# A Workflow Management System Approach To Federated Learning: Application to Industry 4.0

Hamza Safri* †, George Papadimitriou‡, Frédéric Desprez§, Ewa Deelman‡

* Berger-levrault, Toulouse, France
† Grenoble University, Grenoble, France
‡ University of Southern California, Los Angeles, USA
§ INRIA, Grenoble, France
E-mail: hamza.safri@{univ-grenoble-alpes.fr, berger-levrault.com}, {georgpap, deelman}@isi.edu, frederic.desprez@inria.fr

***Abstract*—**
**Federated Learning (FL) combined with the Industrial Internet of Things (IIoT) enhances decision-making in industrial settings by leveraging decentralized machine learning (ML) to ensure data privacy, optimize edge computing, and facilitate adaptive model training. However, implementing FL and IIoT presents challenges due to distributed architectures, including communication, data transfer, and file management across wide area networks. This paper addresses these challenges by introducing FL and Clustered FL (CFL) models using Pegasus workflows. Evaluated with real data from airport baggage conveyor systems, it offers practical insights into FL's application in IIoT environments, contributing to advancements in intelligent industrial decision-making.**

***Index Terms*—Federated learning, IoT, Industrial Internet of Things, workflows management system**

## I. Introduction

The fusion of IoT with system maintenance leads to predictive maintenance (PM), transforming industrial equipment management. Through interconnected sensors, real-time data feeds into an intelligent network, enabling proactive monitoring and preemptive action against malfunctions. PM, powered by advanced data analysis, foresees failures, optimizes maintenance schedules, and boosts productivity by preventing costly downtimes. Challenges persist, including scalability, equipment diversity, and data management issues. Meanwhile, FL, introduced by Google in 2017 [1], addresses privacy concerns and data transfer issues by enabling decentralized ML model training. FL has been extensively studied, with a focus on model aggregations [1], [2] and communication protocols [3]. However, limited emphasis on deployment and management aspects hinders its potential in IIoT environments. Bridging this gap is essential to unlock the full potential of FL in predictive maintenance applications.

To address FL's complexities, we propose utilizing a Workflow Management System (WMS), offering programming interfaces for defining workflows, managing inputs/outputs, and ensuring scalability in studies with numerous participants. Despite the benefits of a WMS, their integration in this domain is largely unexplored. Our work investigates the application of WMS in FL through the implementation of horizontal FL, specifically using the Pegasus-WMS [4] in a real-world predictive maintenance (PM) scenario. Pegasus-WMS, an open-source system, is capable of executing workflows across various computing platforms. The paper contributes a model for WMS orchestration in FL, explores model scalability with participant clustering, and evaluates the approach in a real-world industrial IoT application.

The paper is organized into several sections, starting with the current state-of-the-art, focusing on existing works in FL and workflows. It then proceeds to the description of workflows, starting from basic single-round scenarios and advancing to more sophisticated CFL. The subsequent sections describe experiments geared toward evaluating workflow efficiency with real data from airport conveyor systems. The paper concludes with a summary of the findings and potential future work.

## II. Related work

FL represents a promising approach for modern IoT applications and PM [5]. FL can help overcome the need to transfer vast amounts of data over the wide area network to facilitate processing at a data center, and can ensure timely evaluation of physical infrastructure at the edge.

Many papers are exploring the integration of WMS to streamline FL processes and reduce user overhead in resource management and task orchestration, particularly within IoT deployments. Kamble et al. [6] discuss the challenges of WMS implementation in the IIoT domain, emphasizing the utility of dynamic workflows for intelligent and repetitive tasks like learning and model deployment. While FL-specific WMS like NVIDIA FLARE [7] exist, they often require significant manual configuration tailored to FL scenarios. Colonelli et al. [8] utilize Common Workflow Language (CWL) and streamFlow for FL management but lack detailed workflow descriptions and termination options, while Kontomaris et al. [9] face limitations in client numbers and data transfer management clarity when exploring FL scenarios using CWL.

In this work, we explore the development and management of FL workflows using the Pegasus WMS [4]. More specifically we implement horizontal FL using a real case study for PM. We show that our solution using Pegasus can be adapted to multiple types of FL approaches and can be easily scaled to

many resources and participants, addressing the limitations of existing solutions. To demonstrate our solution, we implement two types of workflows (1) a multi-round FL workflow using FedAvg for multi-round FL; and (2) an FL solution known as CFL to assess the system's capability in managing participants and ensuring the smooth execution of jobs.

## III. FL WORKFLOWS USING PEGASUS-WMS

### A. Pegasus-WMS

Pegasus WMS [4] is a widely-used workflow management system that enables users to design workflows at a high level of abstraction. It transforms resource-independent abstract workflows into executable ones during a planning phase, handling data discovery and data transfer tasks. Pegasus supports containers, manages workflow ensembles, and interfaces with various backend storage systems and data transfer protocols, making it suitable for implementing scalable FL workflows across multiple edge resources with hundreds of participants.

### B. Federated Workflow Generation

The process of FL consists of several key steps: (1) participant selection for local training, (2) distribution of global models to them, (3) aggregation of local updates to build a new global model, (4) participant selection for global model evaluation, and decision-making on whether to initiate a new round based on evaluation results. These steps are executed across two levels (Fig. 1): the ***central node***, responsible for generating and submitting workflows using the Pegasus API, and the ***edge nodes***, where participants perform local model training, update aggregation, and global model updates. The central node manages task coordination to ensure seamless execution and data transfer.

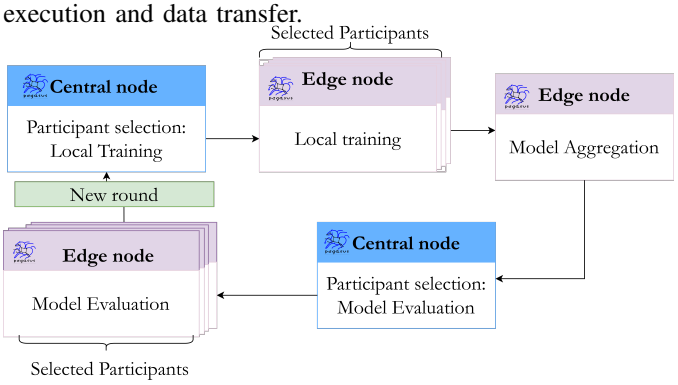


Fig. 1. FL Execution Process.

### C. Multi-Round FL Workflow

*1) Approach:* Fig. 2 depicts a generic multi-round FL workflow structured as a recursive process to accommodate Pegasus' acyclic workflow model. Each iteration represents a single FL round, utilizing the global model from the previous round. This collaborative approach allows for post-workflow output analysis before initiating the next round, focusing on performance metrics and maximum round limits. After each round, the performance evaluation job consolidates evaluation files to determine if a new round is needed, managing input files such as the new global model for local training and participant selection. If predefined thresholds are met or the maximum round limit is reached, the job concludes the FL training process by generating a "noop" workflow, providing flexibility in workflow termination based on specified conditions.

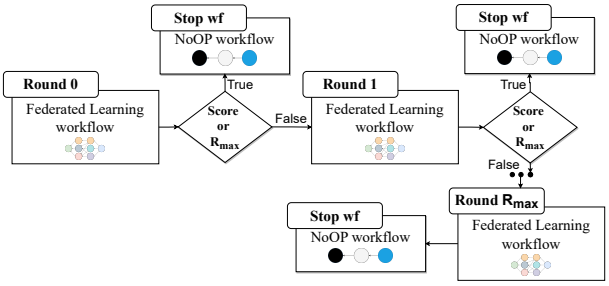


Fig. 2. Multi round workflow overview

*2) Workflow Description:* Before describing the complexities of implementing advanced workflows for FL with multiple rounds or early stopping conditions, in Fig. 3 we present an illustration of FL with a single round.

The workflow begins by selecting participants for two groups: one for local training and the other for global model evaluation. In the "local training" stage, participants independently train their local models using their data and the initial global model. The weights of these local models are then aggregated in the "global model" stage to produce a new global model. This global model is utilized in the "evaluation" stage, where preprocessing, predictions, and metric calculations are performed for each participant. Finally, the "Perf_evaluation" job consolidates model performances, verifies stop conditions, and generates a comprehensive file summarizing the performance of each evaluated client.

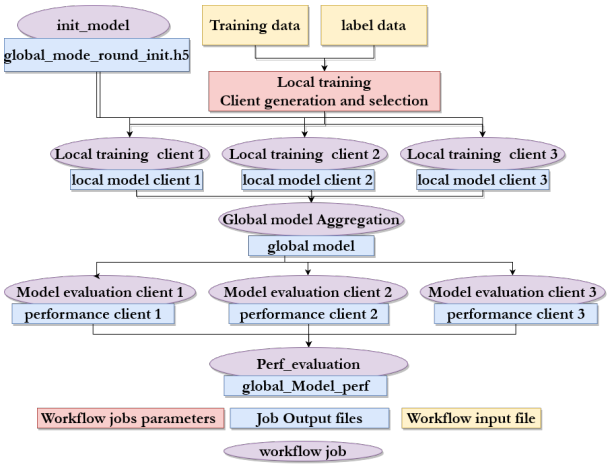


Fig. 3. Single-round workflow: Rectangular shapes for files, oval shapes for edge node jobs, parameters for information shared between jobs.

*3) Multiround ensemble manager implementation:* As outlined in Section III-C2, we have described the workflow for a single round, which forms the cornerstone of our implementation. Subsequently, we utilize the workflows generated at the end of each round to dynamically trigger a new round. Pegasus offers various solutions for this purpose, with one of

them being through the Pegasus Ensemble Manager (Pegasus-EM) service. Fig. 4 depicts the FL implementation using Pegasus-EM, where workflows are organized into separate ensembles, each tailored to its specific logic. In this work, the "FL ensemble" orchestrates the training of N rounds of FL, automatically submitting each workflow through a file pattern trigger. This trigger monitors the next workflow to be submitted, adhering to the pattern *local_wf_round_*.yml* in the workflow jobs output folder. The trigger submits a new workflow by invoking a shell script, showcasing the automation offered by Pegasus-EM. This iterative process continues until reaching the maximum number of rounds or achieving satisfactory model performance. The final ensemble workflow generates a log file with the round number and model performance, serving as a comprehensive record of the FL process.

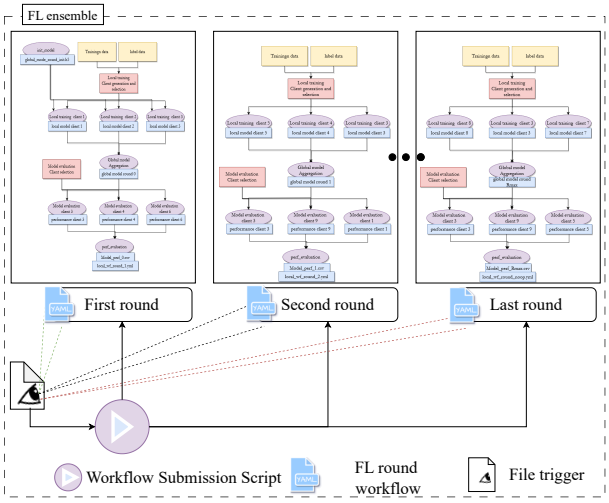


Fig. 4. FL using ensemble manager implementation

### D. Clustered Federated Learning

In [10], the introduction of CFL addresses challenges encountered in constructing a single global model in FL by organizing devices into clusters based on data distributions. In this section, we implement CFL using advanced components of Pegasus, particularly focusing on the Pegasus-EM. This work, employs a clustering approach specifically tailored for IoT applications, which involves a two-phase process: **local clustering**, grouping participants within the same edge based on data characteristics, and **global clustering**, refining these groupings at a higher level by aggregating information from different edges to identify patterns and similarities in participant behaviors. The approach illustrated in Fig. 5 involves executing a series of jobs before commencing the training process.

To accomplish this task, we establish an ensemble as the central entity responsible for executing clustering jobs. The process initiates with "local_clustering_edge_*," where each edge performs local clustering on its managed data, generating result files. These files are then leveraged by the global clustering job to create an overarching clustering file for all connected objects and participating edges. A file trigger monitors the global clustering output, activating a shell script upon detection. The number of clusters isn't predefined, allowing a single cluster to include participants from multiple edges, with edge-related information gathered within the global cluster for Pegasus-WMS job mapping. After acquiring clusters, we generate customized workflows for each, exclusively running on edges containing participants from that cluster. This process integrates data from both local and global clustering, along with edge names obtained from Pegasus-WMS profiles. Before workflow generation, a file trigger dynamically creates a unique ensemble for each global cluster, launching the initial round of FL workflows for each ensemble. Subsequent rounds follow a similar process, with each ensemble operating autonomously through file triggers, ensuring independent operation.

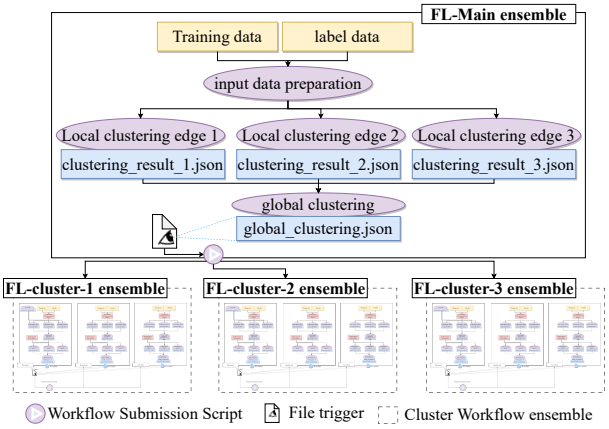


Fig. 5. Clustred FL workflow

## IV. Experiments

### A. Use Case: Conveyors Baggage Detection

This study explores the intricacies of airport baggage handling systems, crucial for air travel efficiency amidst challenges like flight delays and baggage issues arising from their highly automated nature. Leveraging a dataset encompassing two years of lab experiments and over 12 months of operational data from various automated handling systems, the research focuses on optimizing conveyor utilization time. The dataset includes key field descriptions: "Date-Time" for timestamps, "Speed" for speed measurements, "Intensity" for engine intensity readings, "Baggage" indicating presence (1) or absence (0) of baggage, "Running" indicating conveyor operation (1 for running, 0 for not running), and "Onload" indicating if the conveyor is loaded (1) or not (0). This analysis aims to enhance maintenance planning, resource management, and ensure smooth system operation.

*1) Studied Scenarios:* In this paper, we adopt the architecture described in previous works [11], replacing the orchestration component with Pegasus-WMS. Utilizing a geographically distributed infrastructure on FABRIC [12] as shown in Fig. 6, we implement two scenarios for detecting conveyor operation states from real-world deployments using a sequential neural network. The model comprises a Long Short-Term Memory (LSTM) [13] layer with 200 units, a

dropout layer with a 40% dropout rate, and a dense layer with 2 units and softmax activation for binary classification. The **Extreme Edge Architecture** involves each edge managing a single connected object, applying traditional FL workflows to construct global models. Conversely, **the Edge Architecture** features each edge managing multiple connected objects with data collected based on the same schema, utilizing a clustering approach for more scalable and adaptive FL processes.

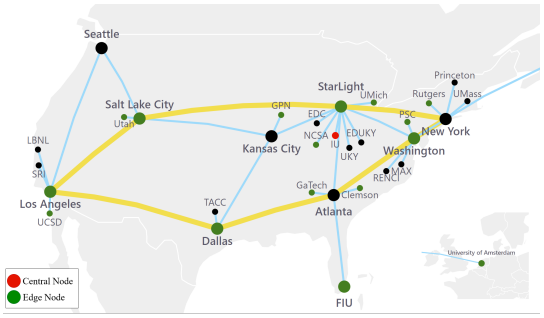


Fig. 6. In our experimental deployment utilizing FABRIC [12], covering 17 sites in the USA and Europe, Indiana University hosted the central node, and each site accommodated 2 edge nodes.

These two scenarios will be analyzed from two perspectives. First, by examining the workflow at different levels, focusing on elements affecting FL parameters such as the number of rounds and participants. Secondly, by evaluating the global model performance, which includes a comparative analysis of FL results from both ML and industrial perspectives. The evaluation involves metrics like accuracy, measuring predicted load periods, and overestimation and underestimation, indicating model performance regarding false positives and false negatives. A superior model exhibits high accuracy close to 100%, with minimal overestimation and underestimation trends approaching 0.

### *B. Results Analysis*

*1) Workflow Analysis:* This paper explores the operational aspects of FL workflows, emphasizing execution details such as participant count, round variations, and host roles in local training and computation evaluation. Three FL implementations are compared: One Round FL, Multi-round FL (10 to 100 rounds), and CFL with host and participant variations. Focusing on a one-round scenario, 25 main jobs are generated for tasks in a PM application, distributed across 11 clients managing connected objects. The workflow includes 90 input files, 47 output files, and 14 auxiliary jobs, resulting in a total of 39 jobs.

In the multi-round FL approach (Section III-C3), the process iterates the one-round FL 100 times, skipping the initialization job since previous models are reused. Consequently, the total number of jobs achievable for a workflow with N rounds can be estimated using $Total\_FL\_Jobs = 25 + 24 \times N$, where 25 represents jobs for one FL round, and 24 excludes initialization tasks. Despite the increase in rounds, empirical observations suggest minimal impact on workflow behavior, with file and job quantities predictable based on one-round FL workflow.

In the CFL scenario, clustering results in four groups: the first with 7 participants across three hosts, followed by two groups with one participant each on single hosts, and a cluster with two participants and one host. The total job count includes jobs from normal FL workflows and those from participants' clustering workflows (5 jobs), reflecting a realistic scenario for the conveyor PM application.

This analysis underscores the critical influence of factors such as the number of participants and the edges where model training occurs on the workflow. Particularly notable is the case of CFL with 2 and 1 participant clusters, which exhibit identical numbers of jobs and file transfers due to participants residing on the same edge. However, the first cluster with 7 participants incurs over 200 jobs due to additional local training and evaluation tasks, resulting in extra output files. The execution time of the workflow is significantly impacted by various parameters, as evident from Table I, including the number of hosts, participants, and rounds. Compared to traditional ML workflows, which take over 2 days for 100 rounds, the CFL approach completes within 1 day, owing to efficient participant distribution and clustering. This substantial reduction in training time suggests potential for further optimization in participant selection to enhance overall execution efficiency.

TABLE I
JOB-LEVEL WORKFLOW ANALYSIS

| Workflows | Clients | Clusters | Clients / Cluster | Rounds | Jobs | Aux Jobs | Input Files | Output Files | Edges | Execution Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| One Round Federated Learning | 11 | 1 | 11 | 1 | 25 | 14 | 90 | 47 | 11 | 1750 |
| Traditional Multi-Round Federated Learning | 11 | 1 | 11 | 10 | 241 | 86 | 900 | 470 | 11 | 17855 |
| | | | | 25 | 601 | 206 | 2250 | 1175 | 11 | 45327 |
| | | | | 50 | 1201 | 406 | 4500 | 2350 | 11 | 91883 |
| | | | | 100 | 2401 | 806 | 9000 | 4700 | 11 | 184681 |
| Multi-Round Clustered Federated Learning | 11 | 4 | 7 | 100 | 805 | 608 | 3300 | 1500 | 3 | 78381 |
| | | | 1 | 100 | 405 | 608 | 1300 | 700 | 1 | |
| | | | 2 | 100 | 405 | 608 | 1300 | 700 | 1 | |
| | | | 1 | 100 | 405 | 608 | 1300 | 700 | 1 | |

Table II summarizes the average resource consumption per job from various workflows. Local model training tasks demand higher CPU and memory resources due to data loading and computations, while global model aggregation and evaluation tasks require significant memory for file and model loading. Job execution times typically range from tens to hundreds of seconds but rarely exceed 3 minutes per job, influenced by factors like model size, data volume, and participant count. Increased number of participants lead to higher resource demands and execution time, especially for tasks like global model aggregation, requiring simultaneous data or model loading from all participants.

TABLE II
RESOURCE-LEVEL WORKFLOW ANALYSIS

| Jobs | Avg. CPU (%) | Peak Mem (MB) | Min Runtime (s) | Max Runtime (s) |
|---|---|---|---|---|
| Local Model Training | 149 | 599 | 57 | 174 |
| Global Model Aggregation | 97 | 528 | 44 | 178 |
| Global Model Evaluation | 96 | 553 | 46 | 150 |
| Performance Analysis | 97 | 100 | 36 | 159 |
| Local Clustering | 95 | 157 | 113 | 113 |
| Global Clustering | 97 | 156 | 150 | 150 |

*2) Model Performances:* After outlining workflow profiling, the evaluation of model performance showcases the advantages of incorporating FL. Fig. 7 illustrates the fluctuations in global model performance across various FL ensembles over 100 training rounds.

In CFL, using four participants per cluster shows clusters behaving similarly. While Figure 7 indicates consistent behavior in each cluster's global model, accuracy remains below 68%. Figure 8 shows load period predictions aren't uniformly accurate across clusters (60%). Notably, load periods are often underestimated (over 80%) and sometimes overestimated (20%). This misclassification could impact subsequent analyses, particularly in conveyor operating times. Improvements could come from adjusting initial global model weights and refining clustering parameters like distribution type and size to enhance model performance.

Traditional FL outperforms CFL, achieving over 78% accuracy with minimal over or underestimations, particularly in the initial 29 rounds. However, accuracy sharply declines afterward, reaching 0%, accompanied by an increased underestimation rate, resulting in solely incorrect predictions. This underscores the effectiveness of traditional FL in developing accurate global models in under 100 rounds, leveraging diverse conveyor data. Leveraging the early stop option in workflows, by utilizing the "Performance Analysis" job to monitor model performance as detailed in Section III-C3, allows for the termination of training once optimal performance is attained, effectively minimizing training time.

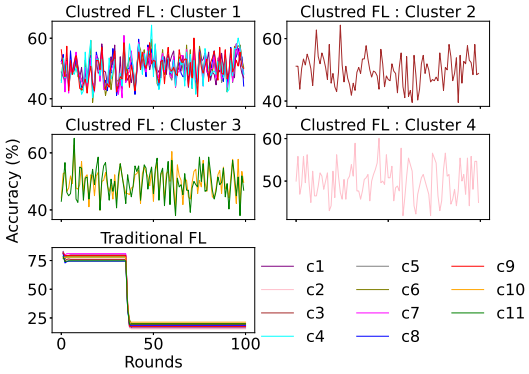


Fig. 7. Accuracy of each global model per round for different participants cluster

## V. Conclusion and Future Work

This paper addresses challenges in FL architectures within IIoT, proposing a novel approach using Pegasus-WMS for FL process development and management. Leveraging Pegasus offers benefits like automated data transfer, task scheduling, and reliable model training execution. We found that clustering participants can reduce model-building time by up to 50%, albeit with a compromise on model accuracy, which never exceeds 68%. We stress the importance of dynamically terminating learning processes to prevent overfitting. Despite showcasing the potential of workflow management systems in FL, we note shortcomings in metric management and model/data versioning for broader FL applications. Further improvements, such as user-level abstractions for modeling training epochs, are necessary.

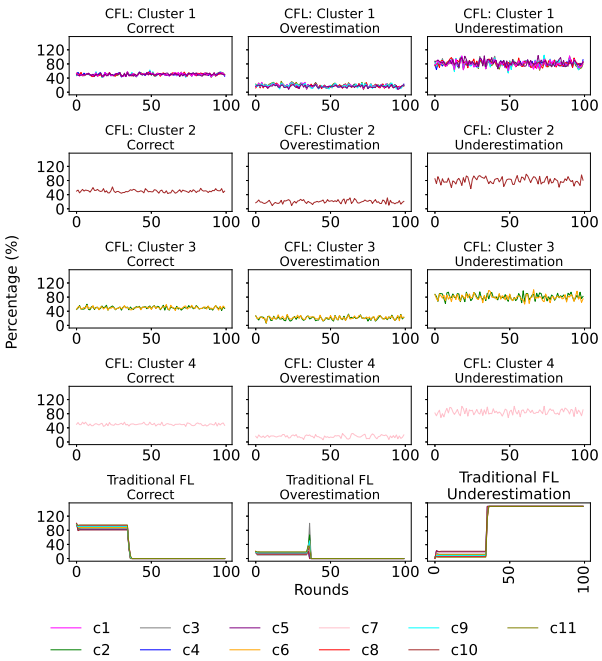


Fig. 8. Industrial performance metrics for the global model.


## Acknowledgment

This work is funded by DOE award #DE-SC0022328 and NSF award #2138286 and #2106147.


## References


[1] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] Y. Li *et al.*, "Secure federated averaging algorithm with differential privacy," in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2020, pp. 1–6.

[3] Y. Chen *et al.*, "Asynchronous online federated learning for edge devices with non-iid data," in *2020 IEEE International Conference on Big Data (Big Data)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2020, pp. 15–24.

[4] E. Deelman *et al.*, "The evolution of the pegasus workflow management software," *Computing in Science Engineering*, vol. 21, no. 4, pp. 22–36, 2019.

[5] V. Pruckovskaja *et al.*, "Federated learning for predictive maintenance and quality inspection in industrial applications," 2023.

[6] N. N. Kamble *et al.*, "Study on workflow management using iot in industry 4.0," in *AIP Conference Proceedings*. AIP Publishing, 2023.

[7] H. R. Roth *et al.*, "Nvidia flare: Federated learning from simulation to real-world," *arXiv preprint arXiv:2210.13291*, 2022.

[8] I. Colonnelli *et al.*, "Federated learning meets hpc and cloud," in *ML4Astro International Conference*. Springer, 2022, pp. 193–199.

[9] C. Kontomaris *et al.*, "Cwl-flops: A novel method for federated learning operations at scale," in *2023 IEEE 19th International Conference on e-Science (e-Science)*, 2023, pp. 1–2.

[10] F. Sattler *et al.*, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 8, pp. 3710–3722, 2020.

[11] H. Safri *et al.*, "A federated learning framework for iot: Application to industry 4.0," in *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2022, pp. 565–574.

[12] I. Baldin *et al.*, "Fabric: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019.

[13] G. Van Houdt *et al.*, "A review on the long short-term memory model," *Artificial Intelligence Review*, vol. 53, pp. 5929–5955, 2020.