Managing Federated Learning on Decentralized Infrastructures as a Reputation-based Collaborative Workflow

Yuandou Wang^a, Zhiming Zhao^{a,b}

^aMultiscale Networked System, University of Amsterdam, Science Park 900, Amsterdam, 1098 XH, The Netherlands

ARTICLE INFO

Keywords: Federated Learning Collaborative Workflows Incentives Reputation Optimal Contract Blockchain

ABSTRACT

Federated Learning (FL) has recently emerged as a collaborative learning paradigm that can train a global model among distributed participants without raw data exchange to satisfy varying requirements. However, there remain several challenges in managing FL in a decentralized environment, where potential candidates exhibit varying motivation levels and reliability in the FL process management: 1) reconfiguring and automating diverse FL workflows are challenging, 2) difficulty in incentivizing potential candidates with high-quality data and high-performance computing to join the FL, and 3) difficulty in ensuring reliable system operations, which may be vulnerable to various malicious attacks from FL participants. To address these challenges, we focus on the workflow-based methods to automate diverse FL pipelines and propose a novel approach to facilitate reliable FL system operations with robust mechanism design and blockchain technology by considering a contribution model, fair committee selection, dynamic reputation updates, reward and penalty methods, and contract theory. Moreover, we study the optimality of contracts to guide the design and implementation of smart contracts that can be deployed in blockchain networks. We perform theoretical analysis and conduct extensive simulation experiments to validate the proposed approach. The results show that our incentive mechanisms are feasible and can achieve fairness in reward allocation in unreliable environment settings.

1. Introduction

Federated learning (FL) is a promising distributed machine learning (ML) paradigm that enables collaborative learning over decentralized data to mitigate many systematic privacy risks and costs resulting from traditional, centralized learning. In recent years, FL has been studied in many research and application domains, particularly in the fields of digital medicine and health [1], open banking [2], and the Internet of Things [3], in which data cannot be shared or exchanged due to privacy and security concerns.

FL often involves a number of participants, which feature highly heterogeneous training infrastructures and nonindependently and identically distributed (non-IID) data [4]. In a traditional setting of FL, several participants, each keeping its data and training process on-premise, aim to collaboratively train a joint model under a central aggregator that initializes, collects, aggregates, and redistributes the models to and from the training participants, as shown in Figure 1 (a). Over time, the federated model aggregation has emerged in varying modes [5] and performed as different topology types in the distributed computing ecosystem [6], while the data and its training infrastructure are still decentralized. It is known that such diversity can be attributed to different FL design choices and customizations to satisfy different user requirements. For instance, some researchers introduced hierarchical FL, as shown in Figure 1 (b), to tackle the bottleneck of communication overhead in the core

y.wang8@uva.nl (Y. Wang); z.zhao@uva.nl (Z. Zhao)

https://www.linkedin.com/in/yuandou-w-aa717b135/ (Y. Wang); https://staff.fnwi.uva.nl/z.zhao/ (Z. Zhao) ORCID(s): 0000-0003-4694-9572 (Y. Wang) network [7, 8]. Some proposed decentralized FL, as depicted in Figure 1 (c), with a Peer-to-Peer (P2P) communication architecture to cope with the drawbacks of a single point of failure and scaling issues for the increasing network size [9, 10]. Moreover, various aggregation strategies associated with FL topologies for updating the FL model parameters have been studied, including but not limited to sequential updates [11], (a)synchronous parallel updates [12], and peer-to-peer approaches [13], as illustrated in Figure 1 (d), (e), and (f), respectively. The final well-trained model can be obtained through the iteratively collaborative training rounds. However, due to the heterogeneity of decentralized infrastructure or platforms, it poses execution challenges.

FL development involves three typical activities: 1) discover potential participants with high-quality data or resource providers and stimulate them to join the FL, 2) customize the FL application workflows for specific purposes, and 3) define a reliable training process with local updates evaluation.

In real-world scenarios, collaborative FL typically follows two main approaches. The first involves pre-established collaborations, where participants are obligated to join FL due to prior agreements within research projects. These participants are considered trustworthy and contribute to the entire FL training process with complete information. The second approach relies on incentive-driven participation, where FL proposers solicit collaboration through incentives [14, 15] from a decentralized community. In this case, participation is voluntary, and the trustworthiness of potential candidates is initially unknown.

Due to diverse availability of local resources and interests, FL participants may have different motivation in

^bLifeWatch ERIC Virtual Lab and Innovation Center (VLIC), Science Park 904, Amsterdam, 1098 XH, The Netherlands

Topologies in FL

Legend Data Training Node (c) Decentralized FL (a) Centralized FL (b) Hierarchical Fl Aggregator Model updates' patterns in FL Federated Node \sum Model Aggregation $\geq \sum$ Initial Model Final Model Weight Exchange (d) Sequential (e) Parallel (f) Peer to Peer

Figure 1: An overview of diverse FL design choices. FL topologies. (a) Centralized FL: a central aggregation server manages the training process by coordinating iterative training rounds. (b) Hierarchical FL: typically, the FL network has a tree structure with at least three tiers. (c) Decentralized FL: each training node is connected to one or more peers and aggregation happens on the selected node. FL model updates' paths. (d) Sequential. (e) Parallel. (f) Peer to Peer.

joining an FL application, and deliver diverse quality in contributions. This heterogeneity introduces several challenges, including unequal participation [16], free-riding [17], and difficulties in accurately assessing and rewarding individual contributions [18]. Encouraging active engagement from diverse stakeholders, such as data owners, infrastructure providers, and model developers, remains a critical challenge. This requires incentivizing participation in customized FL workflows, facilitating local model update sharing, and ensuring fair contributions to global model development within a decentralized community.

Furthermore, these complexities create efficiency challenges, as substantial time and effort are needed to set up FL workflows, identify suitable participants, and maintain learning quality. In addition, FL operations are vulnerable to both intentional and unintentional threats, including data poisoning, malicious servers, inference attacks, system disruptions, and service unavailability [19]. Addressing these risks is particularly difficult when some participants are unreliable during the training process.

In this work, we address the efficiency challenges in FL development and management within decentralized community settings. We begin by automating FL operations by modeling them as workflows and leveraging workflow engines to orchestrate their execution on remote infrastructures at scale. Our approach emphasizes workflow-based FL management, focusing on FL workflow composition, data owner incentivization, and reliable training processes via combining with decentralized technologies.

The reminder of this article is structured as follows: Section 2 reviews the state-of-the-art reliable FL practices, compares with existing work, and analyzes their research gap. Section 3 describes how we can describe FL as a computational workflow and scale it out to decentralized infrastructure via the CWL open standards. Section 4 introduces a decentralized collaboration framework for FL and illustrates the mechanism design approach and studies the contract optimality to answer research questions. Section 5 details the experiments and the analysis of results; finally, Section 6 concludes this work.

2. Related Work

This section reviews the state-of-the-art reliable federated learning operation practices and explores existing frameworks for managing FL in decentralized infrastructures.

In recent years, several FL frameworks have made notable advancements in managing the FL research and development lifecycle. Yang et al. [20] introduced FLScalize, which extends the machine learning operations (MLOps) concept to generate a baseline model, integrating FL clients, the FL server, and model performance to oversee the entire FL lifecycle. Colonnelli et al. [21] employed the open standard workflow language, CWL [22], to abstract and automate FL applications in a hybrid Cloud-HPC environment. Despite these advances, their approach remains constrained by the traditional client-server architecture of centralized FL, although the open standard workflow language holds significant promise for describing diverse FL workflows [23].

Daga et al. [24] introduced the Flame framework, offering flexibility in the topology configuration of FL applications tailored to the specific deployment context using new high-level abstraction topology graphs (TAGs) that incorporate five types of FL topologies. While both Colonnelli et al. [21] and Daga et al. [24] align closely with our study by utilizing high-level abstractions to enable flexibility in topology-aware FL, a key assumption in many existing works is that potential participants will join the FL process unconditionally and are sufficiently trustworthy to contribute throughout the entire FL training process. Our study diverges from previous works in several critical ways. We explore classic workflow patterns [25] and FL architectural patterns [26], making fundamental workflow concepts — such as automation, scalability, abstraction, portability, flexibility, and reusability — applicable to the context of federated learning operations (FLOps) via the open standard CWL. Preliminary results of this approach have been presented in CWL-FLOps [23].

Furthermore, Cheng and Long [27] introduced a novel methodology called FLOps for managing the FL lifecycle continuously and efficiently. FLOps integrates a range of processes, technologies, and tools to enhance the efficiency and quality of developing and deploying cross-silo FL systems. They highlight that workflow-related approaches, such as metadata engineering, dual deployment, and checkpoints, can help establish and automate FLOps practices, enabling smoother and more efficient operations from an engineering perspective. However, the automation of FLOps remains an open challenge, particularly when addressing security and privacy concerns, which differ from those encountered in traditional DevOps [28] and MLOps [29] practices. Moreover, they do not address the issue of incentivizing participants to make high-quality contributions if the assumption of their willingness and trustworthiness does not hold — this is a crucial factor for maintaining a reliable and secure FL process.

Our work connects to the broader discussion of promoting collaborative fairness among federated participants within decentralized communities, where incentives and fairness have been extensively explored [30, 18, 31]. Wang et al. [32] examined incentive mechanism design in the context of resource allocation for FL clients in Blockchain-based Federated Learning (BCFL). They modeled the problem as a two-stage Stackelberg game under both complete and incomplete information. Using the Shapley value approach, they quantified clients' contributions to the training process. By transforming the game model into two optimization problems and solving them sequentially, they derived the optimal strategies for both players. Gao et al. [33] introduced FGFL, an attack-resistant incentive mechanism for FL that detects and repels abnormal updates, thereby protecting the system in unreliable scenarios. The task publisher rewards efficient workers and punishes or eliminates malicious ones based on reputation and contribution indicators.

While both incentive mechanisms are deemed feasible through experimental evaluation, neither explores the optimal contract design required for creating smart contracts for FL operations. The question of whether the incentives embedded in smart contracts are either over-rewarding or insufficient to effectively motivate participants remains unclear. Kang et al. [34] proposed a joint optimization approach combining a reputation-based worker selection scheme with contract theory to determine the optimal computation resources (e.g., contributed CPU cycles) and corresponding rewards for all participants. However, this approach does not fully capture the entire contribution to the FL training process. Furthermore, although existing works touch on fair rewards, there is a clear need for precise fairness metrics that align with specific incentives, particularly in the context of reliable FL systems.

3. Federated Learning as a Workflow

This section explores the advantages of managing federated learning as a workflow within decentralized infrastructure, addressing the following research question: "How can we abstract FL workflows in an open standard manner and organize their execution on remote infrastructure?"

3.1. Federated Learning

We consider an FL application scenario consisting of $N = \{1, 2, \cdots, n\}$ clients, each client i holds a local dataset $D_i = \{(x, y)\} \sim \mathcal{D}_i$ consisting of s_i data samples, where x and y denote the data sample and its corresponding labels, respectively [35]. The total sample size S_{sample} from N clients is given by $S_{\text{sample}} = \sum_{i=1}^n s_i, \forall i \in N$. Additionally, the union of all client datasets denote by $D = \bigcup_{i=1}^n D_i \sim D$. FL aims to minimize a global loss function $\mathcal{L}(\cdot)$ from the N clients' local loss function $\mathcal{L}_{i \in N}(\cdot)$ through the model aggregation strategy techniques. The overall FL problem can be formulated as

$$\min_{\mathbf{w} \in \mathbb{R}} \mathcal{L}(\mathbf{w}) \quad \text{where } \sum_{i=1}^{N} \frac{s_i}{S_{\text{sample}}} \mathcal{L}_i(\mathbf{w})$$
 (1)

Here, $\mathcal{L}_i(\mathbf{w}) = \sum_{k=1}^{s_i} \ell_i(\mathbf{w}; (x_k, y_k) \in D_i)$ is the expected local loss of the i^{th} client on its local dataset D_i , and $\ell_i(\mathbf{w}; (x_k, y_k))$ is the local loss function of the shared model weights \mathbf{w} on sample data $(x_k, y_k) \in D_i$.

Traditionally, clients train their local models $\mathbf{w}_i^{(t)}$ independently at round t by optimizing the loss function $\mathcal{L}_i(\cdot)$ on their local datasets. Clients validate the trained model using a validation dataset and upload their updated models to an aggregator for federated model aggregation. The aggregator then performs the aggregation strategy and updates the shared global model of all the local model updates. Take the model averaging method, i.e., FedAvg [36], as an example. It is a technique developed to reduce the variance of a global model update by periodically averaging models trained over multiple communication rounds in FL. The model averaging

aggregation is often formulated as,

$$\mathbf{w}^{(t+1)} = \sum_{i=1}^{n} \frac{S_i}{S_{\text{sample}}} \mathbf{w}_i^{(t)}$$
 (2)

where $\mathbf{w}^{(t+1)}$ denotes the updated global model weight for the new round t+1. Then, it will be sent back to all active clients to initialize the next round of local training, iteratively. This process repeats until the global loss converges.

It is known that high-quality data, efficient computation, and reliable communication at each round may contribute to the local model training with high model performance (e.g., high accuracy) and can lead to faster convergence of the local loss function $\mathcal{L}_i(\mathbf{w})$. The faster speed of the convergence of the local training and high-quality model updates will make the convergence of the global loss function quicker, and thus, the training time and cost will decrease for a targeted model performance [34]. Therefore, participants with high-quality data, high-performance computation, and communication efficiency can significantly improve the FL quality and efficiency.

3.2. Federated Learning as a Workflow

We define an FL workflow $F = (\mathbf{w}, N, A, tT)$ as a combination of four random sets of variables: the global model \mathbf{w} , the participant set N, the aggregation strategy A, and the topology type tT. These variables consist of the fundamental building blocks of FL workflows, and users can reconfigure such blocks and customize F with their preferences under an agreement of all participants.

We build on existing solutions to develop a flexible, adaptable approach tailored to end users' specific needs. As a first step, we explored the component containerizer in NaaVRE [37] to create and store FL building blocks — such as models and aggregation strategies — as research assets. For instance, a model developer can build ML models in a local NaaVRE environment using small-scale datasets, performing testing and validation. The model provider can then package the ML model or code files with the FL framework into a container and store them on Docker Hub. It facilitates the global distribution of reusable, containerized applications [38]. This approach enhances reproducibility, portability, and scalability, streamlining the integration of these building blocks into reconfigurable FL workflows [39] and accelerating FL development.

In [38], we primarily used docker-nvidia for local client training with CUDA GPU resources, automating FL pipelines while allowing clients to retain control over their data and computation. For large-scale automated deployment, Docker Swarm serves as an alternative solution. The case study on histological image analysis demonstrated the feasibility of the proposed approach, where we utilized the Flower framework [40] within the Jupyter environment for FL code development. However, most existing frameworks, such as TensorFlow Federated (TFF)¹, NVFlare², IBM FL [41],

OpenFL [42], PySyft [43], and Flower, follow a centralized FL paradigm with a client-server architecture, in which the aggregator cannot be replaced or changed during the training process. They typically rely on direct bidirectional communication (e.g., gRPC [44]) between the aggregator and client training nodes, which limits their flexibility in supporting diverse FL deployment scenarios. For example, locations of decentralized data infrastructure can be heterogeneous, exposing different hardware resources and protocols for authentication, communication, resource allocation, and job execution. Plus, they can be independent of each other, meaning that direct communication among them may not be allowed [21].

To overcome this limitation, we model the FL pipeline as a computational workflow and utilize a workflow engine to orchestrate its execution across remote infrastructures. Specifically, we employ CWL open standards to define FL pipelines, and make them more manageable to automate and parallelize the joint model training from the local computer to the remote cloud environments. We first employ the classic workflow patterns to describe diverse computational FL workflows. Table 1 introduces a taxonomy of 43 control patterns and 40 data patterns identified in the Workflow Patterns Initiative (WPI)³, mapping them to a diverse range of FL topology types.

A review of the 43 WPI patterns related to the controlflow perspective reveals that 11 patterns are supported by CWL v1.2 or earlier. Of the 40 data patterns, only 17 are supported by CWL constructs, and just 2 of the 43 resource patterns are covered. Additionally, CWL standards offer limited support for exception handling and do not address the event log imperfection patterns defined in the WPI patterns. For workflow presentation patterns, 19 are supported by CWL v1.2 or earlier, with one pattern still unsupported. Certain environments, such as blockchain networks, mobile devices, and wireless systems, cannot directly interface with CWL-supported workflow patterns.

FL requires loop patterns for describing the iterative training. While CWL v1.2 lacks loop constructs and recursion support, there are alternative methods to enable loops in describing iterative FL workflows. These include 1) combining sub-workflows with loop extensions in the current stable version (v1.2) and 2) the upcoming v1.3.0dev1 version of CWL, which is expected to introduce loop constructs in a future stable release. Figure 2 shows the snippet of the main steps of a decentralized FL workflow example written in CWL version v1.2. It orchestrates multiple steps, including service discovery, client reading, initialization, and decentralized training rounds. This CWL snippet set up a loop within a sub-workflow named decentralized_training_round. It manages decentralized training rounds in FL, iterating until the specified number of rounds is completed. Unlike a fixed aggregator in the centralized FL workflow, the aggregator is randomly chosen from the distributed clients to perform federated

¹https://www.tensorflow.org/federated

²https://github.com/NVIDIA/NVFlare

³http://www.workflowpatterns.com/

Table 1The mappings of different FL topology types on WPI's workflow patterns.

			FL Topology Types											
			Star		Tre	ree D		ecentralized		Minor				
			Synchonous	Asynchronous	Hierarchical	Oynamic	De Mesh	De Wireless	Blockchain	Ring	Clique	G_{rid}	4000	Seni. Ring
		Basic control (5/5)	1	1	1	1	1	1	1	1	1	1	1	1
		Advanced Branching and Synchronization $(1/14)$	1	1	1	1	1	1	1	1	1	1	1	1
	(43)	Iteration $(3/3)$ –>loops extension	1	1	1	1	1	1	1	1	1	1	1	1
		Multiple Instance $(1/7)$	1	1	1	1	0	1	1	0	1	1	1	1
Patterns	Control	State-based $(0/5)$	0	1	0	1	1	1	1	1	1	1	1	1
	ပိ	Trigger (0/2)	1	1	1	1	1	1	1	1	1	1	1	1
		Cancelation and Force Completion $(0/5)$	0	1	1	1	1	1	0	0	0	1	1	1
7		Termination $(1/2)$	0	0	1	1	1	1	0	0	1	1	1	1
		Data Visibility (4/8)	1	1	1	1	0	1	1	1	1	1	1	1
	(40)	Internal Data Interaction (5/6)	1	1	1	1	0	1	1	1	1	1	1	1
	ta (External Data Interaction (0/12)	1	1	1	1	0	1	1	1	1	1	1	1
	Data	Data Transfer (3/7)	1	1	1	1	0	1	1	1	1	1	1	1
		Data-based Routing (2/7)	1	1	1	1	1	1	1	1	1	1	1	1

```
#!/usr/bin/env cwl-runner
   cwlVersion: v1.2
   class: Workflow
      discover:
        run: service_discovery.cwl
          communication_server_ip: communication_server_ip
discover_clients_script: discover_clients_script
10
11
        out: [service_discovery_output]
12
13
      get clients:
        run: read_clients.cwl
15
          datafile: discover/service_discovery_output
16
17
        out: [client_list]
18
19
      initialization:
        run: initialize_decentralized.cwl
21
        scatter: client_url
22
          client_url: get_clients/client_list
24
          round: round
        out: [initialize_decentralized]
25
27
      decentralized training round:
28
        requirements:
          cwltool:Loop:
30
31
             loopWhen: $(inputs.round < inputs.rounds)
32
               round:
33
                 valueFrom: $(inputs.round + 1)
             outputMethod:
```

Figure 2: Snippet of the CWL workflow.

model aggregation in the decentralized training round design. The loop feature implements a dynamic mechanism to handle multiple training rounds, ensuring scalability and flexibility in FL operations. Additionally, it can be fully automated using GitHub Action workflows. More details can be found in the source code, which is available in the GitHub repository⁴.

3.3. Demonstration of FL Workflow

We demonstrate how the workflow-based approach enables scalable automation of FL operations across cloud infrastructures. Additionally, we integrate the implemented CWL-supported FL workflows with the Jupyter environment to facilitate collaborative learning and seamless workflow management. Some related preliminary results have been published in [45, 23].

Visualized CWL-supported FL workflow. Figure 3 displays the screenshot of the visualized CWL workflow graph named decentralizedFL.cwl via the CWL viewer tool⁵. It has been verified with cwltool version 3.1.20230201224320 and consists of five inputs (highlighted in a blue rectangle), three steps (highlighted in a yellow rectangle), and a nested workflow (highlighted in an orange rectangle). The connections between the workflow inputs and steps illustrate the dependencies within the process. This workflow can serve as a research object bundle, enriched with comprehensive metadata and licensing information to ensure its reusability. For example, it is publicly available as an open-source workflow and can be reused under the terms of the Apache License 2.0.

FL workflow execution on hybrid Clouds. To demonstrate the FL workflow execution over decentralized data infrastructures, we utilized a well-known dataset named MNIST⁶ and five cloud instances from different providers. These included one *t4g.small* instance (2 ARM-based vC-PUs, 2 GiB memory) and *t3.small* instance (2 x86-based vC-PUs, 2 GiB memory) from Amazon Web Services (AWS), one *t2a-standard-1* instance (1 ARM-based vCPU, 4 GiB memory) and *e2-medium* instance (2 x86-based vCPUs, 4 GiB memory) from Google Cloud Platform (GCP), and one *lab.uvalight.net* instance (2 x86-based vCPUs, 2 GiB

 $^{^4}$ https://github.com/CWL-FLOps/DecentralizedFL-CWL/

⁵https://view.commonwl.org/

 $^{^6}$ https://git-disl.github.io/GTDLBench/datasets/mnist_datasets/

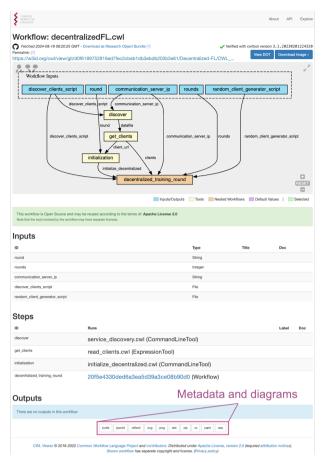


Figure 3: The screenshot of the visualized FL workflow in the CWL viewer with detailed license, specification, and metadata.

memory) from the OpenLab infrastructure provided by the university. Each dataset stored on the cloud instances was split from the MNIST dataset, with a size of approximately 60 MB. The FL source code is based on PyTorch, reproduced from [46], where the author used the FedAvg method for the federated model aggregation. We re-configured the FL workflow with ten local training epochs and 12 communication rounds. Each setup was executed ten times to collect results for statistical analysis.

The average training time is approximately (9.786 \pm 0.238) minutes, and the average test accuracy achieved by the FL workflow execution is $(98.127 \pm 0.085)\%$. These empirical results demonstrate that CWL-supported FL workflows are scalable, reusable, and portable, making them suitable for execution in off-chain decentralized infrastructures such as hybrid cloud environments. Since the CWL community provides standardized alternatives for defining portable and reusable workflows that remain engine- and vendor-neutral, any CWL-compliant workflow execution engine should be able to execute this standardized workflow description and produce consistent results, regardless of the underlying infrastructure [47]. Furthermore, by incorporating the principles of FAIR computational workflows within CWL, we can advance toward realizing FAIR FL workflow management in future work.

3.4. Lessons and Challenges

The empirical results demonstrate that CWL effectively describes FL workflows and automates their execution using cloud technologies. However, this approach assumes a preestablished collaboration, where distributed data sources are predefined within the workflow.

CWL does not inherently address trust and reliable collaboration among multiple participants in FL workflows. The decentralized nature of data providers challenges the traditional centralized workflow management paradigm, where providers are predefined for composing application-specific FL workflows. To overcome this, an effective collaboration framework is needed to enable the dynamic selection of data or resource providers and ensure the quality of their contributions.

In the next section, we will explore how to incentivize collaborations and ensure training quality by addressing the following research questions:

- How can we incentivize participants with high-quality contribution to join the FL training process?
- How can we assure training quality from decentralized participants in an unreliable environment?

4. Decentralized Collaboration Framework

To tackle the challenges identified in 3.4, this section introduces a decentralized collaboration framework that aims to manage the dynamic collaboration among participants within a decentralized community. The basic design ideas include:

- Using blockchain environment to manage the decentralized relation among FL participants and to record their interaction history.
- Managing the collaboration agreement by employing the smart contract of the blockchain, where an FL developer can explicitly describe the conditions for desired participants in a smart contract.
- Verifying the contribution through blockchain ledgers and manage the update of the FL aggregations via dynamic consensus among participants through smart contracts.

4.1. System Overview

Figure 4 depicts the basic idea for FL applications, building on our previous work on the D-VRE framework [48].

This framework equips Jupyter users with essential components for enabling robust FL collaboration, integrating a personal Jupyter working environment, a blockchain network, and off-chain legacy resources, such as data storage, computation, and networking, within a decentralized ecosystem. It enables a group of Jupyter users with different roles to engage in scientific community activities and create FL building blocks as research assets. This is achieved through the *Create FL Building Blocks* function (Component Containerizer within NaaVRE).

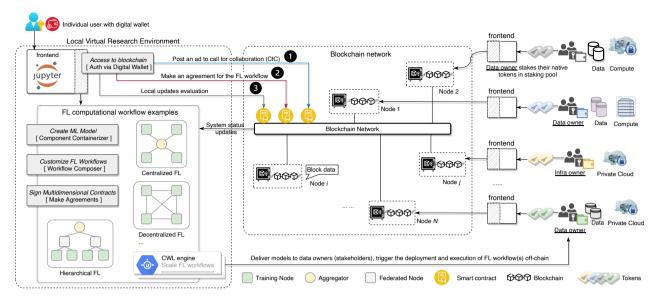


Figure 4: This is an overview of the system for FL applications. A local virtual research environment enables a user to develop ML models on-premise and unlocks the potential of establishing robust collaboration in a decentralized environment via ① ② and ③. It consists of two main interfaces: 1) via blockchain interfaces, users can access to the blockchain network to call for collaboration, make an agreement, and evaluate local updates on-chain; 2) via workflow runtime interfaces, users can create ML models, customize FL workflows, operate collaborative workflows on decentralized infrastructure.

Through the function *Access to blockchain* (Auth via Digital Wallet within D-VRE), the system allows users to configure their digital wallets like MetaMask, to unlock the decentralized Web and applications to call for collaboration and secure asset sharing in a decentralized environment [48].

Due to the diverse data access restrictions, data can not be moved out of the institutions [38]. The system allows model providers (also called FL task publishers) to Call for Collaborations (CfCs) to enable secure data sharing, computing resource sharing, and collaborative learning from the decentralized community, via the *Sign multidimensional contracts* function (Make Agreements within D-VRE). Data or resource owners can respond to the CfCs via the interfaces embedded in their personal frontends.

Each CfC will include specific requirements, e.g., the data and resource specifications, required stakes, targeted model performance, minimum number of participants, and what the publisher can offer (such as reward), rules of penalty, and reputation updates resulting from different contribution behaviours. These requirements can be explicitly described in a smart contract. The stake is related to crypto staking in the blockchain system, which is the practice of locking participants' digital tokens to a blockchain network to earn rewards. For instance, a participant's stake can be returned later with an additional reward if there is no indication of malicious behaviours [49]. The stake plays a crucial role in reliable FL, as 1) it can increase the cost of attacks and 2) promote risk-sharing during FL system operations. The candidate has to join the contract to make agreements.

Via the function of *Customize FL workflow* (Workflow Composer and CWL-based FL workflow composition), users who signed contracts acting as FL participants can

customize a computational FL workflow through verified resources from the off-chain decentralized (data) infrastructures. We assume that data are stored in the infrastructures which provide computation capacity and network for communication. Some participants may only contribute to infrastructure, e.g., computing, storage, and network resources. Each participant can be identified and verified for their resources through trusted external committees, such as Oracle and DAO [50]. Hence, the composition of computational FL workflows can combine the on-chain and off-chain services to scale different FL scenarios.

After the FL workflow with on-chain engagement has been confirmed, the system generates a number of smart contract instances to enforce the contract items and ensures that precise, mutually agreed-upon terms govern collaborations in the decentralized network. The system enables users to start the CWL engine for automating FL workflow deployment and execution over legacy data infrastructures while enabling secure and transparent execution in the blockchain environment. Since the block data has stored ledgers about FL actions for all participants in the blockchain network, it can correlate the captured data with user behaviours to maintain a transparent record of any changes made to an FL operation. Finally, users can evaluate local updates of their FL workflows related to domain-specific subjects and analyze the consequences of participants' actions regarding relevant task contributions, reputation updates, and rewards.

4.2. How to Incentivize Collaboration?

In real-life scenarios, the FL task publisher does not know which users in the system would join the FL training due to the lack of prior information. The local data quality and the computing power of available resources from potential candidates are unknown to the publisher owing to data privacy and security concerns. Therefore, it is essential for the task publisher to design an efficient incentive mechanism to stimulate active engagement from high-quality candidates while reducing the over-reward risks caused by the issues of asymmetric information [34, 51]. This study employs the contract theory with a multidimensional scheme as an efficient approach to address the incentive collaboration problem.

Assume a set of \mathbb{N} potential candidates responds to the CfC task in the system. Let $N=1,2,\cdots,n$, where $N\subseteq\mathbb{N}$, represent the set of identity-verified candidates who have signed contracts with stakes $S=S_1,S_2,\cdots,S_n$ and initial reputations $r^0=r_1^0,r_2^0,\cdots,r_n^0$. These candidates act as FL participants, continuously contributing to the FL network. Each participant's reputation is dynamically updated based on their behaviour.

Let θ denote a standard type space to differentiate the participant nodes. Suppose participants are classified into N types, which can be sorted in a descending order regarding contribution types: $\theta_1 > \theta_2 > \cdots > \theta_n, i \in \{1, 2, \cdots, n\}$. Let $\hat{\theta}_i$ be the claimed type by participant i with the true θ_i . We assume that candidates who join the FL must claim their types $\hat{\theta}$ and select the corresponding contract item $\phi_{\hat{\theta}}$. Although the task publisher does not know about the true type of a given participant due to the information asymmetry, it has information about the reported type $\hat{\theta}_i$ by the participant with the true type θ_i (e.g., data and infrastructure specifications), from which the publisher can inform the probability that a participant belongs to a particular type θ_i with reported (meta)data and computing power categories, denoted as $\sum_{i=1}^N p_i(\hat{\theta}_i) = 1$.

For participants with different contributions, the task publisher signs different contract items with them. We define the contract $\mathcal{C}=(T_{\max},\Phi)$ is comprised of a maximum waiting time T_{\max} and contract terms $\Phi=\{\phi_i\}_{i\in N}$. T_{\max} denotes the maximum allowable time for receiving participants' contributions. Each contract item $\phi_i\triangleq(C_i,S_i,R_i)$ specifies the relationship among each type-i participant's stake S_i , contribution C_i (often associated with a completion time $\tau_i\leq T_{\max}$), and its corresponding reward R_i . Any participant who completes their contributions to the FL network within the required time during each communication round will receive a reward R_i . Conversely, participants whose completion time τ_i exceeds T_{\max} will receive a zero reward, resulting in a zero contract instance.

Contribution model. We define a contribution model for each FL workflow. Let $C_i^{(t)}$ denote an estimated contribution based on the i^{th} participant's local model training f_{local} and its behaviours $g_{\text{behaviour}}$ at round t, which can be formally defined as:

$$C_i^{(t)} = f_{\text{local}}(D_i, \mathbf{w}_i^{(t)}) \cdot g_{\text{behaviour}}(h_i^{(t)}), \quad \forall i \in N$$
 (3)

where $C_i^{(t)} \ge 0$ and $h_i^{(t)}$ is the historical behaviour records regarding reputation and stake which can be used to differentiate honest participant nodes and malicious ones.

Let T_{\max} denote the upper limit on the duration in FL synchronous settings, within which participants must submit their contributions, such as local model updates $\mathbf{w}^{(t)}i$, before the aggregation is triggered within the federation. Specifically, participants with $\tau_i \leq T_{\max}$ can complete the submission of model updates to allow the aggregation of federated models to proceed on time per communication round. We define the contribution value function $V(\cdot)$ for any participant $i \in N$ as follows:

$$V(C_i^{(t)}, \tau_i) = \frac{X_c}{\tau_i} \cdot q_i^{(t)}, \quad q_i^{(t)} = \sigma(\frac{C_i^{(t)} - C_{\min}}{C_{\max} - C_{\min}}) \quad (4)$$

where τ_i is the time cost for participant i to make contribution $C_i^{(t)}$, and $\frac{X_c}{\tau_i}$ represents the unit price of the contribution for participant i, where X_c is a constant factor representing the value associated with the contribution. We define $q_i^{(t)}$ as a dynamic quality parameter via the sigma function $\sigma(\cdot)$, where C_{\max} represents the saturation threshold for contribution, and C_{\min} is the minimum required contribution. A higher value of $V(\cdot)$ indicates better quality in the local training contribution, leading to fewer training iterations needed to reach a targeted model performance score (e.g., accuracy, precision, recall, or F1 score).

Utility function of participants. For a signed contract ϕ_i , we define the utility function of the type-i participant at round t as follows,

$$U_c(\phi_i|\theta_i) = R_i \mathbb{I}_{\text{compl}} - \lambda_s S_i (1 - \mathbb{I}_{\text{compl}}) - c(C_i^{(t)})$$
 (5)

$$\mathbb{I}_{\text{compl}} = \exp(-\sum_{k=1}^{K} \omega_k \cdot \text{VSL}_k). \tag{6}$$

The compliance condition, $\mathbb{I}_{\text{compl}} \in [0, 1]$, is derived from the violation levels. For example, minor violations (e.g., occasional timeouts) will result in a partial retention of proceeds, while severe violations (e.g., malicious behaviour) will reduce the proceeds to zero. The normalized violation level for the K violation types is denoted by $\text{VSL}_k \in [0, 1]$, with ω_k representing the weight of the k^{th} violation type. A value of $\mathbb{I}_{\text{compl}} = 1$ indicates no violations, while $\mathbb{I}_{\text{compl}} = 0$ signifies severe violations.

The parameter λ_s defines the penalty factor applied to the i^{th} participant's stake, S_i . If a node operates without violations, a proportion of the stake, $\lambda_s S_i$, is allocated to the system's risk reserve. However, if violations occur, the participant risks forfeiting the entire stake, $\lambda_s S_i$. Additionally, $c(\cdot)$ represents the effort cost for contributing $C_i^{(t)}$ with time τ_i by the type-i participant. This cost can be further extended to more complex expressions by factoring in the resource utilization required for model training, validation, and aggregation.

Profit function of the task publisher. Task publisher's profit derived from a type-*i* participant is influenced by

multiple factors. Although a high-quality contribution can increase the task publisher's profit, it also incurs a higher reward cost for the publisher. In addition, any violations (e.g., timeouts, malicious actions, or mismatched behaviours) by participants will result in a deduction from their stake (or tokens) as penalties, which are then allocated as profits to the publisher. Therefore, we define the profit function obtained from participant *i* as follows:

$$\pi(\phi_i) = (V(C_i^{(t)}) - R_i) \mathbb{I}_{\text{compl}} + \lambda_s S_i (1 - \mathbb{I}_{\text{compl}}), \quad (7)$$

where $V(C_i^{(t)}) - R_i \ge 0$, the profit is positive; otherwise, the publisher risks incurring a negative profit, even if no violations occur from any participant i.

To make the contract feasible, it must satisfy the following constraints simultaneously.

Definition 1 (Individual Rationality (IR)). Let $U_c(\phi_i|\theta_i)$ be the utility for the type-i participant under the contract ϕ_i . Each type-i participant achieves the non-negative utility if it chooses the contract item ϕ_i that is designed for its own type θ_i , which is given by

$$U_c(\phi_i|\theta_i) \ge 0. \tag{8}$$

Definition 2 (Incentive Compatibility (IC)). Each participant i achieves the maximum utility by truthfully reporting its type θ_i if it chooses the contract item ϕ_i that is designed for its own type θ_i rather than other types θ_{-i} in contract items $\phi_{\theta_{-i}}$. The mechanism is incentive compatible if

$$U_c(\phi_i|\widehat{\theta}_i = \theta_i, \phi_{\theta_i}) \ge U_c(\phi_i|\widehat{\theta}_i \ne \theta_i, \phi_{\theta_i}), \tag{9}$$

$$U_c(\phi_i|\widehat{\theta}_i = \theta_i, \phi_{\theta_i}) \ge U_c(\phi_i|\widehat{\theta}_i = \theta_i, \phi_{\theta_i}), \tag{10}$$

where $\theta_{-i} \cup \theta_i = \theta$, $\theta_{-i} \neq \theta_i$, and $\theta_i \geq \theta_j$ for all participants $i, j \in N$.

As a rational individual, the task publisher aims to maximize its profit in the system [52]. Therefore, the total incentive problem is given by

$$\max \Pi(\Phi) = \sum_{i=1}^{N} p_i(\widehat{\theta}_i) \cdot \pi(\phi_i), \quad \forall \phi_i \in \Phi \quad (11)$$

s.t.,
$$IR(Eq. 8)$$
 and $IC(Eqs. 9, 10)$ constraints. (12)

To mitigate the risks of over-rewarding or insufficient rewards, it is crucial to examine the optimality of the contract problem.

4.3. How to Assure Training Quality along with Contract Optimality?

This section first employs a mechanism design approach that integrates a reputation system and dynamic incentive indicators within contracts to ensure training quality in an unreliable environment. Then, leveraging a quantified reward measurement, we analyze the optimality of the contract problem. Figure 5 illustrates the key steps involved in

assuring training quality, including reputation-based committee selection, malicious detection and penalty, reputation updates, and reward calculation.

Reputation-based committee selection. We introduce a committee of high-reputable and highly effective nodes among participants to do extra work like being responsible for partial model validation and aggregation. Let $\mathcal{M}^{(t)}$ be the final selected committee in each round t, where $\mathcal{M}^{(0)} = \emptyset$ and \mathcal{K} is the committee size. We use a stratified sampling method [53] based on reputation $r^{(t)} = \{r_1^{(t)}, r_2^{(t)}, \cdots, r_n^{(t)}\}$ and a cooling-down mechanism, to balance the advantages of highly reputable nodes.

First, we sort the nodes with reputation descending and divide them into several strata layers. Let $N_{\rm sorted} = {\rm sort}(N,r_i\downarrow)$ be the sorted set and L be the number of strata to divide the nodes into. For each stratum layer $L_k=N_{\rm sorted}[\frac{(k-1)N}{L},\frac{kN}{L}), 1\leq k\leq L$, the initial quota per stratum Q_k is calculated by

$$Q_k = \lfloor \frac{\mathcal{K}}{L} \rfloor + \begin{cases} 1, & \text{if } k \le \mathcal{K} \mod L, \\ 0, & \text{otherwise.} \end{cases}$$
 (13)

For each layer, the set of eligible nodes is $\mathcal{E}_k = \{i \in L_k | cd_i^{(t)} = 0\}$, where $cd_i^{(t)} = 0$ indicates the node i is not cooling down. Hence, we can determine the number of selected nodes $m_k = \max(1, \min(Q_k, |\mathcal{E}_k|))$ per layer, using the selection probability sampling performed as follows:

$$\mathbf{P}(i) = \frac{(r_i)^{\gamma}}{\sum_{j \in \mathcal{E}_k} (r_j)^{\gamma}}, \quad j \in \mathcal{E}_k$$
 (14)

where $\gamma \in (0,1]$ is a decay exponent of reputation coefficient. Then, \mathcal{M}_k is obtained by randomly selecting m_k nodes in \mathcal{E}_k based on selection probabilities $\mathbf{P}(i)$ without replacement. Hence, it follows a hypergeometric distribution in probability theory and statistics [54], which can be formally stated as $\mathcal{M}_k \sim \text{Hypergeometric}(m_k, \mathcal{E}_k, \mathbf{P}(i))$.

formally stated as $\mathcal{M}_k \sim \text{Hypergeometric}(m_k, \mathcal{E}_k, \mathbf{P}(i))$. In the case of $\sum_{k=1}^L |\mathcal{M}_k| < \mathcal{K}$, we calculate a remaining quota $\mathcal{K}_{\text{remain}} = \mathcal{K} - \sum_{k=1}^L |\mathcal{M}_k|$ and gather all eligible nodes not yet selected as a global candidate pool, $\mathcal{E}_{\text{global}} = (\bigcup_{k=1}^L \mathcal{E}_k) \setminus (\bigcup_{k=1}^L \mathcal{M}_k)$. Again, using the selection probability sampling on the global eligible pool to calculate the remaining committee $\mathcal{M}_{\text{remain}}$, which is given by,

$$\mathbf{P}'(i) = \frac{(r_i)^{\gamma}}{\sum_{j \in \mathcal{E}_{\text{global}}} (r_j)^{\gamma}}, \quad j \in \mathcal{E}_{\text{global}}$$
 (15)

Hence, $\mathcal{M}_{\text{remain}} \sim \text{Hypergeometric}(\mathcal{K}_{\text{remain}}, \mathcal{E}_{\text{global}}, \mathbf{P}'(i))$, and the total committee is obtained by $\mathcal{M} = (\bigcup_{k=1}^{L} \mathcal{M}_k) \cup \mathcal{M}_{\text{remain}}$.

After the final selection, the system updates each node's statuses, including the selected history and the cooldown time cd_i , which is formulated by

$$cd_i^{(t+1)} = \begin{cases} 3, & \text{if } i \in \mathcal{M}, \\ \max(0, cd_i^{(t)} - 1), & \text{otherwise.} \end{cases}$$
 (16)

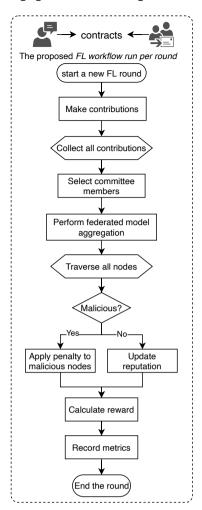


Figure 5: Flowchart of FL system operations per round.

where we specify that if i is selected as a committee, the system will update the cooldown as three at the current round and decrease the cooling time of the unselected nodes by one cooldown per round.

Reputation update model. We employ a dynamic reputation update model to encourage long-term high-quality and stable contributions for all participants through a bonus for the quality and stability of contributions while discouraging malicious behaviours by penalties on reputation. To achieve that, we consider a dynamic decay factor $\delta \in [0,1]$ to incentivize long-term participation by obtaining a reputation compensation from the last round, which is given by

$$\delta = \delta_b + \lambda_p \cdot (1 - \frac{1}{1 + \text{participation}_i / 100}), \tag{17}$$

where δ_b is a base decay factor. λ_p denotes a decay compensation parameter and participation_i is the i^{th} participant's historical participation times. As such, the more participation_i, the higher the reputation compensation $\delta \cdot r_i^{(t-1)}$ for round t.

Besides, a good quality of contribution is good for an increased reputation. Let X_c denote a unit bonus from the

contribution quality to reputation updates, the amount of increased reputation resulting from good contribution is given by $X_c \cdot q_i^{(t)}$, where $q_i^{(t)}$ is quantified by Eq. 4. Additionally, we define $\lambda_{\mathrm{stab}} = 1 - \mathrm{std} \big([C_i^{t-\tau}, \cdots, C_i^t] \big) / \tau$ as a stability parameter during the last τ rounds to avoid malicious nodes with random attacks to obtain rewards. Hence, the new reputation at round t can be defined by

$$r_i^{(t)} = \delta r_i^{(t-1)} + q_i^{(t)} X_c + \lambda_{\text{stab}} X_s,$$
 (18)

where X_s denote unit bonus from the stability to reputation updates and $r_i^{(t)} \in [0, r_{\max}]$. We employ a dynamic maximum capability r_{\max} per round to limit the upper bound of reputation updates.

Malicious detection and penalty. To safeguard the FL process, we consider malicious detection and penalty through a set of conditions. It consists of: 1) persistent low contributions (condition 1), 2) abnormal fluctuations (condition 2), and 3) sudden behavioural changes (condition 3), which serve as indicators for detecting malicious nodes. For ease of analysis, we assume that any participants' behaviours meet the condition set (condition 1 AND condition 2) OR condition 3, the system will detect them as a set of malicious nodes $N_{\rm malicious}$ and perform a penalty to them. The penalty function is defined by,

penalty = min(
$$\lambda_r r_j + \lambda_s S_j, \frac{r_j}{2}$$
), (19)

where λ_r, λ_s denote the factors of applying penalty to reputation updates and stake, respectively. The upper bound $\frac{r_j}{2}$ is intended to avoid excessive penalties on the reputation. Hence, the updated reputation for node j at round t is given by $r_j^{(t)} = r_j^{(t-1)}$ – penalty, $\forall j \in N_{\text{malicious}}$.

Reward calculation. Let *B* denote the base reward of the reward pool. The reward calculation considers a participant's stake, historical contribution performance, and committee bonus. The stake and contribution proportions relative to the overall records influence the rewards participants receive from the reward pool.

We introduce two dynamic weights, α and β , based on reputation to balance the influence of stake and historical contributions in reward calculation, where $\beta=1-\alpha$. The weight α is defined as $\alpha=\sigma(\frac{\bar{r}-r_i^0}{f_{\rm scale}})\cdot\lambda_{\rm stake}$, where $\lambda_{\rm stake}$ is the global stake weight, and reputation r determines the adjustment.

To prevent monopolization, we introduce an effective stake limit for large nodes, ensuring no single node dominates. The effective stake is given by $S_i^{\text{eff}} = \min(S_i, 3 \cdot \overline{S})$, where nodes holding more than three times the average stake \overline{S} are capped.

The historical contribution of participant i over the last τ rounds is computed as $C_i^{\text{hist}} = \sum_{t=0}^{\tau} C_i^{(t-\tau)} \cdot \zeta^t$, where $\zeta^t(\zeta \in (0,1))$ is an exponential decay factor that prioritizes more recent contributions. The total historical contribution across all participants is given by $C_{\text{total}} = \sum_{i \in N} \sum_{t=0}^{\tau} C_i^{(\tau-t)} \cdot \zeta^t$.

Any participant selected as a committee member will receive a bonus for their contributions to model validation and aggregation, which is formulated by,

$$R_{\text{cmm}} = \begin{cases} B_{\text{cmm}} \cdot J_{\text{c}}(r), & \text{if } i \in \mathcal{M}^{(t)}, \\ 0, & \text{otherwise.} \end{cases}$$
 (20)

Here, $J_{\rm c}(r)=\frac{(\sum_{i=1}^{\cal E}r_i)^2}{{\cal K}\cdot\sum_{i=1}^{\cal E}r_i^2+\epsilon}\cdot\sigma(\frac{\bar r}{10}),\,B_{\rm cmm}$ represents a base bonus, and $J_c(r)$ is an improved Jain's fairness index [55] calculated from the reputation scores of the committee. A small constant ϵ is introduced to prevent division by zero errors. The sigmoid function $\sigma(\frac{\bar r}{10})$ ensures that the average reputation score is mapped to the range (0,1). When the mean reputation $\bar r$ is low, the overall score decreases accordingly. If the fairness index $J_c(r)$ is low, the committee reward $R_{\rm cmm}$ is also reduced, which promotes fair and balanced incentives.

Furthermore, we employ J(r) based on reputation, as a fairness indicator to ensure equitable reward distribution among all participants $i \in N$ when earning rewards from their stake and historical contributions. Consequently, the total reward allocated to each participant at round t is formally defined as

$$R_i^{(t)} = (\alpha \cdot B \frac{S_i^{\text{eff}}}{\sum S_i} + \beta \cdot B \frac{C_i^{\text{hist}}}{C_{\text{total}}}) \cdot J(r) + R_{\text{cmm}}$$
 (21)

Note that if a participant has no contribution history (i.e., $C_i^{\text{hist}} = \emptyset$ or $C_i^{(t)} = 0$), the current reward is set to zero, $R_i^{(t)} = 0$. Therefore, it enables adjustable weights and zero-contribution handling to calculate the participant's reward. The entire procedure can be implemented as a reputation-based consensus in a smart contract for reliable FL processes, whose pseudocode is presented in Algorithm 1.

Contract optimality. With the above formulated optimization problem and detailed reward model, we can now study the optimality of the contract problem.

Let $\phi_i^* \triangleq (C_i^*, S_i^*, R_i^*) \in \Phi^*, \forall i \in N$ in the new ordering be the optimal contract that the task publisher can derive from maximizing its total profit $\Pi(\cdot)$ and each participant performs optimal behaviours to maximize its utility. These contract items are found through solving the optimization problem as presented in Eq. 11 under the IC and IR constraints. To derive the optimal contract for this problem, there may be several solutions.

For ease of analysis, we simplify the derivation process. By assuming that: 1) all participants comply with the rules viz $\mathbb{I}_{\text{compl}} = 1$ (no violations), 2) consider a single participant scenario that can ignore the IC constraint, and 3) the cost of contribution $c(C) = \frac{1}{2}\gamma_c C^2$ where $\gamma_c > 0$ is a cost parameter and $C \in [0, C_{\text{max}}]$, the relaxation version of the optimal problem can be formulated as

$$\max_{C,S,R} \Pi = V(C) - R, \quad \text{s.t. } R - \frac{1}{2} \gamma_c C^2 \ge 0. \tag{22}$$

If we ignore the IR constraint and directly maximize profit Π , we need to define the relationship between the reward R

```
Algorithm 1: Reputation-based reliable FL contract.
```

```
Output: Well-trained model w, updated reputation
              r, reward R.
   Input: FL workflow F, consisting of: an initial
            model w, a set of N participants indexed by
            i, an aggregation strategy A, and system
            configuration.
 1 Initialize the FL workflow F;
   /* Define the data structures:
 2 struct Node { id, stake, reputation, totalReward,
     violations, participation, cooldown, ...,
     identityVerified }
3 struct SystemConfig { baseReward, committeeSize,
     stakeWeight, ..., maliciousPercent }
4 contract FLSystem { // Define the main contract
      Initialize the nodes:
      Initialize the SystemConfig;
 7
   /* Core logic of the main contract:
                                                                */
      function runRound() {
      for each round t do
           Collect C^{(t)} = \{C_i^{(t)} \mid \forall i \in N\} within T_{\text{max}};
10
           \mathcal{M}^{(t)} \leftarrow \text{selectCommittee}(r^{(t)}, \mathcal{K}, L);
11
           committee performs aggregation;
12
           /* Malicious detection and penalty:
13
           if \forall i \in N_{malicious} = \emptyset then
Update reputation: r_i^{(t)} \leftarrow \text{Eq. } 18;
14
15
           else
16
                Apply penalty: penalty_i \leftarrow \mathbf{Eq. 19};
17
             r_i^{(t)} \leftarrow r_i^{(t-1)} - penalty_i, \forall i \in N_{\text{malicious}};
violation + +;
18
           Calculate rewards: R_i^{(t)} \leftarrow \mathbf{Eq. 21}, \forall i \in N;
19
   /* Stop when an end condition triggers
       return final model weights w, r, R.
21
```

23 Each FL client executes:

for each client $i \in N$ at round t in parallel do
Update model and calculate contributions: $\mathbf{w}_{i}^{(t)}, C_{i}^{(t)} \leftarrow \mathbf{Eqs. 1, 3};$

26 Committee executes:

22

Aggregate model weights: $\mathbf{w}^{(t+1)} \leftarrow \mathbf{Eq. 2}$;

(Eq. 21) and contribution value V(C) (Eq. 4). We further assume 1) there are no monopoly stakes $S \leq 3\overline{S}$, then $S^{\text{eff}} = S$, 2) $r_i = \overline{r}$, then $\alpha = \lambda_{\text{stake}}$, and 3) both C^{hist} and C_{total} are constants. At this point, maximizing Π is equivalent to differentiating the objective function with respect to the first term C and solving the first-order condition, which can be rearranged as

$$\frac{\partial \Pi}{\partial C} = V'(C) - \frac{(1 - \lambda_{\text{stake}})BJ(r)}{C_{\text{total}}} \cdot \frac{\partial C^{\text{hist}}}{\partial C} = 0, \quad (23)$$

where we assume $C^{\text{hist}} = C \cdot \sum_{t=0}^{\tau} \zeta^{t}$, which leads to

$$V'(C) = \frac{(1 - \lambda_{\text{stake}})BJ(r)}{C_{\text{total}}} \cdot \frac{1 - \zeta^{\tau + 1}}{1 - \zeta}.$$
 (24)

Correspondingly, we define $k = \frac{1}{C_{\text{max}}}$ to extend the V(C) as the full version and calculate its derivative V'(C), which is calculated by,

$$V'(C) = \frac{\partial(\frac{X_c}{\tau} \cdot \frac{1}{1 + e^{-kC}})}{\partial C} = \frac{X_c \cdot ke^{-kC}}{\tau(1 + e^{-kC})^2},$$
 (25)

Thus, the optimal contribution C^* made by participant can be derived from $V'(C^*)$ (Eq. 24) = $V'(C^*)$ (Eq. 25), which is given by

$$C^* = \frac{1}{k} \ln(\frac{X_c \cdot k\tau(1-\zeta)}{(1-\lambda_{\text{stake}})BJ(r)(1-\zeta^{\tau+1})C_{\text{total}}} - 1).$$
(26)

Since $\frac{1}{k} = C_{\text{max}}$, then we have $C^* = C_{\text{max}} \cdot y$ where $y = \ln(x-1)$ is a logarithmically increasing function for $x = \frac{X_c \cdot k\tau(1-\zeta)}{(1-\lambda_{\text{stake}})BJ(r)(1-\zeta^{\tau+1})C_{\text{total}}}$ that asymptotically converges, theoretically, the optimal contribution would approach to positive infinity. However, the maximum practical constraints $C \in [C_{\min} = 0, C_{\max}]$, as a result, the actual optimal solution is $C^* = C_{\max}$. In other words, $C_i^* \propto f_{\text{local}}(D_i, \mathbf{w}) \cdot g_{\text{behaviour}}(h_i)$, adhering to optimal local training while demonstrating good behaviour, becomes the dominant strategy to maximize its utility for any participant. Meanwhile, $\ln(\frac{X_c \cdot k\tau(1-\zeta)}{(1-\lambda_{\text{stake}})BJ(r)(1-\zeta^{\tau+1})C_{\text{total}}} - 1) = 1$ can serve as a guideline for tuning the parameters (e.g., λ_{stake} , X_c and B if given fixed others) to achieve the optimal solution. By binding IR constraint, the optimal reward is $R^* = c(C^*)$. As all participants stake the same amount S, which leads to

$$S^* = \frac{\lambda_{\text{stake}} BJ(r)}{n \cdot (R^* - R_{\text{cmm}} - \frac{(1 - \lambda_{\text{stake}}) BJ(r)C^{\text{hist}}}{C_{\text{total}}})},$$
(27)

$$R^* = c(C^*). \tag{28}$$

If we consider the IR constraint, the optimization problem can be reorganized as a Lagrangian function with a Lagrange multiplier and then repeat the derivation process for the three terms C, S, R, respectively. With the help of math tools (e.g., SciPy methods in Python) to solve for the problem, we can obtain the optimal C^*, S^*, R^* by adjusting the parameters as needed by numerical analysis. Suppose the distribution of the participants' violations (concerning Byzantine failures) is not predictable and the stakes are unbalanced. In those cases, we can consider using advanced approaches, such as Bayesian priors [56] or reinforcement learning optimization [57], to meet the requirements.

5. Experiments and Evaluation

This section illustrates simulation setup, demonstrates the feasibility of the proposed incentive mechanisms, and evaluates the outcomes for validation and guidance of (optimal) smart contract design.

5.1. Simulation Setup

In the simulation, we conducted extensive simulation experiments on a local computer equipped with an Apple M1 chip with eight (four performance and four efficiency) cores, 16GB memory, and a 500 GB Macintosh HD disk.

For the evaluation of solutions, we utilize a normal distribution \mathcal{N} with random fluctuation \mathcal{F} for the ease of training process to present the normal behaviour pattern for participants acting normally, $C_{\text{normal}} = \max(0, \mathcal{N}(\mu = 0))$ $(7, \sigma^2 = 1) \cdot \mathcal{F})$, where μ and σ denote the mean and standard deviation of the set of contributions collected from the normal (or honest) FL participants. For malicious behaviour patterns, we simulate malicious patterns with three different attacks: 1) false high contribution attack, where malicious nodes disguise themselves as high-value nodes through abnormally high contributions to avoid detection based on low contributions; 2) zero contribution attack where malicious nodes directly destroy the federated model aggregation and reduce model performance, and 3) random attack which is given by a random choice of 60% false high contribution attack and 40% zero contribution attack.

We introduce a parameter $\eta_{\rm switch}$ that allows switching between different patterns to simulate various malicious behaviours during the training process. The parameter defines the longest time window threshold at which the system transitions from base to progressive malicious phases — ranging from high-contribution and zero-contribution attacks to randomly hybrid attacks. This enables the implementation of malicious pattern-switch logic for progressive attack testing, periodic behavioural perturbations, and defense mechanism verification. By periodically switching attack strategies (e.g., every 30 rounds), we simulate the adaptability of malicious participants. Additional details are available in the source code.

Additionally, we set different malicious percentages as an adjustable parameter to distinguish between malicious and honest nodes among the total participants. An attack detection function is implemented to identify such malicious behaviours, which allows us to test and evaluate the effects of penalties, reputation updates, and rewards on the system. A summary of the other parameters used in the simulation is presented in Table 2.

To measure the fairness in reward allocation based on actual contributions, we consider two metrics for evaluation. Jain's fairness index [55] is a quantitative measure of fairness and discrimination for resource allocation in shared systems. We have used it to maintain the fairness of committee selection on reputation presented in Eq. 20, whereas it can also be employed to measure the fairness of reward allocation J(R) by replacing the input as R. Additionally, we consider an opposite metric — the Gini coefficient [58], as a measurement of reward inequality, which is formulated by

$$J(R) = \frac{(\sum_{i=1}^{n} R_i)^2}{n\sum_{i=1}^{n} R_i^2 + \epsilon} \cdot \sigma(\frac{\overline{R}}{10}), \tag{29}$$

Table 2 Parameter setting in the simulation.

Parameter	Description	Setting
S_i	initial stake	100
$\frac{S_i}{r_i^0}$	initial reputation score	100
C_{\min}	minimal contribution score	0
$C_{ m max}$	maximum contribution score	10
γ	reputation decay coefficient	0.5
$r_{\text{max}}^{t \le 5}, r_{\text{max}}^{t > 5}$	maximum capabilities of reputation updates	300, 500
ϵ	a small constant	10^{-8}
$cd(\cdot)$	cooldown period	3
Ĺ	number of strata	3
B_{cmm}	base bonus to committee member	40
δ_b	base decay factor	0.88
λ_p^{ν}	decay compensation parameter	0.07
τ	recent historical rounds	5
τ_{stab}	default stability for new participants	0.8
X_c	base bonus of contribution	50
X,	base bonus of stability	30
$\lambda_r^{'}$	penalty factor to reputation	0.3
λ_s	penalty factor to stake	0.1
ζ	historical decay factor	0.9
B	base reward value in the pool	1200
n	the number of participants	100
λ_{stake}	stake weight	0.4
K	committee size	5
m	malicious percent	15%
η_{switch}	the first time window of observing a switch of malicious patterns	5
Round	the total number of the FL rounds	90

$$G(R) = (n+1-2\frac{\sum_{i=1}^{n} \sum_{j=1}^{i} R_j}{\sum_{j=1}^{n} R_j})/n$$
 (30)

where the input set R is performed in ascending order. The source code is available online at the GitHub repository⁷.

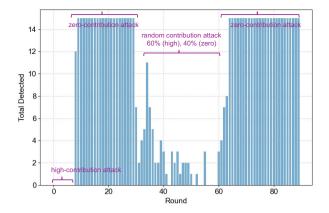
5.2. Simulation Results

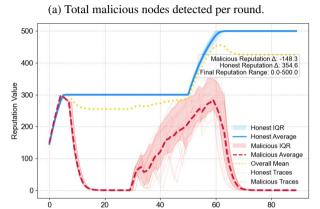
This section presents an overview of the simulation results under the parameter setting presented in Table 2. Figure 6 showcases the dynamics of malicious node detection, reputation updates, and reward distribution per round.

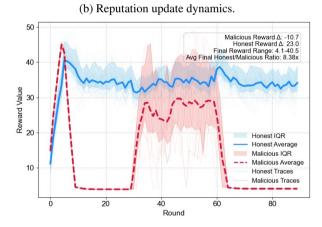
Malicious detection. Sub Figure 6a illustrates the malicious detection performance per round. The detection method follows the condition set (condition 1 AND condition 2) OR condition 3, based on a malicious percentage of 15%. The malicious pattern switch parameter η_{switch} plays a crucial role in shaping the behaviour records of malicious participants, thereby influencing detection performance.

Notably, false high-contribution malicious nodes were not detected during the first five rounds, as the current detection conditions do not account for high-contribution attacks, allowing such behaviours to appear normal. When the round $t \in [5,30)$, the malicious pattern shifts to zero contributions. By the eighth round, the system successfully identified 12 out of 15 malicious nodes. When the round $t \in [30,60)$, the malicious pattern transitions to random attacks, with a 60% probability of high-contribution attacks and a 40% probability of zero-contribution attacks. In the final 30 rounds, malicious behaviours revert to zero-contribution attacks. The detection method proves to be more effective against zero-contribution attacks but remains ineffective against false high-contribution attacks due to current detection limitations.

Reputation updates. Sub Figure 6b illustrates the reputation update dynamics per round, distinguishing between







(c) Reward distribution.

Figure 6: Overview of malicious detection, reputation updates, and reward distribution per round.

honest and malicious nodes among 85 honest participants and 15 malicious ones. Initially, all nodes start with a reputation score of 100, which increases as new contributions are collected during the first few rounds. This growth continues until it reaches a plateau around a reputation score of 300. Notably, there is no difference in reputation growth between honest and malicious nodes until the eighth round.

At the eighth round, some malicious nodes experience a sharp drop in reputation due to detected malicious behaviours, as shown in Figure 6a. Since most malicious nodes

⁷https://github.com/yuandou168/reliableFLOps

were identified during this period, it caused their reputation scores to decline significantly, eventually approaching zero. During the random attack phase, the system fails to detect all malicious activities, allowing some malicious nodes to temporarily regain their reputation through high-contribution attacks. This results in fluctuations in their reputation updates, as not all malicious nodes receive penalties in this phase. However, once all 15 malicious nodes are successfully detected, the fluctuations cease, and their reputation scores rapidly decline, dropping by an average of 148.3 points.

Meanwhile, the reputation of the 85 honest nodes steadily increased as expected in the first five rounds by 300. Then they gradually reach the upper limit of 500. The average reputation increase for honest nodes is approximately 354.6, significantly higher than that of the malicious nodes.

Reward distribution dynamics. Sub Figure 6c illustrates the reward distribution dynamics among 85 honest and 15 malicious nodes out of 100 participants. Because malicious nodes mimic high-contribution participants, they accumulate higher rewards than honest nodes. Hence, they exhibit a higher average reward value and a faster growth rate in the early rounds. However, as the system detects zero-contribution attacks, their rewards drop sharply to zero. Although some malicious nodes temporarily regain their reputation by engaging in high-contribution attacks in the period of random attacks, as more malicious nodes are progressively detected, their corresponding rewards decline.

Although all malicious nodes ultimately reach a very low reward level over the 90 rounds, their total rewards are not zero. Overall, the final reward values range between 4.1 and 40.5 at round 90, satisfying the IR constraint. However, participants acting maliciously or continuing such behaviour under this reward incentive mechanism cannot achieve optimal utility. Specifically, the average reward for malicious nodes dropped by 10.7, while the average reward for honest nodes increased by 23.0. The total reward accumulated by the 85 honest nodes is 8.38 times greater than that of the 15 malicious nodes in the simulation system.

Consequently, this incentive mechanism encourages participants to adopt a more rational strategy — avoiding malicious behaviour to maximize their rewards. Therefore, it preserves the trend of rewarding aligned with the incentives while reducing the risk of over-penalty and over-reward.

Fairness. Figure 7 compares the fairness of reward distribution across 100 participant nodes under five different malicious percentage settings. Similar to the reward dynamics observed in Figure 6c, the fairness initially improves, as indicated by an increase in Jain's fairness index and a decrease in the Gini coefficient (see Eq. 29), that suggests a trend toward a more equitable reward distribution. However, due to the influence of malicious nodes, the reward allocation is not always fair.

For instance, given a malicious percentage 15%, we observed that malicious detection significantly influences reward allocation's fairness. When most or all malicious nodes are successfully identified, i.e., during rounds between 8 and 30, as well as 60 to 90, the fairness index decreases

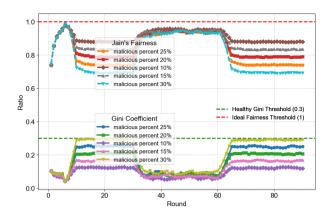


Figure 7: Comparisons of total fairness metric dynamics across the last 90 rounds.

while the Gini coefficient increases. Conversely, fairness remains relatively stable when the system detects only a few malicious nodes. It reflects an imbalanced reward allocation influenced by the performance of malicious detection.

As the percentage of malicious nodes in the system increases, e.g., from 10% to 30%, the reward distribution becomes more inequitable between honest and malicious nodes. However, such inequity remains at a healthy threshold (<0.3). Such a change is positive because it still preserves fairness among honest participants while preventing malicious participants from obtaining unfair rewards.

The optimal parameter space. In the simulation, we employ the controlled variable method and an optimization solver 'SLSQP', which is suitable for constrained optimization, to analyze the applicable approximations of the optimal contract items $(C_i^*, S_i^*, R_i^*) \in \Phi^*$. The numerical analysis validates the optimal contribution $C^* \to C_{\max}$ for all participants. It also verifies that the IR-constrained satisfaction rate is 100% while minimum utility is more than zero. With this numerical analysis, users can adjust their contract terms for different requirements with optimal parameter settings. More technical details have been presented in the source code.

These simulation results suggest that the proposed incentive mechanisms are feasible in an FL system with honest and malicious participants. By coding these mechanisms into smart contracts and bounding blockchain, the system can effectively constrain and guide participant behaviour, ensuring that honest participants can receive fair compensation while discouraging malicious activities. The numerical analysis can help customize (approximately) optimal contract items written in smart contracts to adopt diverse FL scenarios.

6. Summary

This study demonstrated how FL can be structured as a workflow using open workflow standards and executed on remote infrastructure to address automation challenges. To bridge the gap between traditional workflow-based automation and decentralized collaboration, we propose a novel

strategy that builds upon our prior work to enhance reliability in FL management over decentralized infrastructures. We leverage contract theory with a multidimensional scheme to tackle the incentivizing collaboration problem by constructing a contribution model, implementing fair committee selection, dynamically updating reputations, calculating rewards, and defining corresponding profit and utility functions. Additionally, we explore the optimality of contracts to guide the design and implementation of smart contracts that can be deployed on blockchain networks to assure training quality. We conduct extensive simulation experiments to validate the proposed approach. The results demonstrate that our incentive mechanisms are effective, ensuring fair reward allocation despite malicious attacks. Future improvements include encoding the optimal contract items into smart contracts for real-world system performance evaluation and prototyping an integrated on-chain and offchain system demonstration.

Acknowledgments

We thank Mr. Anandan Krishnasamy for running FL workflow experiments over the hybrid cloud environment and validating the FL training results. This research was made possible through partial funding from several European Union projects: CLARIFY (860627), ENVRI-Hub Next (101131141), EVERSE (101129744), BlueCloud-2026 (101094227), OSCARS (101129751), LifeWatch ERIC, BioDT (101057437, through LifeWatch ERIC), and Dutch NWO LTER-LIFE project.

References

- [1] Rodolfo Stoffel Antunes, Cristiano André da Costa, Arne Küderle, Imrana Abdullahi Yari, and Björn Eskofier. Federated learning for healthcare: Systematic review and architecture proposal. ACM Transactions on Intelligent Systems and Technology (TIST), 13(4):1– 23, 2022.
- [2] Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. Federated learning for open banking. In *Federated learning: privacy and incentive*, pages 240–254. Springer, 2020.
- [3] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.
- [4] Biwei Yan, Hongliang Zhang, Minghui Xu, Dongxiao Yu, and Xiuzhen Cheng. Fedrfq: Prototype-based federated learning with reduced redundancy, minimal failure, and enhanced quality. *IEEE Transactions on Computers*, 2024.
- [5] Pian Qi, Diletta Chiaro, Antonella Guzzo, Michele Ianni, Giancarlo Fortino, and Francesco Piccialli. Model aggregation techniques in federated learning: A comprehensive survey. Future Generation Computer Systems, 2023.
- [6] Jiajun Wu, Fan Dong, Henry Leung, Zhuangdi Zhu, Jiayu Zhou, and Steve Drew. Topology-aware federated learning in edge computing: A comprehensive survey. ACM Computing Surveys, 56(10):1–41, 2024.
- [7] Bo Xu, Wenchao Xia, Wanli Wen, Pei Liu, Haitao Zhao, and Hongbo Zhu. Adaptive hierarchical federated learning over wireless networks. IEEE Transactions on Vehicular Technology, 71(2):2070–2083, 2021.
- [8] Lumin Liu, Jun Zhang, Shenghui Song, and Khaled B Letaief. Hierarchical federated learning with quantization: Convergence analysis and system design. *IEEE Transactions on Wireless Communications*, 22(1):2–18, 2022.

- [9] Stefano Savazzi, Monica Nicoli, and Vittorio Rampa. Federated learning with cooperating devices: A consensus approach for massive iot networks. *IEEE Internet of Things Journal*, 7(5):4641–4654, 2020.
- [10] Yuan Liu, Zhengpeng Ai, Shuai Sun, Shuangfeng Zhang, Zelei Liu, and Han Yu. Fedcoin: A peer-to-peer payment system for federated learning. In *Federated learning: privacy and incentive*, pages 125– 138. Springer, 2020.
- [11] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. NPJ digital medicine, 3(1):1–7, 2020.
- [12] Jianchun Liu, Hongli Xu, Lun Wang, Yang Xu, Chen Qian, Jinyang Huang, and He Huang. Adaptive asynchronous federated learning in resource-constrained edge computing. *IEEE Transactions on Mobile Computing*, 22(2):674–690, 2021.
- [13] Minghui Xu, Zongrui Zou, Ye Cheng, Qin Hu, Dongxiao Yu, and Xiuzhen Cheng. Spdl: A blockchain-enabled secure and privacypreserving decentralized learning system. *IEEE Transactions on Computers*, 72(2):548–558, 2022.
- [14] Yufeng Zhan, Jie Zhang, Zicong Hong, Leijie Wu, Peng Li, and Song Guo. A survey of incentive mechanism design for federated learning. *IEEE Transactions on Emerging Topics in Computing*, 10(2):1035– 1044, 2021.
- [15] Xuezhen Tu, Kun Zhu, Nguyen Cong Luong, Dusit Niyato, Yang Zhang, and Juan Li. Incentive mechanisms for federated learning: From economic and game theoretic perspective. *IEEE transactions on cognitive communications and networking*, 8(3):1566–1593, 2022.
- [16] Han Xu, Priyadarsi Nanda, and Jie Liang. Reciprocal federated learning framework: Balancing incentives for model and data owners. Future Generation Computer Systems, 161:146–161, 2024.
- [17] Yann Fraboni, Richard Vidal, and Marco Lorenzi. Free-rider attacks on model aggregation in federated learning. In *International Con*ference on Artificial Intelligence and Statistics, pages 1846–1854. PMLR, 2021.
- [18] Leon Witt, Mathis Heyer, Kentaroh Toyoda, Wojciech Samek, and Dan Li. Decentral and incentivized federated learning frameworks: A systematic literature review. *IEEE Internet of Things Journal*, 10(4): 3642–3663, 2022.
- [19] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.
- [20] Semo Yang, Jihwan Moon, Jinsoo Kim, Kwangkee Lee, and Kangyoon Lee. Flscalize: Federated learning lifecycle management platform. *IEEE Access*, 11:47212–47222, 2023.
- [21] Iacopo Colonnelli, Bruno Casella, Gianluca Mittone, Yasir Arfat, Barbara Cantalupo, Roberto Esposito, Alberto Riccardo Martinelli, Doriana Medić, and Marco Aldinucci. Federated learning meets hpc and cloud. In ML4Astro International Conference, pages 193–199. Springer, 2022.
- [22] Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer, et al. Common workflow language, v1.0, 2016.
- [23] Chronis Kontomaris, Yuandou Wang, and Zhiming Zhao. Cwl-flops: A novel method for federated learning operations at scale. In 2023 IEEE 19th International Conference on e-Science (e-Science), pages 1–2. IEEE, 2023.
- [24] Harshit Daga, Jaemin Shin, Dhruv Garg, Ada Gavrilovska, Myungjin Lee, and Ramana Rao Kompella. Flame: Simplifying topology extension in federated learning. In *Proceedings of the 2023 ACM Symposium on Cloud Computing*, pages 341–357, 2023.
- [25] Wil MP van Der Aalst, Arthur HM Ter Hofstede, Bartek Kiepuszewski, and Alistair P Barros. Workflow patterns. *Distributed and parallel databases*, 14:5–51, 2003.
- [26] Sin Kit Lo, Qinghua Lu, Liming Zhu, Hye-Young Paik, Xiwei Xu, and Chen Wang. Architectural patterns for the design of federated learning systems. *Journal of Systems and Software*, 191:111357, 2022.

- [27] Qi Cheng and Guodong Long. Federated learning operations (flops): Challenges, lifecycle and approaches. In 2022 International Conference on Technologies and Applications of Artificial Intelligence (TAAI), pages 12–17. IEEE, 2022.
- [28] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. Devops. *IEEE software*, 33(3):94–100, 2016.
- [29] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (mlops): Overview, definition, and architecture. *IEEE access*, 11:31866–31879, 2023.
- [30] Rongfei Zeng, Chao Zeng, Xingwei Wang, Bo Li, and Xiaowen Chu. A comprehensive survey of incentive mechanism for federated learning. arXiv preprint arXiv:2106.15406, 2021.
- [31] Nika Haghtalab, Mingda Qiao, and Kunhe Yang. Platforms for efficient and incentive-aware collaboration. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2607–2628. SIAM, 2025.
- [32] Zhilin Wang, Qin Hu, Ruinian Li, Minghui Xu, and Zehui Xiong. Incentive mechanism design for joint resource allocation in blockchainbased federated learning. *IEEE Transactions on Parallel and Dis*tributed Systems, 34(5):1536–1547, 2023.
- [33] Liang Gao, Li Li, Yingwen Chen, ChengZhong Xu, and Ming Xu. Fgfl: A blockchain-based fair incentive governor for federated learning. *Journal of Parallel and Distributed Computing*, 163:283–299, 2022.
- [34] Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal*, 6(6):10700–10714, 2019.
- [35] Tailin Zhou, Zehong Lin, Jun Zhang, and Danny HK Tsang. Understanding and improving model averaging in federated learning on heterogeneous data. *IEEE Transactions on Mobile Computing*, 2024.
- [36] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and* statistics, pages 1273–1282. PMLR, 2017.
- [37] Zhiming Zhao, Spiros Koulouzis, Riccardo Bianchi, Siamak Farshidi, Zeshun Shi, Ruyue Xin, Yuandou Wang, Na Li, Yifang Shi, Joris Timmermans, et al. Notebook-as-a-vre (naavre): From private notebooks to a collaborative cloud virtual research environment. Software: Practice and Experience, 52(9):1947–1966, 2022.
- [38] Laëtitia Launet, Yuandou Wang, Adrián Colomer, Jorge Igual, Cristian Pulgarín-Ospina, Spiros Koulouzis, Riccardo Bianchi, Andrés Mosquera-Zamudio, Carlos Monteagudo, Valery Naranjo, et al. Federating medical deep learning models from private jupyter notebooks to distributed institutions. Applied Sciences, 13(2):919, 2023.
- [39] Yuandou Wang, Spiros Koulouzis, Riccardo Bianchi, Na Li, Yifang Shi, Joris Timmermans, W Daniel Kissling, and Zhiming Zhao. Scaling notebooks as re-configurable cloud workflows. *Data Intelligence*, 4(2):409–425, 2022.
- [40] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. Flower: A friendly federated learning research framework. arXiv preprint arXiv:2007.14390, 2020.
- [41] Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Yi Zhou, Ali Anwar, Shashank Rajamoni, Yuya Ong, Jayaram Radhakrishnan, Ashish Verma, Mathieu Sinn, et al. Ibm federated learning: an enterprise framework white paper vo. 1. arXiv preprint arXiv:2007.10987, 2020.
- [42] G Anthony Reina, Alexey Gruzdev, Patrick Foley, Olga Perepelkina, Mansi Sharma, Igor Davidyuk, Ilya Trushkin, Maksim Radionov, Aleksandr Mokrov, Dmitry Agapov, et al. Openfl: An open-source framework for federated learning. arXiv preprint arXiv:2105.06413, 2021.
- [43] Alexander Ziller, Andrew Trask, Antonio Lopardo, Benjamin Szymkow, Bobby Wagner, Emma Bluemke, Jean-Mickael Nounahon, Jonathan Passerat-Palmbach, Kritika Prakash, Nick Rose, et al. Pysyft: A library for easy federated learning. Federated Learning Systems: Towards Next-Generation AI, pages 111–139, 2021.

- [44] Kasun Indrasiri and Danesh Kuruppu. gRPC: up and running: building cloud native applications with Go and Java for Docker and Kubernetes. O'Reilly Media, 2020.
- [45] Anandan Krishnasamy, Yuandou Wang, and Zhiming Zhao. A collaborative framework for facilitating federated learning among jupyter users. In 2024 IEEE 20th International Conference on e-Science (e-Science), pages 1–2. IEEE, 2024.
- [46] Shaoxiong Ji. A pytorch implementation of federated learning, March 2018. URL https://doi.org/10.5281/zenodo.4321561.
- [47] Carole Goble, Sarah Cohen-Boulakia, Stian Soiland-Reyes, Daniel Garijo, Yolanda Gil, Michael R Crusoe, Kristian Peters, and Daniel Schober. Fair computational workflows. *Data Intelligence*, 2(1-2): 108–121, 2020
- [48] Yuandou Wang, Sheejan Tripathi, Siamak Farshidi, and Zhiming Zhao. D-vre: From a jupyter-enabled private research environment to decentralized collaborative research ecosystem. *Blockchain: Re*search and Applications, page 100244, 2024.
- [49] Huong Nguyen, Hong-Tri Nguyen, Lauri Lovén, and Susanna Pirttikangas. Stake-driven rewards and log-based free rider detection in federated learning. In 2024 21st Annual International Conference on Privacy, Security and Trust (PST), pages 1–10. IEEE, 2024.
- [50] Shuai Wang, Wenwen Ding, Juanjuan Li, Yong Yuan, Liwei Ouyang, and Fei-Yue Wang. Decentralized autonomous organizations: Concept, model, and applications. *IEEE Transactions on Computational Social Systems*. 6(5):870–878, 2019.
- [51] Ningning Ding, Zhixuan Fang, and Jianwei Huang. Optimal contract design for efficient federated learning with multi-dimensional private information. *IEEE Journal on Selected Areas in Communications*, 39 (1):186–200, 2020.
- [52] Zehui Xiong, Wei Yang Bryan Lim, Jiawen Kang, Dusit Niyato, Ping Wang, and Chunyan Miao. Incentive mechanism design for mobile data rewards using multi-dimensional contract. In 2020 IEEE Wireless Communications and Networking Conference (WCNC), pages 1– 6. IEEE, 2020.
- [53] Jerzy Neyman. On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection. In *Breakthroughs in statistics: Methodology and* distribution, pages 123–150. Springer, 1992.
- [54] Anders Hald. The compound hypergeometric distribution and a system of single sampling inspection plans based on prior distributions and costs. *Technometrics*, 2(3):275–340, 1960.
- [55] Rajendra K Jain, Dah-Ming W Chiu, William R Hawe, et al. A quantitative measure of fairness and discrimination. Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA, 21: 1, 1984.
- [56] Peter I Frazier. A tutorial on bayesian optimization. arXiv preprint arXiv:1807.02811, 2018.
- [57] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.
- [58] Robert Dorfman. A formula for the gini coefficient. The review of economics and statistics, pages 146–149, 1979.