When Federated Learning Meets Blockchain: A New Distributed Learning Paradigm

Authors: Chuan Ma, Jun Li, Ming Ding, Long Shi, Taotao Wang, Zhu Han, H. Vincent Poor

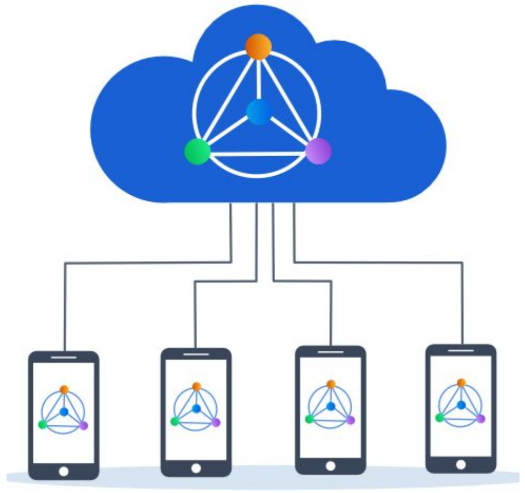
Prepared and Presented by Mahdis Rahmani

Outline:

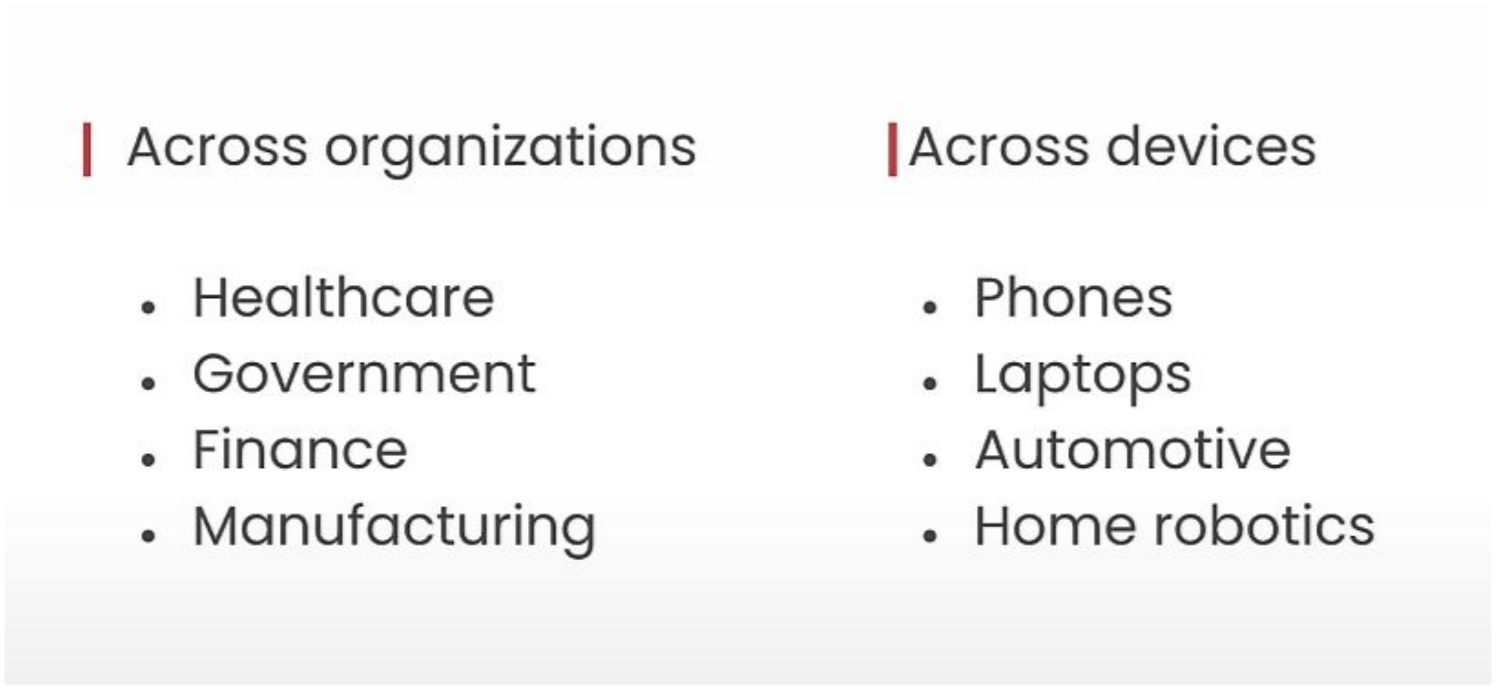
Federated Learning (FL)

A new machine learning paradigm, motivated by:

- explosive computing capabilities at end user equipments
- the growing privacy concerns over sharing sensitive raw data



Data is naturally distributed



| Across organizations

- Healthcare
- Government
- Finance
- Manufacturing

| Across devices

- Phones
- Laptops
- Automotive
- Home robotics

Traditional training assumes central data.

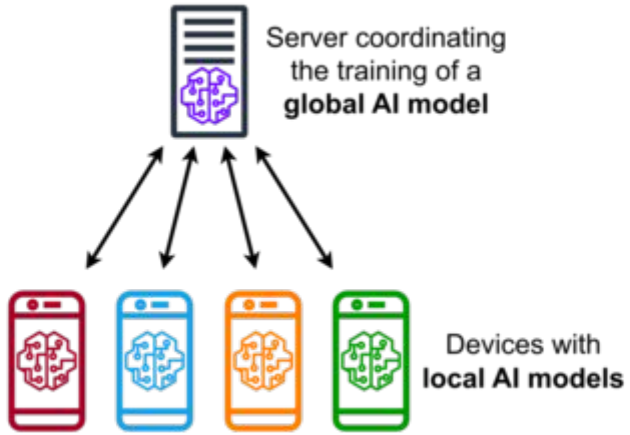
Collecting data often does not work

- Data needs to move
- Often not possible
- Many reasons

- Sensitive data
- Data volume
- Privacy
- Regulations
- Practicality

What is Federated Learning?

Federated learning (also known as **collaborative learning**) is a machine learning technique focusing on settings in which multiple entities (often referred to as clients) collaboratively train a model while ensuring that their data remains decentralized. This stands in contrast to machine learning settings in which data is centrally stored.


Server coordinating the training of a global AI model
Devices with local AI models

Federated Learning (Intuition and Algorithm)

1. Initialization:

Server initializes the global model

2. Communication Round:

For each communication round:

- Server sends the global model to participating clients
- Each client receives the global model

3. Client Training and Model Update:

For each participating client:

- Client trains the received model on its local dataset
- Client sends its locally updated model to the server

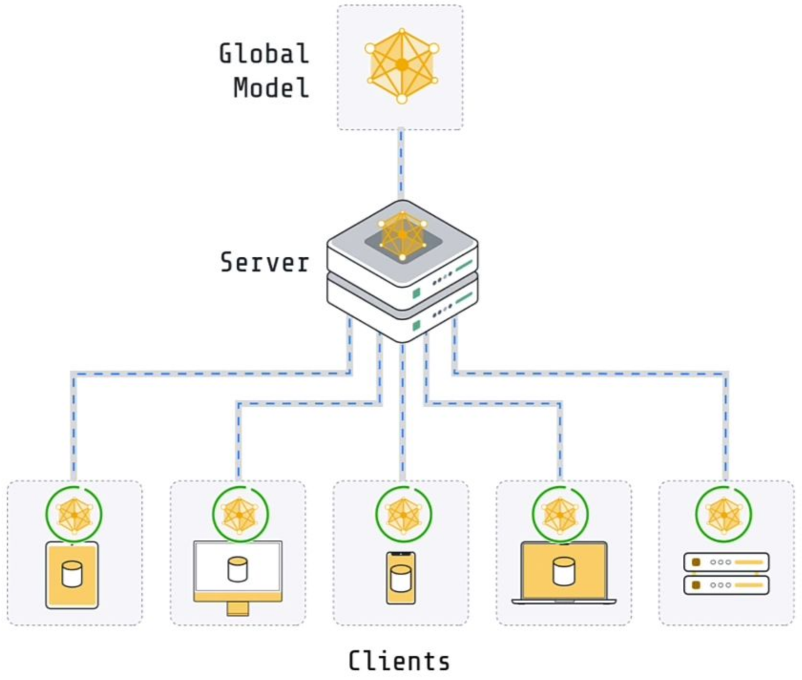
4. Model Aggregation:

Server aggregates the updated models received from all clients using Aggregation Algorithm (for instance, FedAvg)

5. Convergence Check:

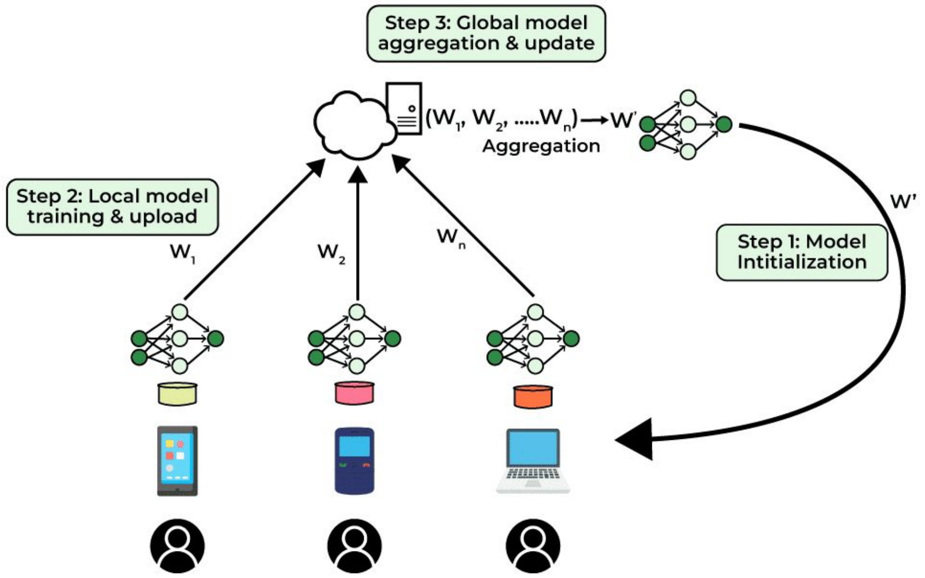
If convergence criteria are met, end the FL process

If not, proceed to the next communication round (step 2)



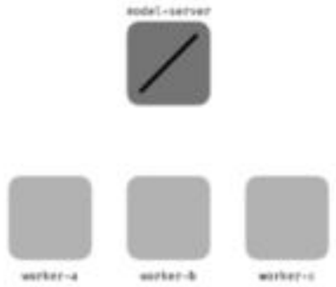
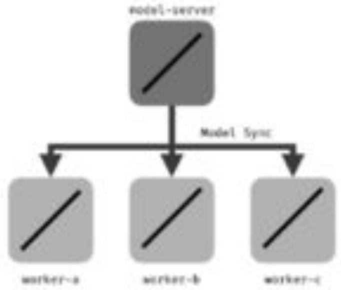
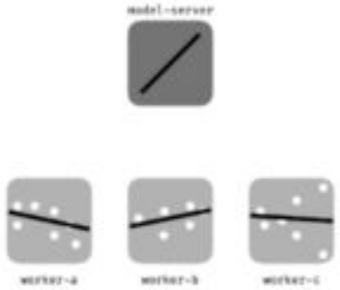
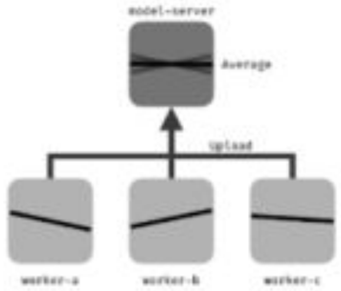
Each round of the FL process:

1. Select edge devices for training
2. Users download the initial model
3. Users train model locally
4. Users update their model updates (parameters/weights) to server
5. Server aggregates received updates
6. Repeat until convergence

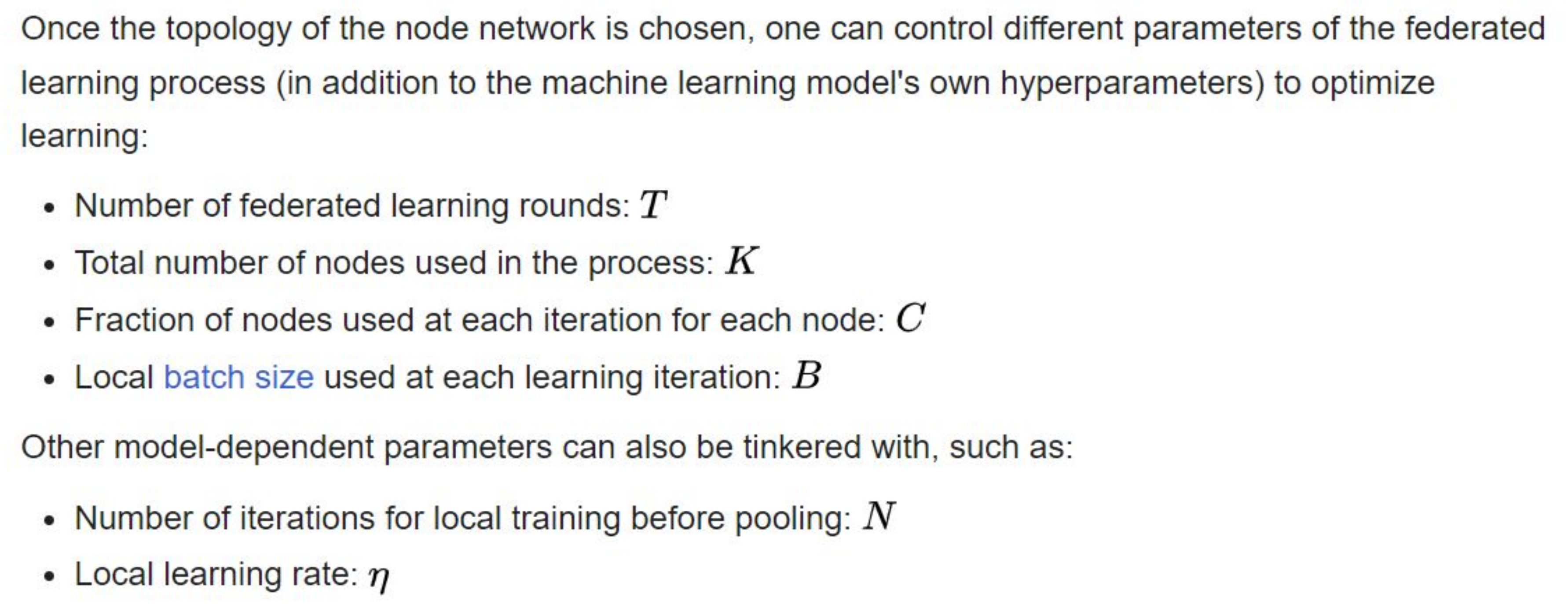


Step 3: Global model aggregation & update
$(W_1, W_2, \ldots W_n) \rightarrow W'$
Aggregation
Step 2: Local model training & upload
W_1
W_2
W_n
W'
Step 1: Model Intitialization

Each round of the FL process:

Step 1	Step 2	Step 3	Step 4
	 model-server Model Sync worker-a worker-b worker-c		 model-server Average Upload worker-a worker-b worker-c
Central server chooses a statistical model to be trained	Central server transmits the initial model to several nodes	Nodes train the model locally with their own data	Central server pools model results and generate one global mode without accessing any data

Federated learning parameters



Once the topology of the node network is chosen, one can control different parameters of the federated learning process (in addition to the machine learning model's own hyperparameters) to optimize learning:

- Number of federated learning rounds: T
- Total number of nodes used in the process: K
- Fraction of nodes used at each iteration for each node: C
- Local batch size used at each learning iteration: B

Other model-dependent parameters can also be tinkered with, such as:

- Number of iterations for local training before pooling: N
- Local learning rate: η

Why Blade-FL?

Traditional FL framework still faces some problems which undermine the reliability of the whole system:

- Single point of failure (the server / aggregator) —> decentralization
- Malicious clients and false or poisonous data —> verifying updates
- Lack of Incentive (lazy nodes) —> mining reward for verifying and mining

This is not THE ONLY architecture, we'll categorize all proposed architecture.

Decentralized Federated Learning:

- Utilize Blockchain in federated learning, some applications include:

- Verify local model updates (uploaded parameters)
- Encryption during model transmission
- Traceability: immutable audit trail of ML models can be created for greater trustworthiness in tracking and proving provenance

Audit Trail

[ˈȯ-dət ˈtrāl]

A step-by-step record by which accounting, trade details, or other financial data can be traced to their source.

Revisiting Blockchain:

Let’s take a look at this demo developed by Anders Brownworth to revisit concepts used in the Blockchain technology.

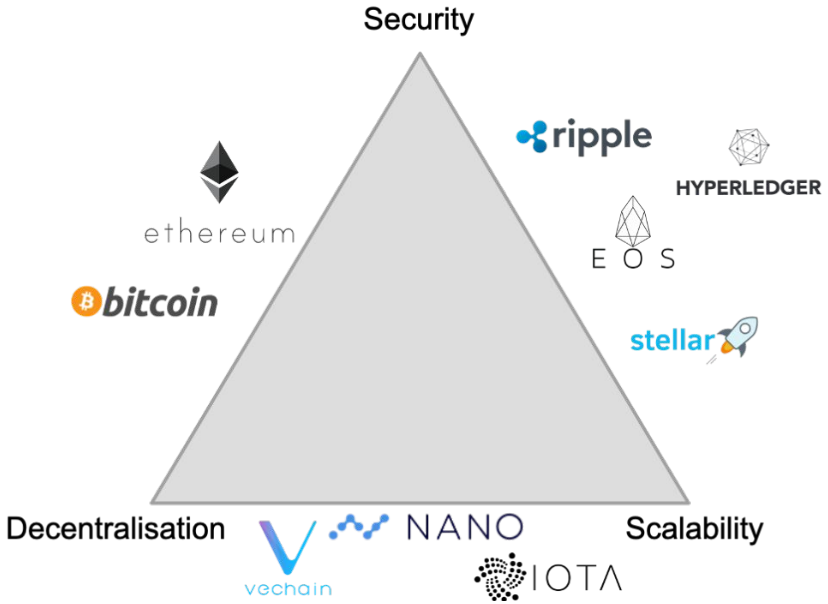
Demo: https://andersbrownworth.com/blockchain/

Code available on: https://github.com/anders94/blockchain-demo

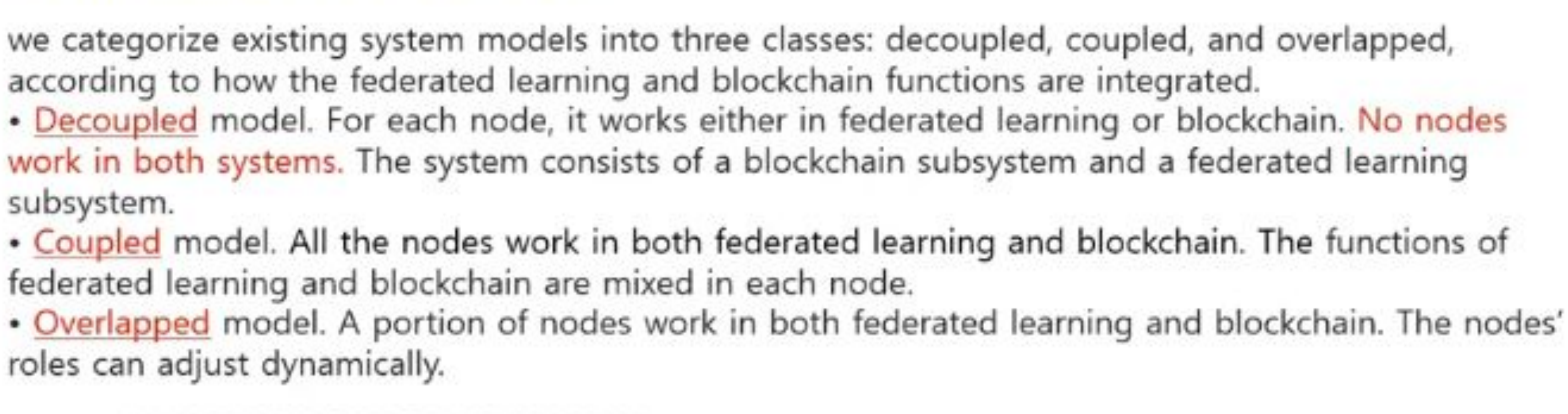
Blockchain Trilemma:

The blockchain trilemma refers to the challenge of achieving three critical aspects of blockchain technology: **security**, **scalability**, and **decentralization**.

The trilemma suggests that optimizing one aspect often compromises the others, making it difficult to achieve all three simultaneously.



Common BC-FL Architectures:



we categorize existing system models into three classes: decoupled, coupled, and overlapped, according to how the federated learning and blockchain functions are integrated.

• Decoupled model. For each node, it works either in federated learning or blockchain. No nodes work in both systems. The system consists of a blockchain subsystem and a federated learning subsystem.

• Coupled model. All the nodes work in both federated learning and blockchain. The functions of federated learning and blockchain are mixed in each node.

• Overlapped model. A portion of nodes work in both federated learning and blockchain. The nodes' roles can adjust dynamically.

Architecture layers

The BLADE-FL framework is composed of three layers:

Network Layer:

- Peer-to-peer communication

Blockchain Layer:

- Secure record-keeping

Application Layer:

- Smart contracts and FL tasks

Network Layer:

In the network layer, the network features a decentralized P2P network that consists of task publishers and training clients, wherein a learning mission is first published by a task publisher, and then completed by the cooperation of several training clients. Different from previous work that model aggregation happens in a trust community in the blockchain, we realize a fully decentralized framework that each client needs to train ML models and mine blocks for publishing aggregating results.

Blockchain Layer:

In the blockchain layer, each FL-related event, such as publishing a task, broadcasting learning models, and aggregating learning results, is tracked by blockchain.

Application Layer:

In the application layer, the SC and FL are utilized to execute the FL-related events.

Next, we will detail the working flow and key components of the BLADE-FL framework.

BLADE-FL Working Flow

1. **Task Publishing:**
   - FL tasks are broadcast using smart contracts (SC).
   - Incentives are defined.
2. **Local Model Training:**
   - Participants train models using their local data.
3. **Model Aggregation:**
   - Decentralized aggregation of updates.
4. **Mining and Block Generation:**
   - Use of Proof of Work (PoW)
5. **Model Distribution:**
   - Clients download aggregated models.
6. **Reward Allocation:**
   - Incentives distributed based on contributions.

Task publishing and node selection

A task publisher broadcasts a FL task through deploying a SC over the blockchain network. In the deployed SC, the task publisher needs to deposit reward as financial incentives to the learning task. The SC selects available training nodes to participate in this learning task.

Local model broadcast

Each training client runs its local training by using its own data samples and broadcasts its local updates and the corresponding processing information (e.g., computation time and local data size) over the P2P network. Privacy leakage may happen during this transmission, and we further investigate this issue in the future.

Model Aggregation

In the BLADE-FL framework, **model aggregation** is carried out in a **decentralized manner**. Unlike traditional Federated Learning (FL), which relies on a central server to aggregate models, BLADE-FL disperses this responsibility across participating clients. Here’s how it works:

1. **Local Aggregation by Clients**:
    - Each client collects locally trained model updates from other participants in the peer-to-peer (P2P) network.
    - Aggregation rules are predefined in a **smart contract (SC)**, ensuring consistency and fairness.
    - Clients aggregate the updates independently to compute the global model.

Model Aggregation

1. **Blockchain-Based Verification**:
   - The aggregated model is recorded as part of a block.
   - This block undergoes a consensus process, such as **Proof of Work (PoW)**, ensuring that the aggregation result is validated by other clients in the blockchain network.
2. **Block Generation and Propagation**:
   - Once the block containing the aggregated model is verified, it is added to the blockchain.
   - This ensures that all clients have access to the same aggregated model, stored immutably in the blockchain.
3. **Incentive for Honest Aggregation**:
   - Clients contributing to aggregation are incentivized through the blockchain's reward mechanism, as defined in the SC.
   - Misbehaving clients uploading low-quality or poisoned models are penalized, ensuring integrity.

This decentralized model aggregation eliminates reliance on a single central server, enhancing the system's robustness and security.

Block generation

Role Transition:

- After training, each client switches roles from **trainer** to **miner** to participate in block generation.

Mining Process:

- Clients mine until they:
  - **Find the required nonce** (solving the cryptographic puzzle).
  - Or **receive a block** generated by another miner.

Block Content:

- Each block contains:
  - Learning results (aggregated models).
  - State changes defined by the Smart Contract (SC).
  - Transactions.

Verification:

- Other clients verify the block by checking:
  - The validity of the nonce.
  - Consistency of the SC-defined state and aggregated results.

Resource Allocation Issue:

- Balancing resources for training and mining is a key challenge, requiring optimization to maximize efficiency.

Block propagation

If a block is verified by the majority of clients, this block will be added on the blockchain and accepted by the whole network. The lazy client issue happens in this step and we further investigate it in the future.

Global model download and update

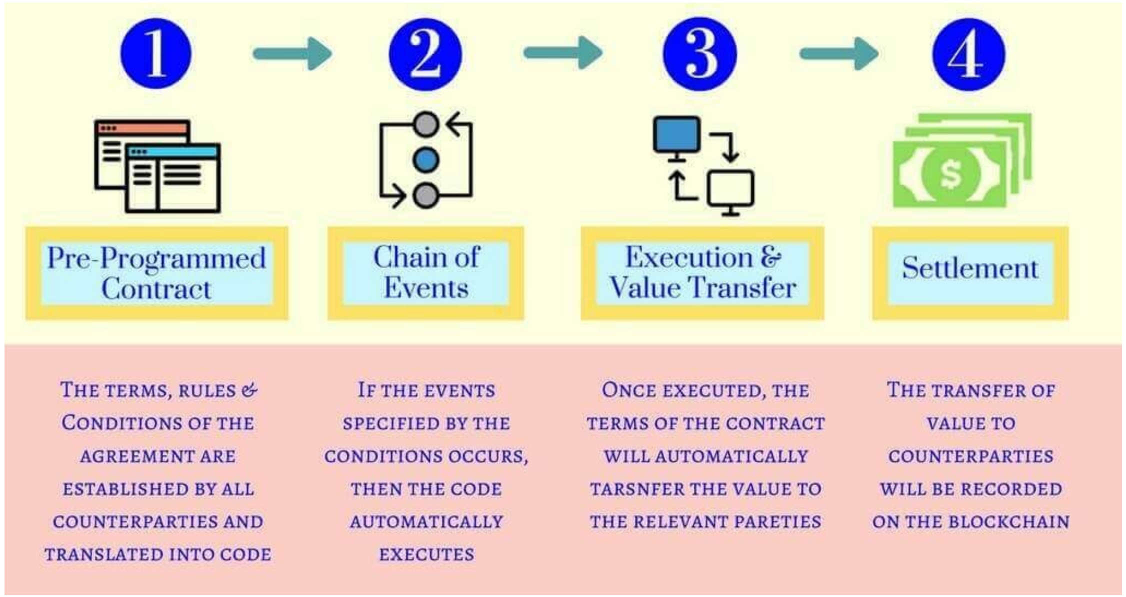
Each training client downloads the aggregated model from the block and performs updates before the next round of learning.

Reward allocation

The SC deployed by the task publisher rewards the training clients according to their contributions in the learning task.

What is a smart contract?

A **smart contract** is a computer program or a transaction protocol that is intended to automatically execute, control or document events and actions according to the terms of a contract or an agreement.


1
2
3
4
Pre-Programmed Contract
Chain of Events
Execution & Value Transfer
Settlement
The terms, rules & conditions of the agreement are established by all counterparties and translated into code
If the events specified by the conditions occurs, then the code automatically executes
Once executed, the terms of the contract will automatically tarsnfer the value to the relevant pareties
The transfer of value to counterparties will be recorded on the blockchain

Smart Contract Design

The SC in BC-FL enables three main functions as follows:

- **Function 1:** Learning task publishing
- **Function 2:** Dynamic bidding for requests and automatic incentive
- **Function 3:** Learning Results Aggregation And Rewards Feedback

Function 1: Learning task publishing

1. **Task Broadcasting**:
    - The task publisher uses a **Smart Contract (SC)** to announce an FL task to all users in the network.
2. **Details in the SC**:
    - **Task Requirements**:
        - Data size
        - Desired training accuracy
        - Latency constraints
    - **Aggregating Rules**:
        - Defines how models are combined to form the global model.
    - **Rewards**:
        - Specifies incentives for participants based on their contributions.
3. **Purpose**:
    - Ensures a fair, transparent, and automated process without relying on a centralized authority.

Function 2: Dynamic bidding for requests and automatic incentive

1. **Bidding Process**:
   - Distributed training nodes act as **auctioneers**, submitting bids with their costs and capabilities.
2. **Deposit Requirement**:
   - Each client stakes a **deposit** to ensure accountability.
3. **Selection of Winners**:
   - The SC evaluates bids and selects nodes with:
     - Higher capabilities (e.g., computational power, data quality).
     - Lower costs.
   - Winning nodes are assigned the FL task.
4. **Post-Bidding Outcomes**:
   - **Losing Clients**: Reclaim their deposits.
   - **Winning Clients**: Deposits are refunded **after verification** of their trustworthy contributions.
5. **Blockchain Role**:
   - All bidding data and outcomes are immutably recorded on the blockchain.

Function 3: Learning Results Aggregation And Rewards Feedback

1. **Model Aggregation**:
    - Each client aggregates uploaded models based on **rules defined in the Smart Contract (SC)**.
    - Contributions from each participant are recorded in the newly generated block.
2. **Reward Mechanism**:
    - The SC automatically calculates and allocates rewards to:
        - **Miners**: For successfully aggregating the model and generating the block.
        - **Training Clients**: Based on their contribution to the FL process.
3. **Transparency and Fairness**:
    - The blockchain ensures that all contributions and rewards are recorded immutably, fostering trust and accountability.

The BLADE-FL Design:

The key steps are illustrated as follows:

- Local model update and upload
- Model aggregation
- Model recording and publishing
- Reward allocation

Local model update and upload

1. Training Nodes:

- **Bid Winners**: Selected nodes with capable devices and sufficient data samples.
- Update local machine learning (ML) models in **parallel** using:
  - The global model from the previous iteration.
  - Local data samples.

2. Broadcasting Updates:

- Each node broadcasts its trained model to the network via the **gossip protocol** in the P2P network.

3. Decentralized Aggregation:

- Instead of relying on a central server, each client:
  - Receives local updates from others.
  - Stores updates in a **model pool** for decentralized aggregation.

Model aggregation

1. Global Model Calculation:

- Each client collects models from the **pool** and calculates the global model updates based on **aggregating rules in the Smart Contract (SC)**.
- Clients are designed to aggregate learning parameters truthfully using a **distributed ledger**.

2. Block Structure:

- **Body**:
  - Stores local model updates, including:
    - Local data size.
    - Computing time of training nodes.
    - Aggregated learning parameters.
- **Header**:
  - Contains metadata such as:
    - Pointer to the previous block.
    - Block generation rate.
    - Proof of Work (PoW) or other consensus-related output values.

3. Integrity:

- This structure ensures transparency, immutability, and accurate aggregation within the blockchain.

Model recording and publishing

Recording and Broadcasting:

- Clients store aggregated models in a block.
- The generated block is broadcasted to the entire network.

Consensus Protocols:

- Block generation uses protocols like:
  - **Proof of Work (PoW)** (preferred for strong security).
  - Proof of Stake (PoS)

Mining Process:

- All miners start simultaneously in a **synchronous schedule**.
- Miners search for a nonce that meets the PoW difficulty level.
- Once found, the block becomes a candidate for the blockchain.

Block Verification:

- Other clients verify the block by checking:
  - The validity of the nonce.
  - Aggregated model results, using a public testing dataset or comparison with the publishing block.
- **Outcome**:
  - **Accepted**: Added to the blockchain if valid.
  - **Rejected**: Discarded if invalid, and mining continues on the last valid block.

Reward allocation

Reward Allocation:

- Task publisher distributes rewards to training nodes.
- Rewards are proportional to:
  - **Size of training data**.
  - **Quality of data samples** (if considered).

Ensuring Trustworthiness:

- Clients verify the **integrity** of local updates to prevent exaggeration of sample sizes or abnormal model updates.
- Nodes are scored or ranked based on model quality:
  - Low-scoring nodes receive **reduced weights** in future aggregations.
  - Persistent low-quality nodes are **ignored** over time.

Secure Verification:

- Guaranteed by technologies like Intel’s **Software Guard Extensions (SGX)**, which provide a secure execution environment.

Miner Rewards:

- Miners earn additional rewards for block generation and model aggregation, akin to a **gas tax** in blockchain systems.

Unique Issues And Potential Solutions:

1. Privacy:

- **Issue**: Sharing model updates risks privacy leakage.
- **Solution**: Use **Local Differential Privacy (LDP)** by adding random noise to updates. Adaptive noise decaying improves performance.

2. Resource Allocation:

- **Issue**: Limited computing resources must balance between local training and mining.
- **Solution**: Optimize computation resources using an allocation strategy based on task constraints.

3. Lazy Nodes:

- **Issue**: Some clients may avoid local training and plagiarize updates to save resources.
- **Solution**: Use **Pseudo-Noise (PN) sequences** to detect and penalize lazy nodes.

Pseudo-Noise (PN) sequences

Pseudo-Noise (PN) sequences are unique, random-like codes generated by each client and added to their model updates. These sequences act as invisible signatures that help identify the true origin of the updates. If a lazy client tries to copy another's updates, the PN sequence will reveal the plagiarism through its distinct correlation pattern.

Future Directions:

- **Enhanced Reward Mechanisms:**
  - Fair distribution based on contributions.
- **Lightweight Model Transmission:**
  - Use quantization techniques to reduce transmission cost.

Thank you for attending this presentation!

Questions? :)