计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(9):69−76 [doi: 10.15888/j.cnki.csa.008075]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于区块链的工业物联网联邦学习系统架构①

于秋雨, 卢清华, 张卫山

(中国石油大学 (华东) 计算机科学与技术学院, 青岛 266580)
通讯作者: 卢清华, E-mail: dr.qinghua.lu@gmail.com

摘　要: 联邦学习是一种新兴的保护隐私的机器学习算法, 它正在广泛应用于工业物联网 (IIoT) 中, 在联邦学习中中心服务器协调多个客户端 (如物联网设备) 在本地训练模型, 最后融合成一个全局模型. 最近, 区块链在工业物联网和联邦学习中得到了利用, 以用来维护数据完整性和实现激励机制, 吸引足够的客户数据和计算资源用于培训. 然而, 基于区块链的联邦学习系统缺乏系统的架构设计来支持系统化开发. 此外, 目前的解决方案没有考虑激励机制设计和区块链的可扩展性问题. 因此, 在本文中, 我们提出了一个应用于工业物联网中基于区块链的联邦学习系统架构, 在此架构中, 每个客户端托管一个用于本地模型训练的服务器, 并管理一个完整的区块链节点. 为了实现客户端数据的可验证完整性, 同时考虑到区块链的可扩展问题, 因此每个客户端服务器会定期创建一个默克尔树, 其中每个叶节点表示一个客户端数据记录, 然后将树的根节点存储在区块链上. 为了鼓励客户积极参与联邦学习, 基于本地模型培训中使用的客户数据集大小, 设计了一种链上激励机制, 准确、及时地计算出每个客户的贡献. 在实验中实现了提出的架构的原型, 并对其可行性、准确性和性能进行了评估. 结果表明, 该方法维护了数据的完整性, 并具有良好的预测精度和性能.

关键词: 区块链; 联邦学习; 机器学习; 边缘计算; 物联网; 人工智能; 故障检测

引用格式: 于秋雨,卢清华,张卫山.基于区块链的工业物联网联邦学习系统架构.计算机系统应用,2021,30(9):69–76. http://www.c-s-a.org.cn/1003-3254/8075.html

Federated Learning System Architecture in Industrail IoT Based on Blockchain

YU Qiu-Yu, LU Qing-Hua, ZHANG Wei-Shan

(College of Computer Science and Technology, China University of Petroleum, Qingdao 266580, China)

Abstract: Federated learning is an emerging privacy-preserving machine learning paradigm widely applied to the Industrial Internet of Things (IIoT), where multiple clients (e.g. IoT devices) train models locally to formulate a global model under the coordination of a central server. Blockchain has been recently leveraged in IIoT federated learning to maintain data integrity and provide incentives to attract sufficient client data and computation resources for training. However, there is a lack of systematic architecture design for blockchain-based federated learning systems to support methodical development in IIoT. Also, the current solutions do not consider the incentive mechanism design and blockchain scalability. Therefore, in this study, we present a platform architecture of blockchain-based federated learning systems in IIoT, where each client hosts a server for local model training and manages a full blockchain node. For verifiable integrity of client data in a scalable way, each client server periodically creates a Merkle tree in which each leaf node represents a client data record and stores the tree root on a blockchain. To encourage clients to participate in federated learning, an on-chain incentive mechanism is designed based on the size of client data used in local model training to accurately and timely calculate each client's contribution. A prototype of the proposed architecture is

① 基金项目: 国家自然科学基金 (62072469)
Foundation item: National Natural Science Foundation of China (62072469)
收稿时间: 2020-12-06; 修改时间: 2021-01-08; 采用时间: 2021-01-20; csa 在线出版时间: 2021-09-02

implemented with our industry partner and evaluated in terms of feasibility, accuracy and performance. The results show that the approach ensures data integrity and has satisfactory prediction accuracy, and performance.

Key words: Blockchain; federated learning; machine learning; edge computing; IoT; AI; failure detection

1 背景简介

近几年来, 物联网 (IoT) 广泛应用于工业, 特别是制造业, 实现了智能工业物联网 (IIoT) 应用. 据估计, 到 2025 年, 全球将有 41.6 亿部物联网设备, 预计将产生 79.4 ZB 的数据[1]. 如何管理和利用这些物联网数据, 以高效、安全、经济的方式实现智能化, 成为一个重要的研究问题.

联邦学习是一种新兴的机器学习[2,3], 他可以保护数据隐私, 减少模型训练中的偏移. 每一轮联邦学习中, 被选中的客户 (如组织、数据中心, 物联网或移动设备) 在保证原始数据不上传的情况下, 在本地训练模型, 并共同训练了全局模型.

区块链技术最近一直被应用于工业物联网的联邦学习中, 用来保障数据完整性, 构造激励机制来吸引丰富的客户数据和计算资源加入训练[4,5]. 然而, 在工业物联网领域却缺少一个系统的整体的架构设计来支撑设备故障检测系统有条不紊地开发, 并高效地解决数据异质等问题.

因此, 在本文中, 我们提出了一种基于区块链的联邦学习系统架构. 在联邦学习的每一轮训练中, 中心服务器会选择一些客户端在本地训练模型, 并将模型更新后的参数 (而不是原始数据) 上传到中心服务器. 然后, 中心服务器将更新后的模型参数聚合以生成一个全面的、无偏的全局模型. 为了确保数据的完整性并解决区块链的性能问题, 在固定周期中, 对客户端数据进行哈希计算创建默克尔树, 并将树的根节点存储在区块链上. 我们还提出了激励机制, 根据参与本地训练的数据集大小, 通过智能合约为客户发放相应的奖励, 以此鼓励客户积极参与联邦学习. 最后, 我们实现了文中所提出的架构, 并且评估了系统架构的可行性、模型的精度和性能. 实验结果表明该架构可行, 并具有良好的精度和性能.

本文的贡献如下:

(1) 提出一个系统架构, 展示了不同部件间的交互, 为开发基于区块链的工业物联网联邦学习系统提供指导. 架构包括以下设计决策: 模型训练、客户数据存储、客户激励机制以及区块链部署.

(2) 提出区块链锚定协议, 确保可验证数据的完整性. 协议规定, 每隔一段时间, 使用客户端边缘设备收集到的数据创建默克尔树 (一个叶子节点代表一条客户数据), 并将根节点上传到区块链.

(3) 提出激励机制, 通过激励智能合约, 来鼓励客户贡献数据和计算资源参与模型训练, 用户会根据自己在本地训练模型的数据大小, 收到相应的代币奖励.

(4) 在真实的工业场景 (轴承故障检测) 中, 对系统架构可行性, 模型精度和性能进行了评估.

本文构造如下: 第 2 节提出方法, 第 3 节讨论系统框架的实现和评估, 第 4 节总结文章并部署未来工作.

2 基于区块链的联邦学习系统架构

通过标准的需求捕获方法, 我们对基于区块链的工业物联网联邦学习系统, 制定了一系列的功能需求和非功能需求. 其中功能需求包括: (1) 在本地服务器进行模型训练; (2) 在不上传本地数据的情况下, 基于本地模型参数生成全局模型; (3) 融合得到的全局模型可用于分类; (4) 根据客户在模型训练过程中的贡献进行奖励. 已确定的非功能需求包括: (1) 保护边缘数据隐私; (2) 具有良好的模型精度; (3) 确保客户端数据的完整性; (4) 维护区块链基础设施的可用性.

在本节中, 我们提出了一个基于联邦学习和区块链的系统架构, 以满足上述功能与非功能性需求. 2.1 节描述了系统整体架构, 并细化了组件及其交互; 2.2 节讨论了主要的架构设计决策; 2.3 节介绍了联邦加权平均算法啊; 2.4 节介绍了客户数据锚定协议以确保数据完整性. 2.5 节设计了一种激励机制, 鼓励客户积极参与模型训练.

2.1 系统整体架构

图 1 展示了由两种不同的参与组织 (中心组织和客户组织) 构成的系统整体架构. 联邦学习系统的所有者 (如制造商公司) 作为中心组织, 它负责根据系统的检测结果来维护所有工业服务; 而客户组织则负责为本地模型训练提供数据和计算资源.

每个客户组织拥有多个传感器、一个客户端设备、一个客户端服务器和一个区块链全节点. 而中心组织托管一个中心服务器和一个区块链全节点. 每个客户端设备 (如树莓派) 通过客户端数据收集器收集被传感器检测到的环境数据, 然后通过客户端数据预处理器清洗数据 (如降噪). 所有收集到的客户端环境数据都被存储在客户端服务器托管的客户端数据库中. 在某一联邦学习中, 当一个客户组被中心服务器选中, 客户端服务器将从中心服务器下载最新版本的全局模型, 通过模型训练器, 在本地数据集上进一步训练模型, 更新模型参数. 然后, 更新后的模型参数被传送到中心服务器上的融合器, 融合器将从客户端接收到的模型参数进行融合, 最终得到一个新的全局模型, 再将其返回客户端服务器和检测器. 在对设备进行故障检测时, 检测结果被加密后发送到中心组织, 操作人员通过数据解密器对接收到的检测结果进行解密, 并做出相应的处理决定.

每个客户端服务器会定期创建默克尔树, 其中一个叶子节点表示一条数据记录. 每一轮联邦学习中, 参与训练的客户信息 (包括默克尔树根节点、训练状态、用于模型训练的客户端数据的大小) 都通过客户端数据锚定器存储在预先部署在区块链上的智能合约中. 若不同组织之间发生了争议 (例如设备故障原因), 中心组织和客户组织的操作员可以使用争议解决器, 通过将数据与存储在链上的默克尔树根节点进行比较, 来验证客户历史数据在某一周期内的完整性. 为奖励客户组织对模型训练的贡献, 根据参与本地训练的客户端数据大小, 奖励客户组织相应数量的代币.

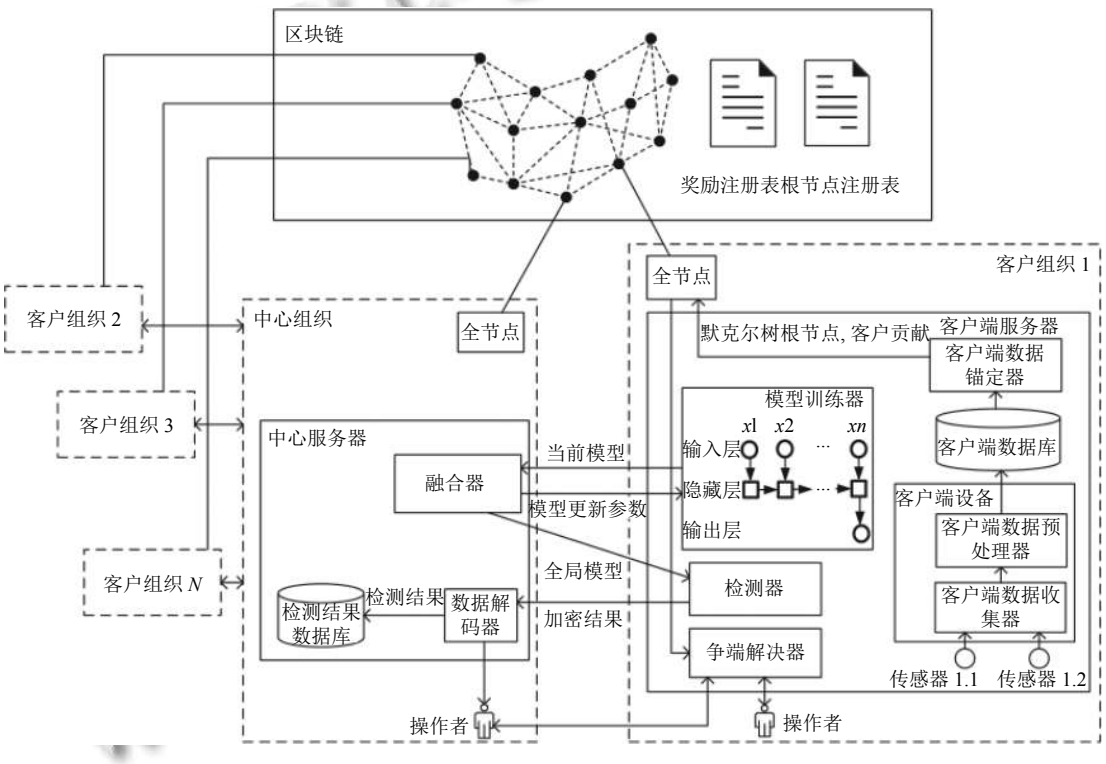


图 1 联邦学习系统架构

2.2 架构设计决策

(1) 本地模型训练: 为了保护客户组织的商业敏感数据 (例如, 机器使用频率和时间), 架构采用联邦学习, 客户端服务器通过模型训练器, 在本地训练模型, 中心服务器则负责通过融合器计算得到一个用于检测的全局模型.

(2) 锚定数据上链: 为了解决中心组织和客户组织之间的争端 (例如, 故障原因), 在架构中, 客户端服务器通过数据锚定器将数据上链, 来实现客户端数据的可验证完整性. 由于区块链的存储能力有限, 客户端服务器定期创建默克尔树 (树中每个叶子节点代表传感器收集的一条数据记录), 并通过数据锚定器, 将其根节点存储在预先部署在链上的智能合约 (根节点注册表) 中.

(3) 基于智能合约的激励机制: 为了鼓励客户组织积极参与模型训练, 该架构设计了基于智能合约的激

励机制. 根据客户组织用于本地训练的数据集大小, 奖励客户组织相应的代币. 所奖励的代币记录在智能合约 (奖励注册表) 中.

(4) 区块链部署: 在区块链网络中, 全节点存储着所有的区块链事务. 在我们提出的架构中, 每个参与组织 (包括中心组织和客户组织) 托管一个区块链全节点. 因此, 每个组织都区块链数据的完整副本, 可以用于审计, 同时确保整个系统的可用性.

2.3 联邦加权平均算法

在算法 1 中, 对于每一轮联邦学习, 中心服务器会根据一些预定义的条件 (如空闲和收费) 选择特定数量的客户端参与训练. 对于每个客户端服务器, 设置本地最小批大小为 B、学习率η、进行 E 期训练. 然后, 所选客户端服务器在数据量为 B 的子集上执行 E 期局部随机梯度下降 (SGD), 然后将更新后的w_t上传中心服务器. 最后, 在中心服务器上, 根据训练数据集的大小对w_t进行加权平均.

算法 1. 联邦加权平均算法

1. /*中心服务器*/
2. Initialize w_0
3. for each round $t = 1, 2, \cdots$ do
4. Select K clients
5. for each client $k \in K$ clients do
6. $w_t^k \leftarrow$ UpdateClient()
7. end for
8. $w_t \leftarrow \frac{\sum_{k=1}^{K} n_k \times w_{t-1}}{\sum_{k=1}^{K} n_k}$
9. end for
10. /*客户端服务器*/
11. UpdateClient(){
12. Initialize local minibatch size B, local epochs E, learning rate η
13. for each epoch $i \in E$ do
14. randomly sample S_i based on size B
15. $w_i \leftarrow w_{i-1} - \eta \nabla g(w_{i-1}; S_i)$
16. end for
17. return w_i}

2.4 锚定协议

锚定协议规定客户端定时用收集到的监控数据构建默克尔树, 并将默克尔树根节点锚定到区块链上, 以维护客户端数据的完整性, 并支持审计 (即解决中心组织和客户组织之间的争议). 同时, 锚定协议实现了激励机制, 并将客户组织的贡献情况记录在智能合约 (奖励注册表) 中.

算法 2 描述了锚定协议的工作原理. 客户数据锚定协议规定, 在每个锚定周期p_i(1 h), 每个客户组织c_j查询锚定周期内收集的客户数据d_{pn}^c, 将每条数据记录d_{pn}^c转换为字符串格式并进行哈希运算, 然后将哈希值用于构建默克尔树 (其中每个叶子节点表示一条数据记录的哈希值).

算法 2. 锚定算法

1. for each period $p_i = 1, 2, \cdots$, do
2. for $c_j \in C$ do
3. for d_{pn}^c do
4. $d_{pn}^c \leftarrow String(d_{pn}^c)$
5. $d_{pn}^c \leftarrow Hash(d_{pn}^c)$
6. $pNodeList[0..N] \leftarrow d_{pn}^c$
7. end for
8. repeat
9. for ($k = 0$; $k <$ *length*(*pNodeList*); k+2) do
10. $l \leftarrow pNodeList[k]$
11. if $k =$ *length*(*pNodeList*)then
12. $r \leftarrow l$
13. else
14. $r \leftarrow pNodeList[k + 1]$
15. end if
16. $r \leftarrow SHA256(l + r)$
17. $temp[j] \leftarrow r$
18. (each $j \in [0, \lfloor \frac{d}{2} \rfloor]$)
19. end for
20. $pNodeList[0..j] \leftarrow temp[0..j]$
21. until *length*(*pNodeList*)= 1
22. $root \leftarrow pNodeList[0]$
23. $time \leftarrow$ current time
24. *UpdRootSC*(*time*, *root*)
25. end for
26. end for
27. for each *training* round do
28. model *training*
29. *UpdStaSC*(*rNo*, *dataSize*)
30. *CalIncentiveSC*()
31. end for

在构造默克尔树的过程中, 该协议初始创建 *pNodeList* 存储父节点, 创建 *temp* 存储两个子节点拼接后哈希计算的结果. 首先, 将所有叶节点值存储在 *pNodeList* 中, 然后顺序将左子节点赋值 *l*, 右子节点赋值 *r*, 将两个子节点值连接后进行哈希计算得到父节点值, 并将其插入 *temp*. 重复此步骤, 直到遍历完 *pNodeList* 中的所有元素后, 将 *temp* 赋值 *pNodeList*. 循环执行这一过程, 直到 *pNodeList* 的长度变为 1, 最终, *pNodeList* 中的唯一值就是默克尔树的根节点.

UpdRootSC() 将默克尔树的根节点和时间戳上传到智能合约 (根节点注册表) 中. 当客户端服务器完成

训练, 并将更新后的模型参数发送到中心服务器后, *UpdStaSC*() 将联邦学习此轮训练的序列号 *rNo*、每个客户端参与本地训练的数据集大小上传到智能合约 (奖励注册表) 中. 然后计算相应的代币奖励, 并通过 *CalIncentiveSC*() 奖励给客户组织. *UpdStaSC*() 和 *CalIncentiveSC*() 都编码在智能合约 (奖励注册表) 中.

2.5 激励机制

算法 3 描述了激励机制工作流程. 该激励机制旨在根据客户组织的贡献 (用于模型训练的数据集大小), 奖励客户组织相应的代币. 在联邦学习的每一轮训练中, 中心服务器选择固定数量的客户端服务器, 并将它们的区块链地址存储在 *Clist* 中. 此外, 中心服务器还定义了一个结构 *training*, 来存储客户端服务器完成的训练任务, 包括训练任务状态, 训练集大小. 所选的客户端服务器在本地训练模型, 并将相应的信息上传到 *training* 中, 然后将这些信息另存为 *contri*. 最后, 根据 *contri* 中的数据, 将代币分发到 *Clist* 中的每个客户端服务器地址. 奖励代币的数量是训练集大小与常数 *C* 的乘积的总和. 这种激励机制的最终目标是鼓励更多的客户服务器参与联邦学习, 以消除模型偏差.

算法 3. 激励计算和分配算法

```
1. /*中心服务器执行*/
2. Clist ← selected clients address
3. /*客户端服务器执行*/
4. struct {finished,dataSize } training
5. for each training round do
6.     if address ∈ Clist then
7.         model training
8.         UpdStaSC(rNo, dataSize)
9.         CalIncentiveSC()
10.    end if
11. end for
12. /*上传训练任务状态 */
13. UpdStaSC(rNo, dataSize){
14.    training.finished ← true
15.    training.dataSize ← dataSize
16.    contri[addr][rNo] ← training
17. }
18. /*计算奖励*/
19. CalIncentiveSC(){
20.    if address ∈ Clist then
21.        if contri[addr][rNo].finished == true then
22.        token[addr] = token[addr]+ contri[addr][rNo].dataSize*C
23.    end if
24. end if
25. }
```

3 实验分析

实验将系统架构应用于一个真实的用例, 并从可行性、准确性和性能方面来评估架构.

3.1 用例

工厂轴承是否故障和确定故障准确位置直接影响到工厂的生产效率. 实验将轴承的使用状况作为用例, 将确定轴承的故障位置作为实验目标.

对于工厂来说, 各种设备参数都具有一定的商业敏感性. 因此, 在实验中, 大部分机器学习任务被分配给安装在工厂的服务器, 以保护工厂的数据隐私. 此外, 该实验旨在鼓励工厂提供轴承故障数据, 并在本地训练模型, 然后根据每个厂的贡献来奖励他们代币, 这些代币日后可以用来购买新产品或服务.

轴承作为工厂大型生产制造机器中的关键部件, 仅仅知道轴承出现故障已不能满足高效生产的要求. 本研究开发了一个轴承故障检测系统架构.

3.2 实现方法

图 2 显示了客户数据锚定和激励机制的智能合约, 根节点注册表用来存储默克尔树根节点和对应的时间戳; 奖励注册表用来存储某一联邦学习轮数, 和这一轮联邦学习中, 每个客户机的训练状态, 客户数据大小以及获得的代币数目.

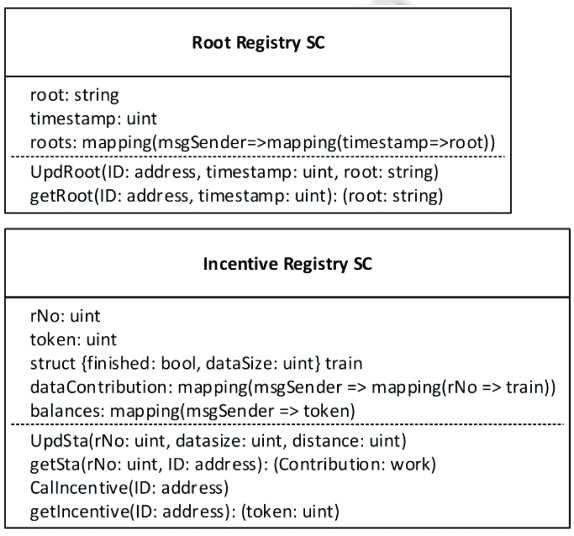


图 2 链上数据结构

实验基于联邦学习框架 Leaf[6] 实现, 并进行了相应改进, 使用以太坊作为底层区块链网络, 采用工作量共识算法 (PoW), 难度配置为 0x4000, 使用 Solidity 语言编写智能合约, 并使用 Solid Compiler 0.4.20 版本进

行编译. 客户端数据锚定器组使用 Java 1.8 开发, 激励机制使用 Python3.6 实现, 选择 MySQL 5.7.25 作为客户端数据库来存储链下数据.

实验将系统架构部署在 6 台阿里巴巴云服务器 (2 VCPU, 8 GB RAM) 上, 其中 1 台云服务器作为中心服务器, 5 台云服务器作为客户端服务器. 用 5 块树莓派 3(系统配置 Raspian 2018-11-3) 模拟 5 台客户端设备. 假定每个客户端服务器被分配到一个工厂, 工厂中有轴承部件 (客户端设备). 每个轴承传感器采集 8 个信号.

实验中, 我们选取转速系统负载设置为 20 Hz-0 V 的轴承的工作状态作为数据集[7], 每条数据记录由上述的 8 个特征和 1 个标签组成, 标签表示故障发生位置 (分别为无故障, 内环和外环有裂痕, 滚珠有裂痕). 一个客户端服务器有一个分类器, 它是一个 4 层神经网络. 第 1 层由 8 个单元组成, 对应 8 个特征, 然后, 有两个隐藏层, 分别包含 150 个单元, 使用 ReLU 函数作为激活函数, 输出层为 Softmax 函数, 用于对非故障和故障位置进行分类. 学习率设置为 0.005.

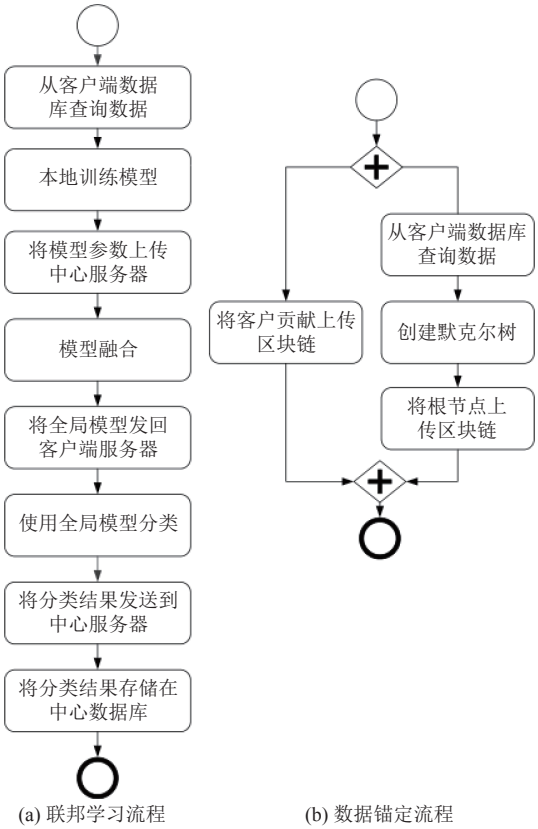
3.3 可行性

图 3 展示了系统架构中, 联邦学习和区块链锚定的工作流程, 实现了第 2 节中提出的功能性和非功能性需求. 功能性需求方面实现了: (1) 架构中, 每个工厂的客户端服务器能够在本地训练模型, 中心服务器将本地模型融合为全局模型, 然后进行故障检测. (2) 工厂操作者通过对比存储在区块链上的默克尔树根节点, 以验证某一时期内客户数据的完整性. (3) 此外, 通过智能合约 (奖励注册表), 工厂获得代币奖励.

非功能性需求方面实现了: (1) 应用联邦学习保护工厂数据的隐私, 中心服务器只需要获得每次更新后的模型参数, 不需要获得所有的轴承使用信息. (2) 在训练数据规模较大的情况下, 在带宽开销上, 联邦学习比传统的模型训练更有优势. (3) 全局模型实现了良好的检测精度. (4) 通过锚定协议维护轴承数据的完整性. 当需要对数据进行审计时, 可以将存储在每个工厂本地的数据重新构建为默克尔树结构, 并与存储在区块链上的根节点进行比较来验证. 每个工厂和轴承制造商都有一个完整的节点, 以保证区块链基础设施在某些节点可能宕机的情况下正常运行.

3.4 定量分析-精度和性能

在定量分析中, 每个客户端服务器存储约 1000 条训练数据和 1000 条测试数据. 每个客户端服务器在本地数据集上训练模型, 中心服务器通过融合本地模型参数, 生成全局模型. 我们在每个单独的客户机服务器上用本地测试集测试故障检测全局模型的准确性. 此外, 我们还比较了联邦学习的全局模型与集中式学习的模型的准确性.



(a) 联邦学习流程 (b) 数据锚定流程

图 3 联邦学习流程和数据锚定流程

3.4.1 区块链锚定时间测量实验

锚定协议的执行时间包括从数据库查询数据, 创建默克尔树, 将默克尔树的根节点和客户端贡献数据上传区块链. 图 4 展示了上传默克尔树根节点和客户贡献数据到区块链的时间, 执行锚定协议消耗了大部分时间. 我们测量了交易写入时间, 该时间用于将智能合约交易写入块中. 锚定默克尔树根节点的时间约为 18 s, 而锚定客户端贡献数据的平均编写时间约为 21 s. 区块链上的写入延迟比传统数据库中要长得多, 因为传播交易/块和各节点达成一致需要时间. 由于区块链用于审计和实现激励机制, 因此在本工作中不考虑区块链的写入延迟. 我们还检查了默克尔树的创建时间, 大约需要 250 ms. 由于默克尔树的创建时间比交易写入块的时间少得多, 因此在图 4 中没有包含它.

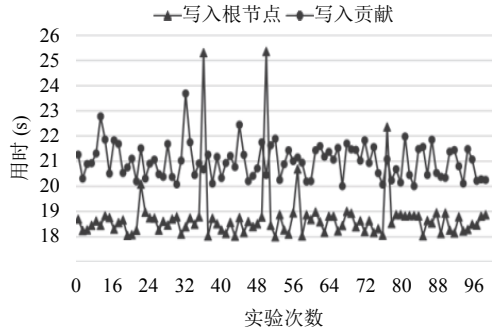


图 4 数据锚定时间

3.4.2 故障检测实验

在 100 轮训练中, 我们分别测量了联邦学习和集中式学习的模型精度. 对于每一轮联邦学习, 模型在每个客户机上进行 40 期的本地训练, 然后将更新后的模型参数发送到中心服务器用以生成一个全局模型. 每个客户端使用全局模型进行故障检测. 对于每一轮集中式学习, 一个中央服务器将模型训练 40 期, 每个客户端使用这个模型来检测故障. 与在本地数据集上训练的联邦学习模型相比, 集中式训练模型是由所有客户端的本地数据集组成的数据集上训练得到的. 实验中, 我们使用相同的测试集测试了集中式学习模型和联邦学习模型.

测试结果如图 5 所示. x 轴表示训练的次数, y 轴表示精度. 在图 5 中, 每个子图包含 2 条线, 显示了在不同测试集上, 不同学习方法的模型精度. 总体而言, 从图 5 可以看出, 联邦学习模型可以取得和集中式学习模型相同的的精度.

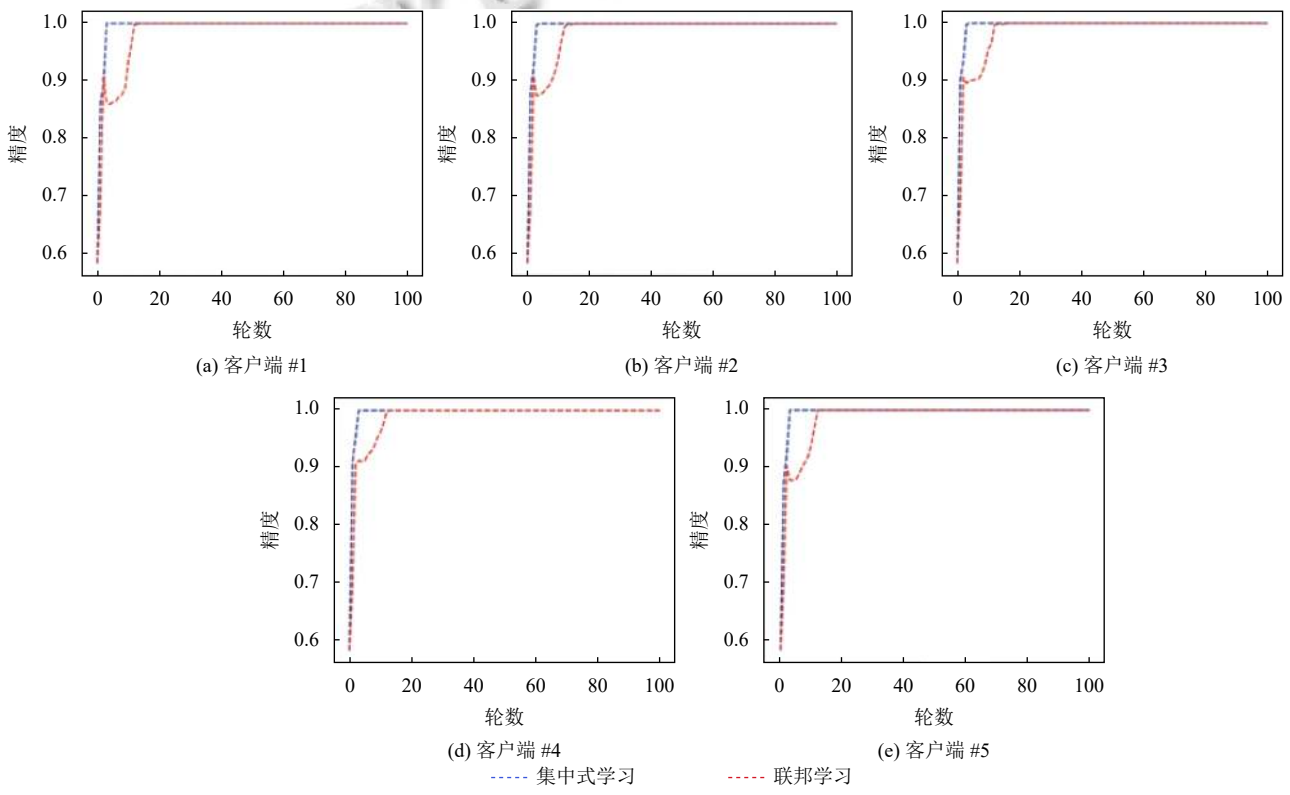


图 5 联邦学习模型与集中式学习模型精度

3.4.3 通信开销测量实验

我们测量了联邦学习 (FL) 和集中式学习 (CL) 的通信开销. 采用数据集大小分别为 1 KB, 1 MB, 1 GB. 实验结果显示: (1) 在集中式学习中, 所有的数据都需要从客户端服务器发送到中心服务器, 因此, 集中式学习的通信开销与数据集大小一样, 通信开销在 3 个数据集下, 分别为 1 KB, 1 MB, 1 GB. (2) 在联邦学习测量实验中, 我们用 FL(N) 表示每轮随机选择 N 个客户端来训练模型, 测量了 100 轮联邦训练的通信开销, 每轮训练中测量了模型参数上传中心服务器和全局模型发送回客户端的通信开销. 对于不同数量客户端, 只有模型参数被发送到中央服务器, 通信开销与数据集大小

无关, 具体结果显示使用联邦学习, 客户端取值 N 为 1 至 5 时, 通信开销分别为 15 KB, 30 KB, 45 KB, 60 KB 及 75 KB.

可以得到结论, 集中式学习的通信开销要比联邦学习大得多, 当数据集非常大时, 采用联邦学习可以显著降低通信开销.

4 结论与展望

本文提出了一种基于区块链的联邦学习系统架构. 在架构中, 一个中心服务器协调多个客户端服务器训练一个用于分类的全局模型, 并将原始数据存储在本地. 该架构采用区块链来实现数据的可验证完整性和激励机制.

实验中, 我们在一个真实的用例中实现了所提出的系统架构, 并评估了该方法的可行性、模型精度和性能. 实验结果表明该架构实现了文中提出的全部目标.

在本文中, 大部分联邦学习任务是在客户端服务器上完成的. 在接下的研究工作中, 我们计划将所有训练任务从客户端服务器移到具有更强大计算和存储能力的客户端设备上. 此外, 我们计划进一步探讨如何提高客户端设备在模型融合过程中的可信度. 再者, 我们计划在未来的研究中将激励机制与更多的因素关联.

参考文献

1 IDC. IoT growth demands rethink of long-term storage strategies, says IDC. https://www.idc.com/getdoc.jsp?containerId=prAP46737220. [2020-7-28].

2 Seshadri SS, Rodriguez D, Subedi M, *et al*. IoTCop: A blockchain-based monitoring framework for detection and isolation of malicious devices in Internet-of-Things systems. IEEE Internet of Things Journal, 2021, 8(5): 3346–3359. [doi: 10.1109/JIOT.2020.3022033]

3 Lo SK, Lu QH, Wang C, *et al*. A systematic literature review on federated machine learning: From a software engineering perspective. arXiv: 2007.11354, 2020.

4 Kairouz P, McMahan HB, Avent B, *et al*. Advances and open problems in federated learning. arXiv: 1912.04977, 2019.

5 Kim H, Park J, Bennis M, *et al*. Blockchained on-device federated learning. IEEE Communications Letters, 2020, 24(6): 1279–1283. [doi: 10.1109/LCOMM.2019.2921755]

6 Caldas S, Duddu SMK, Wu P, *et al*. LEAF: A benchmark for federated settings. arXiv: 1812.01097, 2018.

7 Shao SY, McAleer S, Yan RQ, *et al*. Highly accurate machine fault diagnosis using deep transfer learning. IEEE Transactions on Industrial Informatics, 2019, 15(4): 2446–2455. [doi: 10.1109/TII.2018.2864759]