CWL-FLOps: A Novel Method for Federated Learning Operations at Scale

Chronis Kontomaris[1], Yuandou Wang[1], Zhiming Zhao[1]
[1]Informatics Institute, University of Amsterdam, the Netherlands
Email: xronistk@gmail.com, {y.wang8, z.zhao}@uva.nl

***Abstract*—Federated Learning (FL) has attracted much attention in recent years because it enables users with private data sets to train a global model collaboratively without raw data exchange. However, due to a lack of automation, researchers often struggled to develop, deploy, track, and manage all the data, steps, and configuration setup for all FL participating nodes. Federated Learning Operations (FLOps) is recently emerging in the FL community, a new methodology for developing FL systems efficiently and continuously. Some research works discussed approaches for FLOps, but only a few solutions address managing FL application scenarios from the workflow perspective. This poster proposes *CWL-FLOps*, a novel CWL-based method for FLOps, which can improve the flexibility of FL abstraction and fully automate the FL deployment and execution by mapping high-level descriptions onto distributed resource nodes. Our experiments demonstrate the feasibility of describing centralized and decentralized FL scenarios using CWL abstracted definitions without relying on heavily customized or external software for execution.**

***Index Terms*—Federated Learning Operations, Common Workflow Language, Cloud computing, Docker**

I. Introduction

Federated learning (FL) recently has attracted a great deal of attention in diverse domains [1], such as healthcare, the internet of drones (IoD), and international financial crime detection. However, running FL efficiently and continuously is still challenging [2] [3]. These challenges mainly focus on the automation and flexibility of FL application scenarios due to the diversity of the FL ecosystem, e.g., different data sets, heterogeneous infrastructures, and various FL source codes. FL libraries and open-source frameworks such as TensorFlow Federated (TFF), NVFlare, FATE, FedML, IBM FL, OpenFL, PySyft, and Flower, have made significant progress from communities and are often sufficient for deploying on diverse resource nodes [4]. However, most frameworks are based on centralized FL with a client/server architecture and often require direct bidirectional communications (e.g., MQTT, gRPC, and MPI) between the aggregator and each client training node. In addition, they cannot be easily extended to support various FL deployment scenarios [5].

Federated Learning Operations (FLOps) focuses on the FL life cycle management and gains much attention because it is essential for running FL jobs and managing any FL system efficiently and continuously [2]. In addition, scientific workflow languages (e.g., Workflow Description Language (WDL) and Common Workflow Language (CWL) [6]) provide a way to describe the dependencies, inputs, outputs, and execution logic of individual tasks or jobs with a high-level description. That makes it easy to scale complex data analysis and machine learning workflows from a single developer's laptop to massively parallel cluster, cloud, and high-performance computing environments.

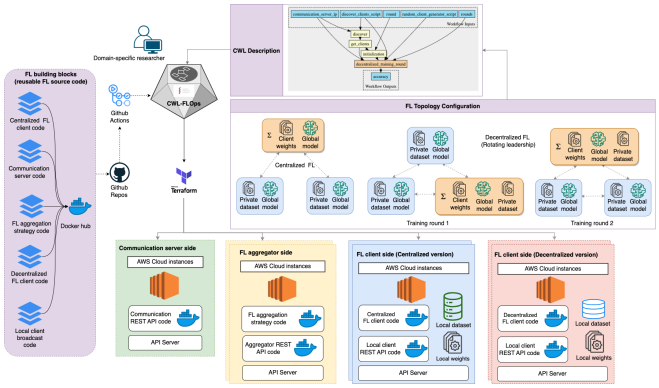

Fig. 1. An overview of our CWL-FLOps architecture.

In this poster, we propose **CWL-FLOps** method based on a MSc Software Engineering thesis, that enables fast deployment for fine-grained FL services and simplified implementation using CWL. This approach makes critical workflow concepts such as automation, scalability, abstraction, portability or flexibility, and reusability come true around the specifics of a FLOps context. We demonstrate its feasibility and our results in the experiment section for showcasing.

II. Problem Statement and Related Work

FL contains several different topologies and general architecture considerations, making supporting their diversity in a FLOps solution difficult. Daga *et al.* [7] presented Flame, which uses a high-level description to map distributed FL services onto specific resource nodes. Colonnelli *et al.* proposed StreamFlow [5] , by describing hybrid workflows under CWL standards on top of heterogeneous and geographically distributed architectures. Similarly, Yang *et al.* [8] proposed an FL life cycle management platform, FLScalize, to continuously update and deploy the latest FL server or client code, enabling an FL CI/CD pipeline. However, Flame [7] does not integrate widely adopted open standards as CWL for FL diverse scenario deployments. In addition, StreamFlow [5] and FLScalize [8] lack strong focus on the variations of FL scenarios and rely on the backend for FL experiment execution.

TABLE 1. The mappings of FL description on CWL patterns.

FL scenarios	Sequence	Stru. Loop	Sync	Simpl Merge	Parallel Split	Gen.Sync merge	MI	Impl. term
Centralized FL	✓	✓	✓	✓			✓	✓
Decentralized FL	✓	✓	✓	✓			✓	✓
Hierarchical FL	✓	✓	✓	✓		✓	✓	✓
Async FL	✓	✓	✓	✓	✓		✓	✓

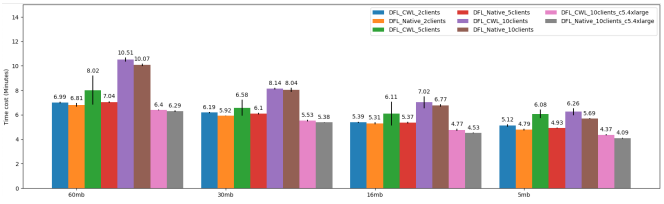

Fig. 2. Comparisons between CWL-based and python-native FL operations on decentralized FL in different data sizes at scale: 1) 2 clients, 2) 5 clients, 3) 10 clients.

We considered 43 control workflow patterns and 40 data flow patterns from the literature to study the feasibility of using CWL-supported workflow patterns to describe various FL scenarios. We observed that only 17 data flow and 8 control flow patterns are being supported based on CWL community research study[1]. To bridge the gap and enhance the supported control flow representations, we conduct our analysis in Table 1 to investigate CWL feasibility for describing the most important FL scenarios (e.g., centralized, decentralized, hierarchical, and asynchronous FL). Based on this analysis, we designed and prototyped our CWL-FLOps framework.

III. CWL-FLOps Framework

A. Architecture

We have designed and implemented the CWL-FLOps architecture in combination with CWL, Github, Docker, and Cloud technologies in a distributed cloud environment (See Fig. 1). It consists of 3 main components: 1) an FL topology configuration with several FL requirements defined in CWL; 2) a cloud infrastructure setup that consists of FL client and server node instances using Terraform; 3) FL client and aggregator source code that is being built, pushed and deployed to all FL nodes using GitHub Actions jobs. Commonly used workflows include exchanging model updates in parallel based on Curl.

B. Experiments and Results

To validate the feasibility of our approach, we employed AWS EC2 instances with t2.small CPUs for deployment and FL training. CWL Workflows are executed locally or using AWS EC2 c5.4xlarge CPU instance to leverage increased parallelization, demonstrating less overall overhead. We utilized the MNIST dataset, partitioned across ten different datasets of sizes 5, 16, 30, and 60 MB, for decentralized and centralized FL workflows without considering deployment time. Fig. 2 demonstrates the average decentralized FL execution time in minutes based on the dataset size, number of clients, and experiment type. Our CWL workflow implementations prove our feasibility and scalability hypothesis in practice. Although CWL tends to introduce overhead in the execution, we strongly argue that its benefits in FL development, re-usability, and automation far outweigh this drawback.

[1]https://github.com/common-workflow-library/cwl-patterns

IV. Conclusion

This poster presents a novel tool named CWL-FLOps that leverages CWL workflow's significant capabilities for automating and abstracting FL scenario description and execution. Our work demonstrates CWL integration in a scalable FLOps pipeline to automate the deployment and execution of several diverse FL scenarios. In the future, we will integrate CWL-FLOps as a component of the NaaVRE framework to extend our automation for Jupyter Notebook users [9].

Acknowledgment

This work has been partially funded by the European Union project CLARIFY (860627), ENVRI-FAIR (824068), BlueCloud-2026 (101094227) and LifeWatch ERIC.

References

[1] Q. Li, Z. Wen *et al.*, "A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 3347–3366, 2021.

[2] Q. Cheng and G. Long, "Federated Learning Operations (FLOps): Challenges, Lifecycle and Approaches," *2022 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pp. 12–17, 2023.

[3] T. Li, A. K. Sahu *et al.*, "Federated learning: Challenges, methods, and future directions," *IEEE signal processing magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[4] L. U. Khan, W. Saad *et al.*, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 3, pp. 1759–1799, 2021.

[5] I. Colonnelli, B. Casella *et al.*, "Federated learning meets hpc and cloud," *ASTROPHYSICS AND SPACE SCIENCE PROCEEDINGS*, pp. 1–6, 2022.

[6] M. R. Crusoe, S. Abeln *et al.*, "Methods included: standardizing computational reuse and portability with the common workflow language," *Communications of the ACM*, vol. 65, no. 6, pp. 54–63, 2022.

[7] H. Daga, J. Shin *et al.*, "Federated learning operations made simple with flame," *arXiv preprint arXiv:2305.05118*, 2023.

[8] S. Yang, J. Moon *et al.*, "FLScalize: Federated Learning Lifecycle Management Platform," *IEEE Access*, vol. 11, no. March, pp. 47 212–47 222, 2023.

[9] Z. Zhao, S. Koulouzis *et al.*, "Notebook-as-a-VRE (NaaVRE): From private notebooks to a collaborative cloud virtual research environment," *Software: Practice and Experience*, vol. 52, no. 9, pp. 1947–1966, Sep. 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/spe.3098