Poster Abstract: OrderlessFL: A CRDT-Enabled Permissioned Blockchain for Federated Learning

Pezhman Nasirifard
Technical University of Munich
Munich, Germany
p.nasirifard@tum.de

Ruben Mayer
Technical University of Munich
Munich, Germany
ruben.mayer@tum.de

Hans-Arno Jacobsen
University of Toronto
Toronto, Canada
jacobsen@eecg.toronto.edu

ABSTRACT

Industries produce a large amount of data that can improve *Machine Learning* models. However, due to privacy issues, the data cannot be shared. Several *Federated Learning (FL)* systems have been introduced as private alternatives without considering Byzantine actors. Also, these systems are affected by the gradient staleness problem. Several blockchain-based FL systems are introduced to address Byzantine actors, which rely on *Proof-of-Work-based (PoW)* protocols and suffer from their limitations. We introduce OrderlessFL, a safe permissioned blockchain-based FL system using flCRDT, a CRDT for concurrent ML training and mitigating gradient staleness.

CCS CONCEPTS

• **Computer systems organization** → **Distributed architectures**;
• **Computing methodologies** → **Machine learning**.

KEYWORDS

Conflict-free Replicated Data Type, Federated Learning, Blockchain

ACM Reference Format:
Pezhman Nasirifard, Ruben Mayer, and Hans-Arno Jacobsen. 2022. Poster Abstract: OrderlessFL: A CRDT-Enabled Permissioned Blockchain for Federated Learning. In *23rd International Middleware Conference Demos and Posters (Middleware '22), November 7–11, 2022, Quebec, QC, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3565386.3565487

1 INTRODUCTION

Industries produce a large amount of data that can be used for improving *Machine Learning (ML)* models [11]. However, the raw data can often not be shared among organizations for privacy reasons. Hence, *Federated Learning (FL)* has gained popularity as a solution to privacy-preserving ML [3]. A synchronous FL system consists of several workers and a central *Parameter Server (PS)* [1]. The workers receive a global ML model from PS and train the model based on their local data. The PS aggregates the local updates with the global model.

Despite the enhanced privacy of FL, the central and potentially Byzantine PS can jeopardize the system [8]. Several *Proof-of-Work-based (PoW)* blockchains for FL have been introduced to handle a Byzantine PS [11]. However, PoW has performance limitations [2, 5]. Also, due to the blockchain's storage limitation, they depend on off-chain storage solutions, which could potentially be Byzantine.

Furthermore, the performance of synchronous FL is affected by slow workers, and asynchronous FL is affected by gradient staleness, where the workers train on an outdated global model [10]. Training a model using FL shares similar problems with the concurrent reads and writes in distributed computing. One technique for addressing the problems is *Conflict-free Replicated Data Types (CRDTs)* [9].

In this work, we introduce flCRDT, a novel CRDT enabling the concurrent and asynchronous aggregation of models in FL. flCRDT uses the properties of CRDTs to mitigate the gradient staleness problem. We also introduce OrderlessFL, a permissioned blockchain-based FL system using flCRDT. Our system offers a safe protocol for storing and aggregating models where Byzantine actors cannot tamper with the models.

2 ARCHITECTURE AND FL PROTOCOL

OrderlessFL is a permissioned blockchain consisting of organizations and workers. Organizations act as PS, store and distribute models, and receive and aggregate updates. The workers can communicate with every non-failed organization. The system uses the following *training-aggregation* asynchronous FL protocol, shown in Figure 1. We open-sourced the system's code [1]. We also published an extended paper on the blockchain part of the system [6, 7].

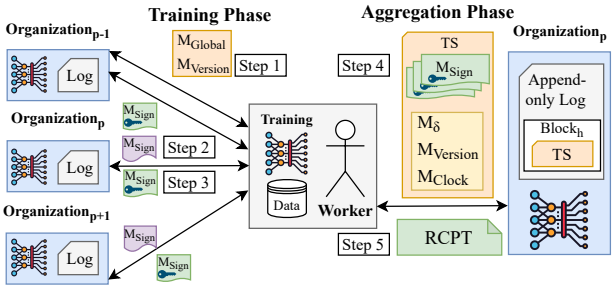

Figure 1: Workflow for training a model.

Training Phase – The worker first contacts any organization to receive $< M_{Global}, M_{Version} >$, the weights of the global model and the model's current version (Step 1 in Figure 1). The model architecture and learning algorithm are global system configuration settings. The worker initializes a model with M_{Global} and trains the model using its local data. Afterward, $M_\delta = M_{Local} - M_{Global}$, the weight difference between the locally trained and global models, is calculated. The worker also keeps track of a logical clock M_{Clock}, incrementing it with every update. Then, the worker creates a model signature

[1] https://github.com/orderlesschain/orderlessfl

Middleware '22, November 7–11, 2022, Quebec, QC, Canada

ACM ISBN 978-1-4503-9931-9/22/11…$15.00
https://doi.org/10.1145/3565386.3565487

$M_{Sign} = Hash(< M_{\delta}, M_{Version}, M_{Clock} >)$ and sends it to the organizations (Step 2). The organization signs M_{Sign} with its private key based on public-key cryptography and sends the signed response, also known as an endorsement, to the worker (Step 3).

Aggregation Phase – The worker waits to receive the endorsements from organizations and verifies the signatures' validity. If every endorsement is valid, a transaction *TS* that contains the endorsements and model update is created and sent to the organizations (Step 4). The organization appends *TS* to its append-only hash-chain log by creating a block $Block_h = < Hash(TS, Block_{h-1}) >$, which also contains the hash of the previous block, and appends the block to the log. The organization verifies whether every endorsement in *TS* is valid. This ensures that every organization has received identical data from the worker and prevents Byzantine behavior. A signed receipt containing the block's hash for valid transactions is sent to the worker (Step 5). A signed rejection is sent otherwise. As the receipt contains the block's hash value which is also dependent on the previous blocks, any Byzantine modification of the organization invalidates the receipts of *TS* and other transactions. Finally, the organization aggregates M_{δ} of the valid transaction into the global model, explained below.

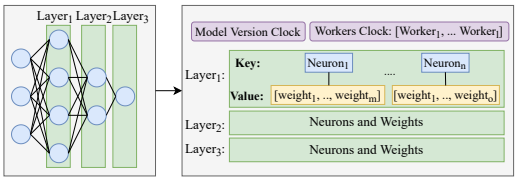

Figure 2: Modeling a DNN with FLCRDT.

Algorithm 1: Aggregating the updates.

```
1  AggregateToGlobalModel (M_Global, M_δ, M_Version, M_Clock, WorkerID)
2      if M_Clock − M_Global.WorkerClocks[WorkerID] == 1 then
3          M_Global.Version += 1
4          M_Global.WorkerClocks[WorkerID] += 1
5          StalenessPenalty = (M_Global.Version − M_Version)^−1
6          UpdateRate = StalenessPenalty ∗ M_Global.WorkerClocks.Length^−1
7          foreach layer in MW_δ.Layers do
8              M_Global.Layers[layer] += M_δ.Layers[layer] ∗ UpdateRate
9      else
10         M_Global.EnqueueUpdate(< M_δ, M_Version, M_Clock, WorkerID >)
11     M_Global.ProcessQueuedUpdates()
```

3 FLCRDT: THE FEDERATED LEARNING CRDT

We introduce FLCRDT, a nested CRDT for modeling a wide range of ML models. In this work, we only discuss modeling a *Deep Neural Network (DNN)*, as shown in Figure 2. A DNN model consists of several layers, with every layer consisting of several neurons with their weights. As FLCRDT is a nested data structure, the layers and neurons can be modeled as nested data structures in an instance of FLCRDT. The root of FLCRDT is a map consisting of key-value pairs, where the key is a unique identifier of the layer, and the value consists of nested maps containing other layers or the neurons' weights. FLCRDT also contains two logical clocks: a model clock and a workers' vector clock. The model clock stores the model version and increments it with every update. The workers' vector clock stores the clocks of workers.

Algorithm 1 displays our approach used by organizations to aggregate M_{δ} with the global model. The organization proceeds to aggregate M_{δ} if it has observed all previous updates sent by the worker (Line 2). Otherwise, the update is queued (Line 10). Before updating the model, the model version and the worker's clock are incremented (Lines 3 and 4). For mitigating the gradient staleness, a staleness penalty is calculated based on the current model version of M_{Global} and the global model used by the worker (Line 5). We calculate an update rate based on the staleness penalty and the number of workers in the workers' vector clock (Line 6), iterate over the layers, and aggregate the updates (Lines 7 and 8). Finally, the queued updates are processed following the same procedure (Line 11).

4 EVALUATION

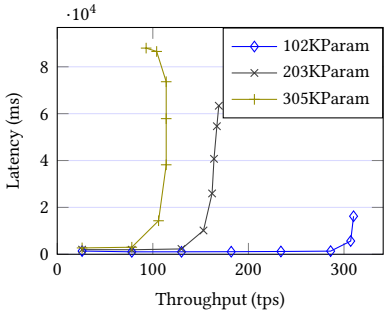

Figure 3: Latency to throughput.

We deployed eight organizations and evaluated the protocol regarding aggregating updates. We gradually increased the transactions arrival rate from 50 tps to 400 tps on three models with an increasing number of trainable parameters. Each transaction submits a pretrained update. The model is a convolutional DNN trained on the MNIST dataset using TensorFlow/Keras. The average latency to throughput is shown in Figure 3. We observe that the latency gradually increases due to the CPU saturation on our experimental setup, which causes transactions to be queued.

5 RELATED WORK

Although the applicability of CRDTs in other fields has been studied [5], its applicability to FL has received limited attention. Studies have proposed asynchronous FL protocols [3, 10] that use logical clocks to bound gradient staleness without considering the Byzantine actors. Various blockchain and PoW-based FL systems, such as BlockFlow and BAFFLE, have been proposed to ensure trust [4, 8, 11].

REFERENCES

[1] K. Bonawitz, H. Eichner, W. Grieskamp, and *et al.* 2019. Towards Federated Learning at Scale: System Design. *PMLR* (2019).
[2] F. Kohlbrenner, P. Nasirifard, C. Löbel, and H.-A. Jacobsen. 2019. A Blockchain-Based Payment and Validity Check System for Vehicle Services. In *Middleware Demos and Posters.*
[3] Q. Li, Z. Wen, Z. Wu, and *et al.* 2021. A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection. *IEEE TKDE* (2021).
[4] V. Mugunthan, R. Rahman, and L. Kagal. 2020. BlockFLow: An Accountable and Privacy-Preserving Solution for Federated Learning. *CoRR* (2020).
[5] P. Nasirifard, R. Mayer, and H.-A. Jacobsen. 2019. FabricCRDT: A Conflict-free Replicated Datatypes Approach to Permissioned Blockchains. In *Middleware.*
[6] P. Nasirifard, R. Mayer, and H.-A. Jacobsen. 2022. OrderlessChain: A CRDT-Enabled Blockchain Without Total Global Order of Transactions. In *Middleware Demos and Posters.*
[7] P. Nasirifard, R. Mayer, and H.-A. Jacobsen. 2022. OrderlessChain: Do Permissioned Blockchains Need Total Global Order of Transactions? *CoRR* (2022). https://doi.org/10.48550/arXiv.2210.01477 arXiv:2210.01477
[8] P. Ramanan and K. Nakayama. 2020. BAFFLE : Blockchain Based Aggregator Free Federated Learning. In *IEEE Blockchain.*
[9] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. 2011. Conflict-Free Replicated Data Types. In *SSS.*
[10] W. Zhang, S. Gupta, X. Lian, and J. Liu. 2016. Staleness-Aware Async-SGD for Distributed Deep Learning. In *IJCAI.*
[11] W. Zhilin and H. Qin. 2021. Blockchain-based Federated Learning: A Comprehensive Survey. *CoRR* (2021). arXiv:2110.02182