Data Orchestration Platform for AI Workflows Execution Across Computing Continuum

Gabriel Ioan Arcas[1][a] and Tudor Cioara[2][b]
[1]Engineering and Data Solutions Department, Bosch Engineering, Center, Cluj-Napoca, Romania
[2]Computer Science Department, Technical University of Cluj Napoca, Romania

Keywords: Data Orchestration, AI Workflow, Computing Continuum, Lambda Architecture, Edge-Fog-Cloud.

Abstract: Cloud AI technologies have emerged to exploit the vast amount of data produced by digitized activities. However, despite these advancements, they still face challenges in several areas, including data processing, achieving fast response times, and reducing latency. This paper proposes a data orchestration platform for AI workflows, considering the computing continuum setup. The edge layer of the platform focuses on immediate data collection, the fog layer provides intermediate processing, and the cloud layer manages long-term storage and complex data analysis. The orchestration platform incorporates the Lambda Architecture principles for flexibility in managing batch processing and real-time data streams, enabling effective management of large data volumes for AI workflows. The platform was used to manage an AI workflow dealing with the prediction of household energy consumption, showcasing how each layer supports different stages of the machine learning pipeline. The results are promising the models are being trained, validated, and deployed effectively, with reduced latency and use of computational resources.

1 INTRODUCTION

The current era is marked by rapid technological progress with vast amounts of data being produced from activities such as social media interactions, online transactions, and IoT devices (Nižetić et al., 2020). New AI technologies such as machine learning emerged to exploit big data, transforming every aspect of our lives, from how we work and communicate to how we interact with our environment. Distributed systems are constantly exposed to a continuous flow of information and need to incorporate advanced processing technologies to achieve effective and reliable AI workflow execution and integration. However, these systems encounter significant hurdles, particularly in handling data processing demanding tasks, achieving fast response times, and reducing latency (Steidl et al., 2023). Moreover, considerable challenges, such as improving response time and minimizing the latency are faced when AI pipelines are executed.

Migrating AI workflows from a cloud-based architecture to a distributed computing continuum requires a fundamental redesign of applications (Rosendo et al., 2022). A major challenge is decomposing tightly coupled monoliths into smaller components that can be distributed toward the edge. Additionally, managing network latency becomes critical when deploying these components on remote resources, unlike in cloud systems where components run in the same data center with low-latency communication (Bulej et al., 2021). In a system distributed over the commuting continuum, components are spread across multiple servers, which can introduce latency and additional communication costs. This necessitates careful analysis of how the components should communicate with each other and the use of technologies such as message queues and service discovery. The decentralization of computational processes towards the edge is interesting for large-scale utility infrastructures such as the smart grid (Ferahtia et al., 2024). In such infrastructure, the adoption of IoT metering devices generates vast amounts of data that are used to optimize the delivery of electricity. AI workflows are used to balance energy supply and demand

[a] https://orcid.org/0009-0002-2246-5928
[b] https://orcid.org/0000-0003-1177-5795

Arcas, G. I. and Cioara, T.
Data Orchestration Platform for AI Workflows Execution Across Computing Continuum.
DOI: 10.5220/0013140600003950

In *Proceedings of the 15th International Conference on Cloud Computing and Services Science (CLOSER 2025)*, pages 103-110
ISBN: 978-989-758-747-4; ISSN: 2184-5042

proactively, by predicting those values in advance. However, they require both batch processing as well as real-time analytics and a lot of information to ingest, making the processing in the cloud inefficient with costs for data migration leading to latency and delay in the decision-making process (Arcas et al., 2024a). Thus, the demand for robust, scalable, and responsive frameworks to handle these complex tasks is more pressing than ever. Additionally, due to privacy constraints, the citizens or electricity companies are reluctant to move their data to cloud data centres preferring to keep them locally to the point of generation (Arcas et al., 2024b). This further emphasizes the need to adopt decentralized data orchestration pipelines that can run AI workflows across the computing continuum layers, while data and computation are distributed towards the edge.

In this paper, we address these challenges by proposing a data orchestration platform for AI workflows that addresses some requirements and exploits the benefits of the computing continuum. Each platform layer has a unique and supportive role: the edge layer, consisting of smart meters and IoT devices, focuses on immediate data collection and preprocessing near the source, reducing latency. Located between the edge and the cloud, the fog layer offers intermediate processing, easing the load on the cloud and facilitating near real-time responses. The cloud layer, with its advanced computational power, handles long-term storage, complex data analysis, and model retraining. This layered approach enhances data management efficiency and ensures that the system can scale effectively to meet the growing needs of contemporary AI workflows. Additionally, the proposed platform incorporates the Lambda Architecture offering flexibility in managing batch processing as well as the real-time data streams. This integration enables the effective management of large data volumes, offering timely insights and alerts while preserving the accuracy and depth needed for AI workflows. For evaluation the platform was used for managing an AI workflow addressing the prediction of household's energy consumption, illustrating how each layer supports different stages of the machine learning pipeline. The edge layer is responsible for data ingestion, capturing raw data rapidly near the source to minimize latency. Additionally, the edge performs basic preprocessing, such as data filtering and normalization, reducing the data volume sent to subsequent layers. This dual functionality ensures both efficient data capture and preliminary processing, which is important for real-time applications.

The rest of the paper is structured as follows: Section II presents the state of the art in data orchestration for AI. Section III describes the design and technologies of the data orchestration platform, along with its mapping to the Lambda Architecture. Section IV presents an evaluation of use cases in the context of a smart grid, specifically focusing on the prediction of household energy demand. Finally, Section V concludes the paper.

2 RELATED WORKS

Federated Learning enables decentralized model training on mobile devices, allowing local data to remain on the devices, which reduces data privacy risks. But in some cases, communication between devices and the server can increase because the model updates are sent back and forth multiple times. To address this, ongoing research focuses on reducing communication overhead, for example, by compressing updates or reducing the number of communication rounds (McMahan et al., 2017). Several additional factors drive efficiency (Sakthidevi et al., 2023). Specialized architectures like Symphony, which focuses on managing data movement through the memory hierarchy, can significantly cut down on unnecessary data transfers and distances. This not only boosts performance but also enhances energy efficiency (Pellauer et al., 2023). Innovative frameworks like Kafka-ML handle machine learning pipelines through data streams, effectively managing the large volumes of continuous data from IoT devices, which improves both training and prediction quality (Chaves et al., 2023). Advanced memory orchestration techniques optimize local and remote memory resources, cutting latency and boosting throughput for memory-heavy machine learning tasks (Bae et al., 2020). Moreover, coordinating tensor movements between GPUs (Lin et al., 2023) can boost speed, especially when dealing with memory constraints.

Orchestrating across the edge, fog, and cloud is key to making data processing more efficient for machine learning applications in smart grids. By tapping into distributed computing resources from edge to cloud, these setups handle the growing data loads from IoT devices, improving response times, energy usage, and network performance in smart grids (Arcas et al., 2024b). Offloading tasks to the edge and fog layers cut down on latency and allows for real-time processing, which is essential for things like predictive maintenance and smart grid management (Belcastro et al., 2024). Plus, integrating

fog computing into smart grids boosts communication efficiency, user satisfaction, and security—using methods like EPri-MDAS to keep data secure, authenticated, and private (Zhang et al., 2024). In short, this orchestration fine-tunes data processing, enabling advanced machine-learning algorithms to run smoothly and reliably in smart grid operations (Shi et al., 2023). By using edge computing devices for real-time data crunching (Siddiqa et al., 2024), you can implement a four-layer framework that combines edge and cloud computing with application layers, thereby ramping up data processing capabilities (Gamal et al., 2024). Moreover, the blend of cloud-edge-fog systems with omnidirectional offloading helps optimize energy consumption and enhances system responsiveness, leading to significant energy savings and lower latency for processing data in smart grids (Kuswiradyo et al., 2024). This approach harmonizes computing resources across layers, making sure that the vast amounts of data needed for machine learning in smart grid environments are processed effectively (Duan & Lu, 2024).

Decentralized machine learning algorithms like EdgeSGD (Kamath et al., 2016) allow smart grids to handle data closer to the source, optimizing bandwidth and ensuring quick delivery of information to decision-makers. Also, using machine learning models like Support Vector Machines at the cloud's edge for anomaly detection strengthens security in IoT environments (Zissis, 2017). Incorporating Hadoop and Spark cloud platforms into smart grid management systems helps to distribute storage, computing, and data analysis. This integration enhances grid monitoring, predicts abnormalities, and pinpoints faults for automated dispatching services (Guo et al., 2017). Altogether, these advancements make automatic dispatching in smart grids more efficient by maximizing the use of monitoring data and uncovering valuable data relationships. By co-designing cloud and edge processing in fog radio access networks, enhanced remote radio heads with local caches and baseband processing can store frequently requested data, optimizing delivery through pre-fetching strategies (Park et al., 2016). Distributed gradient descent approaches allow learning model parameters from data across multiple edge nodes without needing to send raw data to a centralized location, which keeps model training efficient while addressing bandwidth and privacy concerns (Wang et al., 2018). Moreover, modern machine learning techniques like reinforcement learning and neural networks, can automatically create highly efficient scheduling policies for data processing jobs on distributed clusters, significantly speeding up job completion times without requiring manual adjustments (Mao et al., 2019).

Existing approaches to data orchestration in the computing continuum, such as (Sakthidevi et al., 2023), (Pellauer et al., 2023), primarily address either real-time or batch processing but do not provide a unified framework that integrates both. Additionally, many solutions focus on specific layers, such as edge or cloud, without leveraging the full potential of the fog layer for intermediate processing. The lack of robust orchestration frameworks that address scalability, latency, and resource utilization across all three layers (edge, fog, cloud) highlights a significant gap. This work addresses these challenges by proposing a comprehensive data orchestration platform designed to optimize AI workflows across the computing continuum.

3 DATA ORCHESTRATION PLATFORM

3.1 Architecture

Figure 1 shows the data orchestration platform designed and tailored to consider the strengths and requirements of each layer of the computing continuum layer.

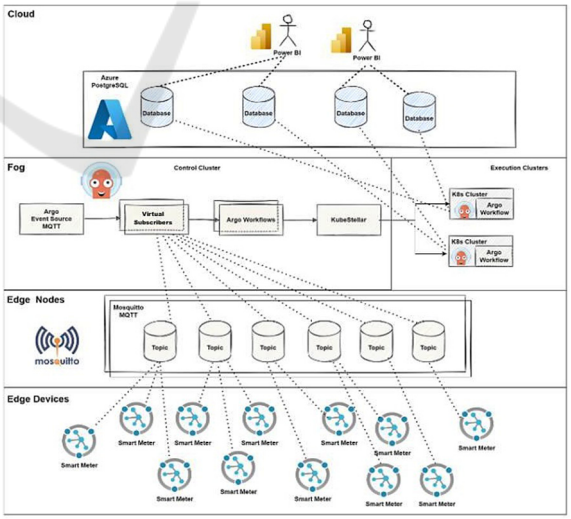

Figure 1: Platform architecture.

The Edge Devices Layer comprises sensors close to the data source with internet connectivity. These devices measure and collect data rapidly and with minimal latency. This data is essential for other

computational layers to execute AI-driven tasks like predicting energy consumption, detecting faults, and performing predictive diagnostics. The Edge Devices Layer acts as the foundational element of the orchestration solution. Therefore, data representation schemas are defined to ensure that the collected data is consistently formatted and facilitates effective analysis and processing in later stages of the architecture.

The Edge Nodes layer functions as a broker, consisting of multiple communication channels. It employs a publish/subscribe model being located close to the Edge Devices. The layer nodes feature low latency, although their computational resources are limited. As a result, a message broker is essential for collecting data from the smart meters. In our setup, the smart meters from the Edge Devices layer act as publishers, while the sensors described in the subsequent layer serve as subscribers. This approach addresses the scalability limitations of traditional client-server applications when dealing with numerous devices. The broker processes messages using an event-driven approach, enabling parallel processing. Additionally, it ensures that the system is reliable and can handle multiple devices with concurrent connections without affecting overall performance. Furthermore, it supports data caching and routes messages to the appropriate subscribers. We use Eclipse Mosquitto (Mosquitto, n.d.) an open-source message broker that implements the MQTT protocol. Its lightweight nature allows for easy deployment across all computing layers, including the edge nodes, ingesting data, and subsequently processing and storing it in the following layers.

The Fog Layer represents the data processing component of the architecture. It includes a set of virtual subscribers to message brokers in the Edge Nodes layer, triggering specific workflows based on incoming messages. These workflows encompass data pipelines, including those for machine learning, ETL processes, and additional types. This layer employs event-driven architecture to facilitate its operations. Positioned closer to the Edge Nodes than data centres from cloud providers, the Fog Layer connects the cloud with the edge, therefore minimizing response time and latency. The Fog Layer includes two main components: the control cluster and the execution clusters, which are organized based on location. To achieve a multi-cluster setup and manage configurations effectively for the Edge, the KubeStellar (KubeStellar, n.d.) is utilized. KubeStellar simplifies the deployment and configuration of applications across multiple clusters. In this architecture, the control of message reception is separated from data processing tasks. This separation enhances response time, reduces latency, and supports parallel processing. Additionally, it enables seamless network communication between clusters, which is necessary as the Workflow engine technology remains consistent across them. The control cluster is responsible for delegating workflow execution to the appropriate cluster. The control cluster contains an event-based dependency manager called Argo Events (Argo Events, n.d.) and a workflow automation framework named Argo Workflows (Argo Workflows, n.d.), both running within Kubernetes (Kind, n.d.). Argo Events facilitates the triggering of computational components and objects within Kubernetes, including Argo Workflows, serverless applications, and more. Argo Workflows define AI-driven operations based on MQTT messages, supporting both Directed Acyclic Graphs (DAG) and step-based workflows. To interface with MQTT brokers from the Edge Nodes layer, with the clusters from the Fog Layer a virtual subscriber is defined. It consists of a set of events, where inputs are messages from broker topics, and triggers correspond to workflows. It uses an event bus to manage dependencies and execute triggers, with dependencies being events that the virtual subscriber waits for to activate.

The Cloud Layer is responsible for the long-term storage of processed data. It connects to databases in the cloud where the results of completed workflows from the Fog Layer are stored. In our platform setup, we have used Azure as the cloud provider and PostgreSQL. This setup enhances data durability and minimizes the risk of data loss, providing robust backup and recovery options. Database connection strings are managed as Sealed Secrets within Kubernetes. Sealed Secrets addresses the challenge of storing sensitive information in version control systems like Git by encrypting and decrypting secrets through a dedicated controller. This controller ensures that secrets can only be decrypted by the controller operating within the target cluster.

3.2 Lambda Architecture Mapping

The proposed data orchestration platform is designed to consider the strengths of the Lambda Architecture, ensuring a robust, flexible, and scalable solution capable of handling data processing requirements specific to AI driven workflows. The Lambda Architecture (Kumar, Y., 2020) seamlessly integrates both batch and stream processing, to manage the challenges posed by massive data volumes. The architecture features three distinct layers. The Batch

Layer is dedicated to processing extensive datasets in batch mode, prioritizing accuracy, and precision. It employs sophisticated algorithms to conduct in-depth analysis of historical data, ensuring comprehensive insights over time. The Speed Layer focuses on real-time data, this layer processes information as it is ingested, facilitating immediate insights and triggering alerts. The main objective is to deliver rapid responses to immediate data fluctuations, facilitating prompt decision-making. The Serving Layer acts as the interface for user queries, this layer integrates the outputs from both the batch and speed layers, presenting a cohesive and unified view of the data.

Lambda Architecture balances the demand for immediate, real-time insights with the need for precise, long-term data analysis. This balance makes it an invaluable framework for big data applications, where both speed and accuracy are critical. The flexible design of the architecture supports straightforward integration with new technologies, maintaining its applicability and efficiency within AI algorithms. Moreover, the use of redundant layers ensures data integrity and fault tolerance, further solidifying its role as a cornerstone in modern data processing architectures (Kumar, Y., 2020).

Our data orchestration platform maps the principles of the Lambda Architecture, particularly at the Edge Nodes and Fog Layers (see Figure 2). The mapping helps in clarifying how the platform will handle different types of data processing requirements such as batches requiring more computational power and storage and stream processing that need low-latency processing, often managed at the edge. The proposed platform ensures that data is efficiently routed to the appropriate processing layer optimizing resource usage and minimizing latency.

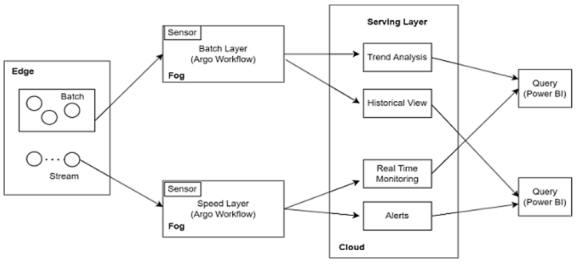

Figure 2: Flow chart for Lambda integration.

At the Edge Nodes level, our platform foresees two methods for data ingestion. For batch ingestion, the data is collected over a specified period being tailored for scenarios that allow for the aggregation of large datasets before analysis. In the case of stream ingestion, the data is captured and processed in real-time, handling each data point as it arrives addressing scenarios requiring immediate processing and analysis, such as real-time monitoring and alerts. Once ingested, data is directed to the Fog Layer, where it is segregated into two distinct processing paths, each triggered by dedicated virtual subscribers. The batch processing path uses a virtual subscriber to trigger workflows that periodically process and store data, generating aggregated reports and enabling deep historical analysis for long-term insights. The stream processing path, by contrast, handles real-time data, triggering workflows for immediate processing of each data point. This ensures responsiveness for live dashboards, instant insights, and alerts. Following the Lambda architecture, the system presents data in two views: batch views provide historical insights for trend analysis, while real-time views update continuously, supporting live monitoring and decision-making.

4 EVALUATION RESULTS

A use case evaluation was conducted on top of the proposed data orchestration platform considering an AI-based workflow for predicting the energy consumption of households in a smart grid. Data Collection takes place on Edge Devices as the smart energy meters collect and transmit raw data. Data Aggregation and Preprocessing run on the Edge Nodes, then it is routed to the Fog Layer. The Data Processing and Model Execution are mapped on the Fog Layer. The control cluster triggers machine learning workflows, and execution clusters handle model training, validation, and inference. Finally, the Storage and Analysis is executed on Cloud Layer as the processed data and model outputs are stored in cloud databases, with additional analysis and model retraining performed as needed.

For evaluation purposes, we have used a data set that contains half-hourly energy consumption readings for 5,567 London households (Greater London Authority, n.d.). It includes approximately 167 million rows of data, totalling around 10GB. The data allows analysis of energy consumption patterns under different pricing schemes, offering insights into consumer behaviour and smart grid management. We have considered one smart meter attached to a household and simulated its functionality. An MQTT broker runs a small cluster with two topics, one for each execution cluster from the Fog Layer. A Python script implements the producer by parsing the data from the data set, formatting it in JSON, and sending batches of 300 measurements to the MQTT topic. On the Fog Layer, we have set up one control cluster and

two execution clusters. The control cluster is based on Argo Events with one event source for MQTT connected to the 2 topics. Argo Workflows is used to define the AI workflow that trains a Linear Regression model to make predictions for each batch of data. On the execution clusters workflow model instances are being deployed and executed based on the triggers received from the control cluster. Each workflow execution takes around 30 seconds. To avoid potential bottlenecks on the Argo Server for each JSON message coming from the MQTT topic, we set a time delay of 30 seconds before dispatching it to the corresponding workload instance new workflow. On the Cloud Layer, we deployed an Azure Postgres Database for storing the master data set and Power BI for querying the data to create visualization.

In total, 300 workflows were executed. Each workflow involved the ingestion of approximately 90,000 events from 160 smart meters. These smart meters were divided into two groups of 80 each, with each group connected to a separate MQTT topic to facilitate workflow execution in one of the two clusters. For analysis, we will randomly select 10 workflows to examine in the subsequent sections. Figure 3 shows the average duration for running the workflows. As can be seen, the execution time was around 38 seconds indicating a stable and predictable AI workflow process in our platform, the impact of workflow changes being minimal.

Figure 3: Execution time per workflow.

In terms of computational resources, each workflow requires on average 2 CPUs and 50 MB of memory to execute (see Figure 4). The memory usage includes both the batch data and the container size, with the container being approximately 30 MB. However, it is notable that the workflow 5 required 4 CPUs. This was due to the batch data containing varying energy data values, which necessitated additional computation by the machine learning model to make accurate predictions.

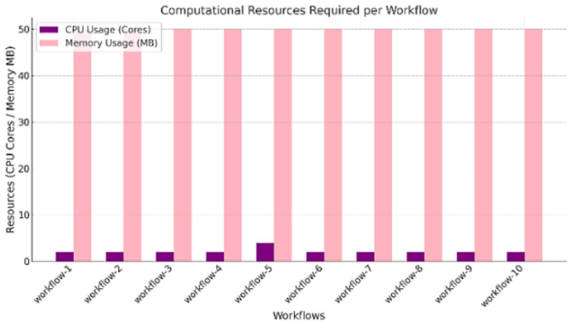

Figure 4: The computational resources per workflow.

In the 2nd experiment, we explored the impact of varying batch sizes of smart meter data on the performance of the AI workflow aimed at predicting half-hourly energy consumption. We tested 300 workflows, varying batch sizes from 10 to 300 events, to assess their impact on execution time and resource usage. Figure 5 illustrates execution times across runs, showing longer durations with larger batches, peaking near 60 seconds. Despite fluctuations, average durations were stable, highlighting the system's adaptability to varying loads. Outliers indicated potential bottlenecks. Future experiments will scale workloads further, testing thousands of smart meters and larger batches to evaluate performance under extreme data volumes. The results affirm that the fog layer's event-driven architecture supports sustained performance and low latency under diverse conditions.

Figure 5: Workflows execution time.

Figure 6 illustrates the computational resources required for each run, specifically CPU and memory usage. The CPU usage remained low throughout, indicating that the AI workflow wasn't particularly CPU-intensive. Memory usage fluctuated as batch sizes grew, leading to a higher demand. This suggests that batch sizing is a crucial factor in processing and has a big impact on the overall performance of workflow execution in our platform. Additionally, the

experiment highlighted a distinct connection between batch size and system efficiency, showing that memory usage is significantly influenced by the volume of data being processed. These insights are essential for optimizing the deployment of AI workloads in smart grid scenarios ensuring timely energy consumption predictions.

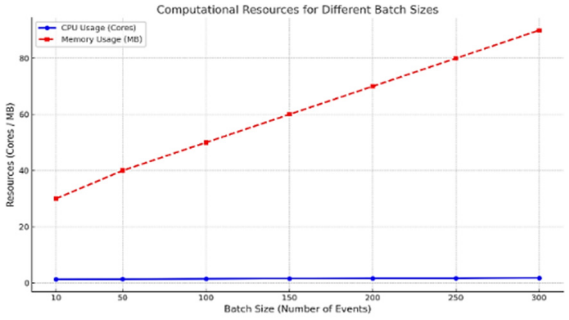

Figure 6: Workflows computational resources usage.

The proposed data orchestration platform's performance was assessed through a comparative study against a centralized cloud-based orchestration system. Table 1 showcases the findings, demonstrating decreases in latency (approximately 20–30%) and enhanced resource efficiency (around 15–25%) achieved by the proposed solution. The fog layer enhances performance by handling intermediate processing, reducing cloud reliance, and enabling faster data handling. This approach minimizes bandwidth use and network delays but introduces resource overhead from additional execution clusters. These clusters, while necessary for workflow management, may affect efficiency in resource-constrained fog environments. Balancing latency and resource use depends on application needs: latency-sensitive tasks benefit more from the fog layer, while batch processing may favour cloud-based solutions to optimize resources.

Table 1: Performance comparison.

Metric	Centralized Cloud	Our Platform
Average Latency (ms)	250	175
Execution Time (s)	45	36
Resource Efficiency (CPU utilization)	80%	60%
Bandwidth (MB)	100	70

The comparative results provide evidence of the benefits offered by distributing computation across the edge, fog, and cloud layers. Specifically, the reduction in processing time and resource utilization highlights the advantages of the computing continuum for real-time and resource-intensive AI workflows.

5 CONCLUSIONS

In this paper, we have proposed a data orchestration platform for AI workflows leveraging the computing continuum features. It enables efficient data orchestration across edge, fog, and cloud layers to maintain minimal delays, rapid execution times, and scalable resource usage which are important factors for AI-driven workflows. Moreover, the orchestration platform effectively aligns with the principles of the Lambda Architecture, particularly within the Edge Nodes and Fog Layers (see Figure 3). This alignment clarifies how the platform will handle varying data processing needs, optimizing performance across different layers of the architecture.

For evaluation, we have considered smart grid scenarios and workflows dealing with energy prediction of household's energy consumption. The findings show that our platform effectively manages data flows and computational tasks, enhancing performance in terms of execution time and resource usage efficiency. Additionally, the results highlight a link between the size of the data batches and memory usage. Larger batches require more memory, while CPU usage remains relatively low, highlighting the importance of batch sizing for efficient handling of varying data loads.

Future investigations will focus on incorporating an analysis of end-to-end latency, which will capture the full duration from data collection to model output. Additionally, evaluating cost efficiency will be essential to understand the trade-offs between computational expenses and prediction accuracy.

ACKNOWLEDGEMENTS

This work has been conducted within the HEDGE-IoT project, grant number 101136216, funded by the European Commission as part of the Horizon Europe Framework Programme and by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI - UEFISCDI, project number PN-IV-P8-8.1-PRE-HE-ORG-2024-0194, within PNCDI IV.

REFERENCES

Nižetić, S., Šolić, P., (2020). Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future. *Journal of Cleaner Production*, 274,

Steidl, M., Felderer, M., & Ramler, R. (2023). The pipeline for the continuous development of artificial intelligence models—Current state of research and practice. *Journal of Systems and Software*, 199, 111615.

Rosendo, D., Costan, A., Valduriez, P., & Antoniu, G. (2022). Distributed intelligence on the Edge-to-Cloud Continuum: A systematic literature review. *Journal of Parallel and Distributed Computing*, 166, 71–94.

Bulej, L., et al. (2021). Managing latency in edge–cloud environment. *Journal of Systems and Software*, 172, 110872.

Ferahtia, S., Houari, A., Cioara, T., Bouznit, M., Rezk, H., (2024). Recent advances on energy management and control of direct current microgrid for smart cities and industry: A survey. *Applied Energy*, 368.

Arcas, G. I., Cioara, T., & Anghel, I. (2024a). Whale Optimization for Cloud–Edge-Offloading Decision-Making for Smart Grid Services. *Biomimetics*, 9, 302.

Arcas, G. I., Cioara, T., Anghel, I., Lazea, D., & Hangan, A. (2024b). Edge Offloading in Smart Grid. *Smart Cities*.

Sakthidevi, I., Sangeetha, A., et al. (2023). Machine Learning Orchestration in Cloud Environments: Automating the Training and Deployment of Distributed Machine Learning AI Model. *I-SMAC*, 10.1109/i-smac58438.2023.10290278

Pellauer, M., Clemons, J., Balaji, V., Crago, N., Jaleel, A., Lee, D., … Emer, J. (2023). Symphony: Orchestrating Sparse and Dense Tensors with Hierarchical Heterogeneous Processing. *ACM Transactions on Computer Systems*.

Bae, J., Su, G., Iyengar, A., Wu, Y., & Liu, L. (2020). Efficient Orchestration of Host and Remote Shared Memory for Memory Intensive Workloads.

McMahan, B. H., Moore, E., Ramage, D., Hampson, S., & Aguera y Arcas, B. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data.

Lin, S. F., Chen, Y. J., Cheng, H. Y., & Yang, C. L. (2023). Tensor Movement Orchestration in Multi-GPU Training Systems.

Belcastro, L., et al. (2024). Edge-Cloud Solutions for Big Data Analysis and Distributed Machine Learning. *Future Generation Computer Systems*.

Zhang, J., Zhang, W., Wei, X., & Liu, H. (2024). EPri-MDAS: An efficient privacy-preserving multiple data aggregation scheme without trusted authority for fog-based smart grid. *High-Confidence Computing*.

Shi, H., Zhao, J., Gu, C., Wang, M., & Huang, H. (2023). Enabling Efficient Multidimensional Encrypted Data Aggregation for Fog-Cloud-Based Smart Grid.

Siddiqa, A., Khan, W. Z., Alkinani, M. H., Aldhahri, E., & Khan, M. K. (2024). Edge-assisted federated learning framework for smart crowd management. *Internet of Things*.

Gamal, M., Awad, S., Abdel-Kader, R. F., & Elsalam, K. A. (2024). Efficient offloading and task scheduling in internet of things-cloud-fog environment. *International Journal of Electrical and Computer Engineering*.

Kuswiradyo, P., Kar, B., & Shen, S. H. (2024). Optimizing the energy consumption in three-tier cloud–edge–fog federated systems with omnidirectional offloading. *Computer Networks*.

Duan, Q., & Lu, Z. (2024). Edge Cloud Computing and Federated–Split Learning in Internet of Things. *Future Internet*.

Kamath, G., Agnihotri, P., Valero, M., Sarker, K., & Song, W. Z. (2016). Pushing Analytics to the Edge.

Zissis, D. (2017). Intelligent security on the edge of the cloud.

Guo, Z., Mu, Y., Yuexing, P., & Gao, X. (2017). Cloud Computing Platform Design and Machine Learning-Based Fault Location Method in Automatic Dispatching System of Smart Grid.

Park, S. H., Simeone, O., & Shitz, S. S. (2016). Joint Optimization of Cloud and Edge Processing for Fog Radio Access Networks. *IEEE Transactions on Wireless Communications*.

Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., & Chan, K. S. (2018). When Edge Meets Learning: Adaptive Control for Resource-Constrained Distributed Machine Learning.

Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., & Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters.

Mosquitto. (n.d.). Eclipse Mosquitto: An open-source MQTT broker. Retrieved from https://mosquitto.org/

Argo Events. (n.d.). Argo Events: Event-based dependency manager for Kubernetes. Retrieved from https://argo proj.github.io/argo-events/

Argo Workflows. (n.d.). Argo Workflows: Open-source container-native workflow engine for Kubernetes. Retrieved from https://argoproj.github.io/workflows/

Kind. (n.d.). Kubernetes IN Docker: Easily run local Kubernetes clusters. Retrieved from https://kind.sigs. k8s.io/

KubeStellar. (n.d.). KubeStellar: Manage workloads across multiple Kubernetes clusters. Retrieved from https://github.com/kubestellar/kubestellar

Kumar, Y. (2020). Lambda Architecture – Realtime Data Processing. *Social Science Research Network*.

Greater London Authority. (n.d.). Smart meter energy use data in London households. Retrieved from https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households

Chaves García, A., Martín, C., Kim, K. S., Shahid, A., & Díaz, M. (2024). Federated learning meets blockchain: A Kafka-ML integration for reliable model training using data streams. In *Proceedings of the 2024 IEEE International Conference on Big Data* (pp. 7677–7686). IEEE. https://doi.org/10.1109/BigData62323.2024.10826034