COMP4651 Fall 2024 Project Report

An IPFS-Blockchain-based Decentralized Federated Learning System for E-Commerce

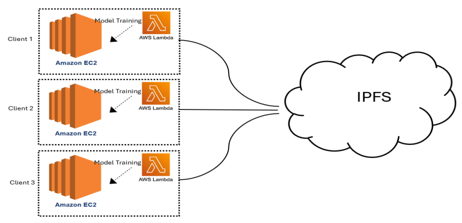
CHAN, Ho Wing (20853661, hwchanat), LEE, Pak Nin (20807222, pnleeab), LIU, Kwun Ho (20959984, khliuae)
Group 15, November 28, 2024

1 Introduction

In deep learning, accessing a large enough dataset and training a model on them are the most common challenges [1]. Especially in e-commerce, it is difficult for a small enterprise to acquire a large enough data sample to train an accurate recommendation system. While pooling data from multiple e-commerce clients to train a centralized model could enhance recommendation performance, such an approach is infeasible due to privacy concerns and long training time. Referencing from the distributed cloud computing anatomy, the Federated Learning (FL) system has emerged as an alternative [2]. Participating entities train models on their local dataset and share updates. However this framework relies on a trusting centralized server. To address these limitations, decentralized FL frameworks have been proposed, leveraging peer-to-peer interactions to eliminate the need for a central coordinator.

We propose an *IPFS-Blockchain-based Distributed Decentralized Federated Learning System* to facilitate cross-merchant recommendation model training. By leveraging FL systems, combined with blockchain, distributed storage systems, and cloud computing via Function as a Service provider (FaaS), this system provides privacy-protected collaboration and scalability while maintaining robustness. This project primarily focuses on the realization of distributed and parallel collaboration over the Internet rather than AI model optimization.

2 Design



This project consists of 4 components, a front-end UI, a neural network for recommendation system, a IPFS-blockchain storage system, and a federated learning system.

Figure 1: AWS service usage diagram

2.1 Frontend Structure

The front-end design consists of 4 different pages: index, personal, recommendation, and contribution.

2.1.1 Index Page

It serves as a landing page where the client (representing merchant) can login by entering their client ID. For minimal implementation, we assume only 3 merchants - *Shop 0*, *Shop 1* and *Shop 2*. There are several elements on the index page.

(a) a logo located at the top, adding a branding element and visual interest.

(b) A drop-down list for the client to select their IDs with clear labels and placeholders for guiding the user.

(c) a fault checker in case the user enters a client ID that doesn't exist in our database.

2.2.2 Personal Page

It serves as a page where 3 "Today's Featured Items" are randomly showcased. At the bottom, there are 3 buttons: "Contribute", "Get Recommendations" and "LOG OUT" linking to the corresponding pages.

2.2.3 Recommendation Page

This page gives recommendations to merchants on which products they should promote more to meet customers' needs. The model inference pipeline is shown in Figure 2.

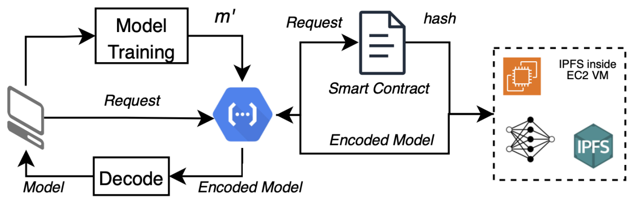


Figure 2: Model Inference Pipeline (Blue icon represent Google Cloud Platform Cloud Run Function)

2.2.4 Contribution Page

The core page of the system, clients can upload their transaction records to model training, and initiate the federated learning pipeline via HTTP requests.

2.2 Recommendation System Model Architecture

We used the *"eCommerce behavior data from multi-category store"* dataset, an open source dataset containing a total of about 285,000,000 rows and 9 columns [3]. However, due to the lack of training resources, we will only use around 30,000 data points. We separated the 30,000 data points into 3 batches of 10,000 data points, spread across 3 clients.

The model is a simple Stochastic Gradient Descent, with inputs consisting of a User to Product matrix (mapped for each User and Product Pair) and a label Tensor based on the action of the user toward a specific product.

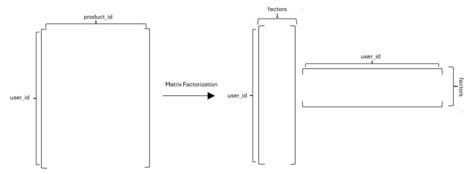


Figure 3: Matrix Factorization

We used a collaborative filtering system for the users and items as it doesn't require feature information of users and items besides the user-item interaction data [4]. To do that, we have to learn the matrix factorization of the User to Product matrix to represent each of the products and users by 20-dimensional vectors (factors) by default. For the architecture of the model, we used 4 embedding layers for the "user factors", "item factors", "user biases" and "item biases". All the biases are initialized to zeros. To learn the factorization, we used Stochastic Gradient Descent.

2.3 IPFS-Blockchain Storage System

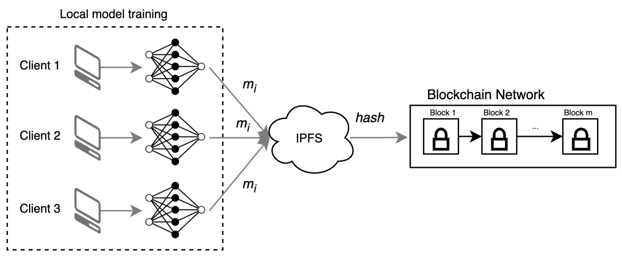


Figure 4. Our IPFS-Blockchain storage architecture

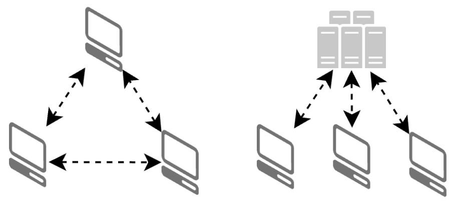


Figure 5. Decentralized(left) vs Centralized(right) FL

2.4.1 InterPlanetary File System (IPFS)

IPFS is a peer-to-peer file sharing protocol that provides decentralized and distributed file storage [5]. It replaces location-based addressing with unique content-based hashes (CID). IPFS is proven to have low latency, while avoiding single-point failure given a distributed manner [6].

2.4.2 Blockchain Smart Contract

Blockchain is a decentralized ledger technology that leverages a series of cryptographically linked blocks [7]. In this system, Ethereum smart contracts are used to manage the CID hashes from IPFS [8]. The objective is to ensure trustless collaboration given the immutability and transparency of blockchain that eliminate the need for a central authority.

2.4.3 Overall Architecture

As shown in Figure 4, each client performs model training locally. The model is then uploaded to IPFS in return for a unique CID hash. This hash is uploaded to an Ethereum smart contract, which acts as a decentralized registry. Under this architecture, the CID hash remains undercovered and no one knows where their model is stored, each client can only retrieve the model by interacting with the smart contract. This provides a high level of security, traceability and a transparent record to protect all business data.

2.5 Federated Learning Architecture

Federated Learning(FL) is a machine learning system that allows multiple clients to collaboratively train a model without exchanging raw data directly [9]. The two architectures of FL are shown in Figure 5. In centralized FL, a centralized server is required to aggregate models across all entities; While in decentralized FL, each entity is interconnected and communicated in a peer-to-peer manner. Decentralized architecture was adopted in this project to uphold privacy protection, fault tolerance, and most importantly scalability [10].

3 Implementation

3.1 Frontend Interface

For the implementation of the front end, we used the Python package Flask for hosting on three separate Amazon EC2 instances to emulate three different clients. We have written the layout and design in HTML and CSS.

3.2 Recommendation System

We used PyTorch for the model building. The data preprocessing is modified from *"recommendation_engine"* [11]. The dataset is first truncated to only relevant columns (user_id, product_id, interaction). Interaction is a new column created using elements from event_type and a random deviation. The actual event_types (view, cart, purchase) are used as validation. The train/test split ratio is 9:1, which is 9000 train data and 1000 test data.

The model is modified from *"Explicit Recommender System"* [12]. We train the model for 100 epochs, then the model is saved. Inference can be done by passing a user tensor and a products tensor (which is simply a range of all product ids), which returns a tensor of predicted ratings. The indices from sorting by decreasing order (topk or argmax) yields the predicted recommended products.

3.3 IPFS-Blockchain Storage System

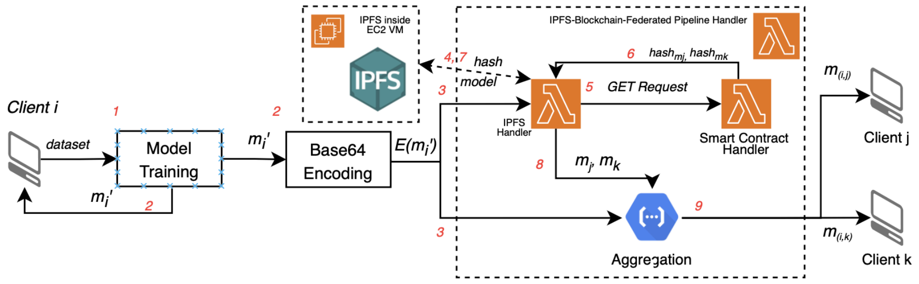


Figure 5. Our IPFS-Blockchain-Federated Learning pipeline architecture

We implemented 4 cloud functions on AWS Lambda and GCP Cloud Run Function. We utilized the IPFS distribution from ipfs.tech inside a EC2 VM. The smart contract was built using Solidity to store and read CID hashes that represent where their models are stored in IPFS. This contract is deployed on Ethereum Sepolia Testnet and linked with MetaMask cryptocurrency wallet.

The client first trained a new model locally with the above mentioned methods, then it was encoded as Base64 string and triggered the storage Lambda function to store it on IPFS. After that we retrieve the models of all other clients via a smart contract handler to get CID hashes plus the IPFS handler to get the model file. At last we trigger the aggregation function on GCP to update models of each client and distribute to them.

3.4 Federated Learning System

We implemented a basic aggregation function to combine model updates from clients. After triggering the aggregation system, the system retrieves weights from their models and performs below arithmetics operations.

$$w'_j = \frac{1}{N}\sum_{i=1}^{N} w_i$$

w_j represents weights and biases from each model, and we perform averaging to update the model.

While there are other advanced aggregation approaches like FedAvg, Differential Privacy (DP) and Secure Multi-Party Computation (SMPC), we chose averaging due to the limited time frame [13][14][15]. Despite the simplicity of the chosen approach, future iterations of this system can incorporate these advanced methods to improve performance.

4 Evaluation

4.1 Model Prediction

Aside from showing the recommended items in frontend, we use another metric to evaluate the model. During the training of the model, the train and test loss of the model is recorded. We can see that the loss of the model decreases each epoch. Therefore, this demonstrates the model is more effective in replicating the original data.

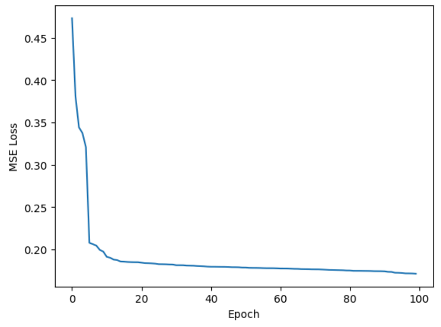


Figure 6. Model training Loss against epoch

4.2 Performance of IPFS

We performed streaming tests on IPFS hosted on EC2 vs AWS S3 buckets by measuring upload and download time for files of different sizes. We can see that generally IPFS takes more time in both operations. Movever, as file size increases, upload time of IPFS increases significantly. This highlights the trade-offs between distributed and centralized storage, while IPFS prevents single-point failure, hashing and distributing files across the network becomes a bottleneck, which may not be ideal for high-throughput scenarios involving large files.

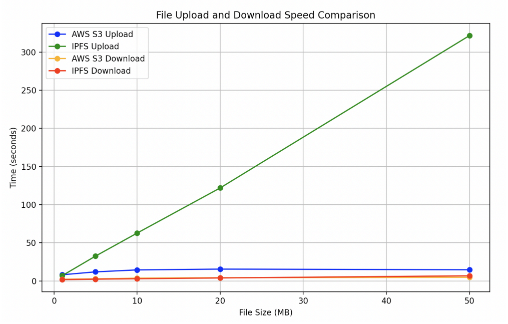
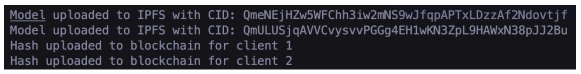


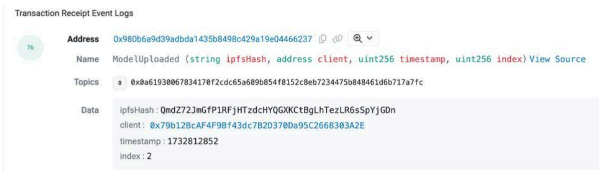
Figure 7. Streaming speed of IPFS and S3

4.3 Transparent Record

Supported by Blockchain architecture, each hash update is recorded which is immutable. This provides transparent records of which merchants updated and how the model is being used.



```
Model uploaded to IPFS with CID: QmeNEjHZw5WFChh3iw2mNS9wJfqpAPTxLDzzAf2Ndovtjf
Model uploaded to IPFS with CID: QmULUSjqAVVCvysvvPGGg4EH1wKN3ZpL9HAWxN38pJJ2Bu
Hash uploaded to blockchain for client 1
Hash uploaded to blockchain for client 2
```



```
Transaction Receipt Event Logs

76   Address  0x980b6a9d39adbda1435b8498c429a19e04466237
     Name     ModelUploaded (string ipfsHash, address client, uint256 timestamp, uint256 index) View Source
     Topics   0 0x0a61930067834170f2cdc65a689b854f8152c8eb7234475b848461d6b717a7fc
     Data     ipfsHash : QmdZ72JmGfP1RFjHTzdcHYQGXKCtBgLhTezLR6sSpYjGDn
              client : 0x79b12BcAF4F9Bf43dc7B2D370Da95C2668303A2E
              timestamp : 1732812852
              index : 2
```

Figure 8. Smart Contract Transaction Records

References

1. Roy, A. G., Siddiqui, S., Pölsterl, S., Navab, N., & Wachinger, C. (2019). Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*.
2. McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B., “Communication-Efficient Learning of Deep Networks from Decentralized Data”, <i>arXiv e-prints</i>, Art. no. arXiv:1602.05629, 2016. doi:10.48550/arXiv.1602.05629.
3. Kechinov, M. (2019) *“eCommerce behavior data from multi-category store”*. Available at: https://www.kaggle.com/datasets/mkechinov/ecommerce-behavior-data-from-multi-category-store
4. Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000, December). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (pp. 241-250).
5. Benet J. Ipfs-content addressed, versioned, p2p file system[J]. arXiv preprint arXiv:1407.3561, 2014.
6. Y. Chen, H. Li, K. Li and J. Zhang, "An improved P2P file system scheme based on IPFS and Blockchain," *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, USA, 2017, pp. 2652-2657, doi: 10.1109/BigData.2017.8258226.
7. Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf
8. Buterin, V. (2013). Ethereum white paper. *GitHub repository*, *1*, 22-23.
9. Bharati, S., Mondal, M. R. H., Podder, P., & Prasath, V. S. (2022). Federated learning: Applications, challenges and future directions. *International Journal of Hybrid Intelligent Systems*, *18*(1-2), 19-35
10. Beltrán, E. T. M., Pérez, M. Q., Sánchez, P. M. S., Bernal, S. L., Bovet, G., Pérez, M. G., ... & Celdrán, A. H. (2023). Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys & Tutorials*.
11. derinsu. (2023, May 12). recommendation_engine. Kaggle.com; Kaggle. https://www.kaggle.com/code/derinsu/recommendation-engine#Recommend-Products-to-Users-Viewing-a-Product
12. Buoy, R. (2019, December 9). Explicit Recommender System - Rina Buoy - Medium. Medium. https://medium.com/@rinabuoy13/explicit-recommender-system-matrix-factorization-in-pytorch-f3779bb55d74
13. Li, X., Huang, K., Yang, W., Wang, S., & Zhang, Z. (2019). On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*.
14. Geyer, R. C., Klein, T., & Nabi, M. (2017). Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*.
15. Goldreich, O. (1998). Secure multi-party computation. *Manuscript. Preliminary version*, *78*(110), 1-108.