ICOS nodes (Step 0), where the Telemetry Agents continuously collect metrics from the CPU of ICOS nodes. These metrics are transmitted through the telemetry gateway to the ICOS Controller. The Telemetry Controller (Step 1) processes these metrics (Step 3) and exports them via the Metrics Export API (Step 4). This API provides the metrics to the Intelligence API (Step 5), which communicates with the Model Repository (Step 6) to access or store models for training or inference. The resulting trained model is utilized for real-time inference, completing the feedback loop to optimize CPU utilization dynamically. For instance, model inference may produce hourly predictions regarding the upcoming CPU usage at the ICOS nodes, representing their overall load status. Note that final predictions are posted back in the Telemetry Controller, which is interfaced with Meta-kernel components (e.g., Policy Manager or Matchmaking component) for further AI-aided operations (e.g., intelligent matchmaking to assign application components to idle or non-overloaded ICOS nodes).

The modular design of the ICOS MetaOS allows seamless integration of components and ensures the scalability of the telemetry-to-intelligence pipeline. This framework ensures robust data collection,

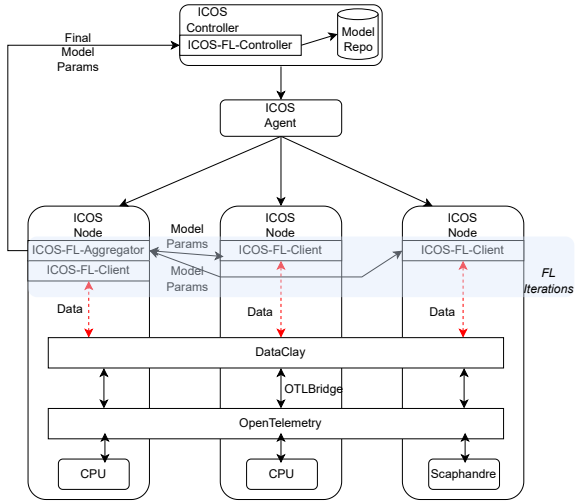

Figure 5: Architecture of the ICOS-based Federated Learning sequence.

processing, and model deployment for efficient resource utilization in a distributed environment.

4.1.2 Federated Learning Sequence. Figure 5 demonstrates the architecture for FL within the ICOS MetaOS. In this sequence, the ICOS controller coordinates FL rounds by initializing the ICOS-FL-Controller module. This controller interacts with the Model Repository to distribute the initial global model parameters to ICOS nodes.

Each ICOS node hosts an ICOS-FL-Client module, which participates in FL rounds by training local models on their respective datasets. The training process leverages OpenTelemetry and DataClay layers for data collection and management. For nodes that support energy monitoring, such as those equipped with Scaphandre, additional metrics are integrated to optimize energy consumption during training.

Once the local training is complete, the ICOS-FL-Clients send the updated model parameters back to the ICOS-FL-Aggregator, which combines them to produce an updated global model. This process iterates until the desired model performance is achieved. The final model is then stored in the Model Repository and is available for future inference by Meta-kernel components.

By leveraging FL, the ICOS framework enhances privacy and reduces bandwidth usage, making it suitable for applications where data cannot be centralized and ICOS node owners require data privacy preservation (i.e., FL clients share only their local model parameters).

4.2 Numerical Results

In this subsection, we evaluate the performance of ICOS MetaOS in predicting CPU utilization under two scenarios: standalone and federated learning.

4.2.1 Standalone Learning Performance. We consider CPU utilization time-series data from three ICOS nodes. The sampling rate of each CPU dataset is one hour, whereas the size of the datasets is 5000 data points. CPU utilization is measured as a percentage (0–100%). Each ICOS node has been monitored during the execution of different computational tasks, including:

- The task underlying the dataset of ICOS Node 1 was an automated database backup system characterized by highly regular periodicity.
- The task corresponding to the second node's dataset was a video rendering farm, presenting wave-like CPU behavior with gradual transitions (20%–80%) with predictable usage patterns.
- ICOS Node 3 was monitored when serving as a production web application server, showing irregular CPU fluctuations (62%–76%).

Table 1: Evaluation Metrics of ML Models across ICOS Nodes

Node	Model	MAE	MSE	RMSE	R^2
Node 1	**LSTM**	0.04319	0.00442	0.06646	0.99559
	GRU	0.05577	0.00513	0.07161	0.99488
	RNN	0.04959	0.00409	0.06398	0.99591
	Trans.	**0.04297**	**0.00328**	**0.05728**	**0.99672**
Node 2	**LSTM**	**0.04190**	**0.00275**	**0.05243**	**0.99725**
	GRU	0.04353	0.00294	0.05423	0.99706
	RNN	0.06439	0.00626	0.07911	0.99374
	Trans.	0.05499	0.00453	0.06727	0.99547
Node 3	**LSTM**	0.35873	0.15138	0.38907	-0.95337
	GRU	**0.10137**	**0.01664**	**0.12899**	**0.78529**
	RNN	0.42552	0.19355	0.43994	-1.49755
	Trans.	0.60247	0.41862	0.64701	-4.40195

For each node, four machine learning models, namely RNN, GRU, Transformer, and LSTM, were independently trained and tested to forecast CPU usage. For all models, historical CPU utilization data was first scaled and then segmented using a recursion window of 5 samples (i.e., a lookback window of 5 hours), with a prediction step of one value ahead (i.e., forecasting the CPU usage in the next hour). The models were trained using the Mean Squared Error (MSE) as a loss function and optimized via grid-search hyperparameter tuning (including learning rate, number of hidden layers, and activation functions). The training progress was monitored through loss-versus-epoch curves.

As shown in Table 1, key performance metrics, including Mean Absolute Error (MAE), MSE, Root MSE (RMSE), and R^2, were calculated to compare the predictive capabilities of the models across the three nodes. Note that the bold-noted rows in Table 1 correspond to the best-performing models per ICOS Node. Figure 6 shows the fitness of the predicted testing values on the actual curve for each node-specific optimal model.

Evidently, the ML performance differences across ICOS nodes stem from the distinct temporal patterns in CPU utilization. In Node 1, where CPU load is highly periodic, LSTMs and Transformers excel by capturing long-term dependencies and recurring trends. In Node 2, with smooth wave-like transitions, LSTMs perform best as they efficiently model gradual variations. In Node 3, where CPU usage is irregular, GRUs outperform others due to their ability to adapt to short-term dependencies and volatile patterns.

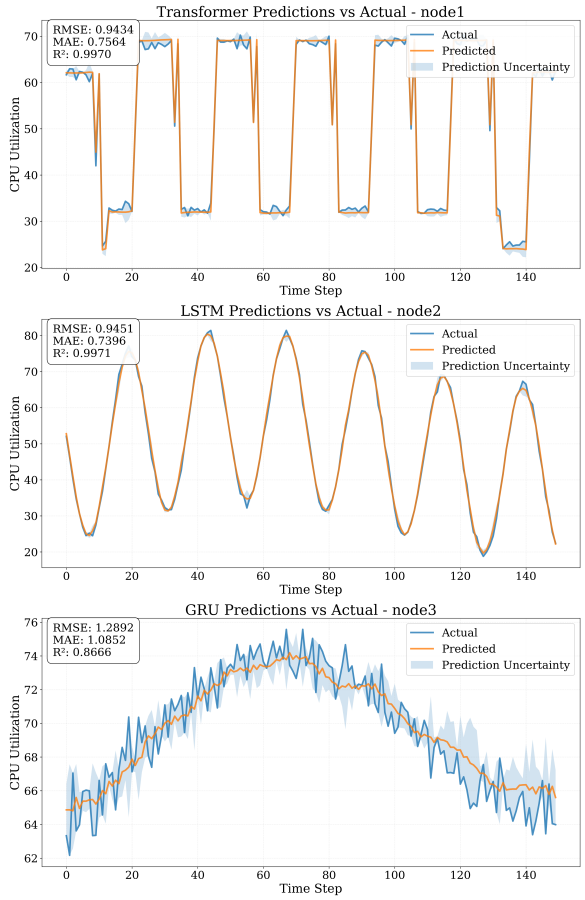

Figure 6: Actual versus predicted CPU usage curves considering the optimal model of each ICOS node.

4.2.2 Federated Learning Accuracy. For the FL scenario, the accuracy of the aggregated global model was compared against the standalone models. The federated approach achieved comparable or superior accuracy while preserving data privacy. Figure 7 depicts the actual and the predicted values as they derived from the FL model. Specifically, the FL approach shows strong overall performance for Nodes 1 and 2 with R^2 values of approximately 0.9936. For Node 3, the evaluation score yields an R^2 of 0.8393. This discrepancy suggests that, although the FL model generalizes well for the broader patterns, it struggles to capture finer-scale variations in Node 3.

Overall, the experimental results validate the effectiveness of ICOS MetaOS in supporting advanced learning frameworks, enabling intelligent resource management in distributed environments.

5 Conclusions and Future Work

In this paper, the purpose and main goal of the ICOS MetaOS has been presented. The system architecture has been described, distinguishing between two main roles: ICOS Controller and ICOS Agent. The main focus of this paper has been put on the Intelligence Layer,

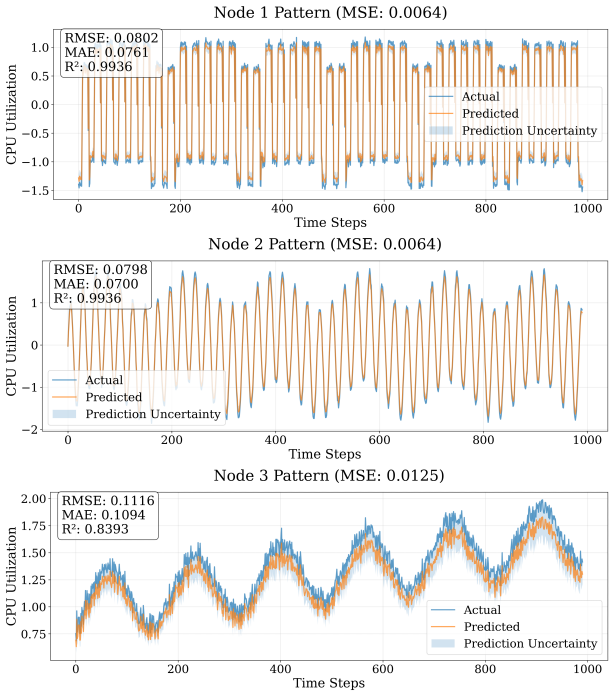

Figure 7: Actual versus predicted CPU usage curves considering the Federated Learning model applied to each ICOS node.

which provides vital tools for optimizing operations performed throughout the edge-to-cloud continuum.

The conducted experiment, which has been thoroughly presented, confirmed that the learning framework, implemented as a part of the Intelligence Layer, stood up to its task when it came to predicting onboarded nodes' CPU utilization, both in the standalone and federated learning paradigms. The federated learning approach not only achieved comparable or superior results compared to standalone learning but also allowed data privacy to be preserved. At this stage of the project, the experiment's results are promising, as the overall system's performance can still be enhanced in the following months.

The contribution of this work is a functional edge-to-cloud metaOS, utilizing AI tools to improve the system's performance and efficiency. The future research directions consist of, but are not limited to: a) apply the prediction mechanisms to a larger set of metrics including system performance, security and application performance metrics; b) deeply integrate the AI predictions with orchestration and policy engines to timely react to predicted disturbances of the system efficiency and applications performance and re-establish the optimal conditions; c) correlate metrics and predictions to the delivered applications' quality of experience to enable optimizations tailored on the specific applications.

Acknowledgments

This work has been supported by the EU Horizon programme under grant 101070177 (ICOS project), and grant PID2021-124463OB-I00 funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU.

References

[1] aerOS authors. 2025. aerOS. https://aeros-project.eu/

[2] Angelos Angelopoulos, Anastasios Giannopoulos, Nikolaos Nomikos, Alexandros Kalafatelis, Antonios Hatziefremidis, and Panagiotis Trakadas. 2024. Federated Learning-Aided Prognostics in the Shipping 4.0: Principles, Workflow, and Use Cases. *IEEE Access* (2024).

[3] FLUIDOS authors. 2025. FLUIDOS. https://fluidos.eu/

[4] ICOS authors. 2025. ICOS. https://icos-project.eu/

[5] NebulOuS authors. 2025. NebulOuS. https://nebulouscloud.eu/

[6] NEMO authors. 2025. NEMO. https://meta-os.eu/

[7] Nephele authors. 2025. Nephele. https://nephele-project.eu/

[8] Anastasios Giannopoulos, Ilias Paralikas, Sotirios Spantideas, and Panagiotis Trakadas. 2024. COOLER: Cooperative Computation Offloading in Edge-Cloud Continuum Under Latency Constraints via Multi-Agent Deep Reinforcement Learning. In *2024 International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS)*. IEEE, 9–16.

[9] Anastasios Giannopoulos, Ilias Paralikas, Sotirios Spantideas, and Panagiotis Trakadas. 2024. HOODIE: Hybrid Computation Offloading via Distributed Deep Reinforcement Learning in Delay-Aware Cloud-Edge Continuum. *IEEE Open Journal of the Communications Society* (2024).

[10] Panagiotis Gkonis, Anastasios Giannopoulos, Panagiotis Trakadas, Xavi Masip-Bruin, and Francesco D'Andria. 2023. A survey on IoT-edge-cloud continuum systems: status, challenges, use cases, and open issues. *Future Internet* 15, 12 (2023), 383.

[11] ICOS. July 2024. *ICOS Architectural Design (IT-2).* Project deliverable, EU.

[12] Sebastián A. Cajas Ordóñez, Jaydeep Samanta, Andrés L. Suárez-Cetrulo, and Ricardo Simón Carbajo. 2025. Adaptive Machine Learning for Resource-Constrained Environments. In *Discovering Drift Phenomena in Evolving Landscapes*. Springer Nature Switzerland, DELTA 2024, Workshop at ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2024), Barcelona, Catalonia, Spain, 3–19.

[13] Rosaria Rossini, Terpsichori-Helen Velivassaki, Artemis Voulkidis, Theodore Zahariadis, Panagiotis Karkazis, Dimitrios Skias, and Enric Pere Pages Montanera. 2024. Open Source in NExt Generation Meta Operating Systems (NEMO). In *2024 9th International Conference on Smart and Sustainable Technologies (SpliTech)*. 1–5. doi:10.23919/SpliTech61897.2024.10612369

[14] Olga Segou, Dimitris S. Skias, Terpsichori-Helen Velivassaki, Theodore Zahariadis, Enric Pages, Rubén Ramiro, Rosaria Rossini, Panagiotis A. Karkazis, Alejandro Muniz, Luis Contreras, Alberto del Rio, Javier Serrano, David Jimenez, Maria Belesioti, Ioannis Chochliouros, and Spyridon Vantolas. 2024. NExt generation Meta Operating systems (NEMO) and Data Space: envisioning the future. In *Proceedings of the 4th Eclipse Security, AI, Architecture and Modelling Conference on Data Space* (Mainz, Germany) *(eSAAM '24)*. Association for Computing Machinery, New York, NY, USA, 41–49. doi:10.1145/3685651.3686661

[15] Andrés L. Suárez-Cetrulo, David Quintana, and Alejandro Cervantes. 2023. A survey on machine learning for recurring concept drifting data streams. *Expert Systems with Applications* 213 (2023), 118934.

[16] Eduardo Vyhmeister and Gabriel G Castane. 2024. When Industry meets trustworthy AI: a systematic review of AI for Industry 5.0. *arXiv preprint arXiv:2403.03061* (2024).