计算机科学
COMPUTER SCIENCE

FL_Raft:基于联邦学习模型的选举共识方案

荣宝俊, 郑朝晖

引用本文

荣宝俊, 郑朝晖. FL_Raft:基于联邦学习模型的选举共识方案[J]. 计算机科学, 2023, 50(11): 364-373.

RONG Baojun, ZHENG Zhaohui. FL_Raft:Election Consensus Programme Based on Federated Learning Model [J]. Computer Science, 2023, 50(11): 364-373.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.221000134

FL_Raft:基于联邦学习模型的选举共识方案

荣宝俊[1]　郑朝晖[1,2]
1 苏州大学计算机科学与技术学院　江苏 苏州 215006
2 苏州大学网络空间安全工程实验室　江苏 苏州 215006
(20204227067@stu.suda.edu.cn)

摘　要　针对异构集群中 Raft 共识算法的投票分裂和领导者频繁更换造成的吞吐量低、共识时延高和安全性低等问题,提出了一种基于联邦学习模型的 Raft 选举共识方案 FL_Raft。首先,联邦学习聚合运行于每轮领导者迭代后,调用节点的本地特征数据,通过联邦学习训练模型筛选高性能节点组;其次,建立基于行为的权益计算模型,对集群中每个节点的权益值进行动态调整;最后,建立权益选举模型,由队列选举准领导者节点,选取出的节点经全体节点投票选举后成为最终领导者节点。实验结果表明,在保证各节点数据隐私性的前提下,相比 Raft,FL_Raft 的选举时延降低了 50%,领导者可靠性达到 95%以上,共识时延缩短了 20%,吞吐量提高了 13%。FL_Raft 共识算法保证了选举的效率和安全,提高了集群的稳定性和领导者的可用性。

关键词:共识算法;联邦学习;模型选举;数据隐私;异构集群
中图法分类号　TP399

FL_Raft:Election Consensus Programme Based on Federated Learning Model

RONG Baojun[1] and ZHENG Zhaohui[1,2]
1 School of Computer Science and Technology,Soochow University,Suzhou,Jiangsu 215006,China
2 Cyberspace Security Engineering Laboratory,Soochow University,Suzhou,Jiangsu 215006,China

Abstract　Aiming at the problems of low throughput,high consensus delay and low security caused by vote splitting and frequent leader change of Raft consensus algorithm in heterogeneous clusters,a Raft election consensus programme FL_Raft based on federated learning model is proposed. First,federated learning aggregation runs after each leader iteration,invokes the local characteristic data of nodes,and filters high-performance node groups through the federated learning training model. Secondly,a behavior-based equity calculation model is established to dynamically adjust the equity value of each node in the cluster. Finally,the equity election model is established to elect the quasi leader node,which becomes the final leader node after all nodes vote. Experimental results show that under the premise of ensuring the data privacy of each node,compared with Raft,the FL_Raft election delay reduces by 50%,the leader reliability is more than 95%,the consensus delay reduces by 20%,and the throughput increases by 13%. The FL_Raft consensus algorithm ensures the efficiency and security of the election,and improves the stability of the cluster and the availability of leaders.

Keywords　Consensus algorithm,Federal learning,Model election,Data privacy,Heterogeneous cluster

1　引言

区块链[1-2]如今是分布式应用的大方向,它融合了 P2P 传输[3]、加密算法[4]、共识算法[5]、分布式账本[6]等技术,在去中心化的环境中让所有节点参与并维护共享的数据。共识算法在解决区块链系统中各个节点日志内容一致性问题的同时,也使其具备一定的容错能力。根据应用场景可将区块链分为公有链[7]、私有链[8]和联盟链 3 类[9],每一种链中都有适用的共识算法,如公有链的 PoW(Proof of Work)[10]和 PoS(Proof of Stake)[11],私有链的 Paxos[12-13]和 VR(Viewstamped Replication)[14],联盟链的 PBFT[15],DPoS[16]和 Raft[17]。

良好的共识算法可以解决区块链可用性、共识时延、吞吐量和安全性等问题。自从 2013 年 Ongaro 等[17]提出了 Raft 共识算法后,它的易理解和易实现的特点迅速代替 Paxos 成为主流共识算法,数据吞吐量与 Multi-Paxos[18]不相上下。许多分布式应用也采用 Raft 作为其底层共识算法,如联盟链 Quorum[19],redis[20]和 etcd[21]等。

到稿日期:2022-10-15　返修日期:2023-03-10
基金项目:江苏省高校自然科学研究项目(19KJA550002);江苏高校优势学科建设工程资助项目
This work was supported by the Natural Science Research Project of Colleges and Universities in Jiangsu Province(19KJA550002) and Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.
通信作者:郑朝晖(zhengzh@suda.edu.cn)

Raft 共识算法属于非拜占庭容错类型，它具备较高的数据吞吐量，同时易于实现和改进。Raft 算法的领导者是集群中不可或缺的节点，通常情况下只有唯一一个，它需要接收用户的日志添加请求，并且让集群中的其他节点就日志消息达成共识，因此它成为了影响共识效率的重要因素。

目前 Raft 共识算法仍然存在一些挑战：

1）投票分裂现象。Raft 中会经常出现多个候选者同时发起投票而平分投票后无法选出领导者的情况，这会增加任期并重新发起投票过程，造成选举时延过高。

2）领导者的可靠性无法得到保证。Raft 集群中节点的性能存在差异，其选举和时间因素相关，具有随机性，若选出的领导者性能较差，则会产生网络分区，甚至出现突然下线的问题。

3）数据隐私问题。节点数据隐私目前变得越来越重要，当集群安全性较低时，一旦节点数据泄露，将造成十分严重的后果。

机器学习常被用来预测结果与数据分类，将分布式与机器学习结合也是目前的研究热点。联邦学习[22]作为机器学习的一种方式，不仅可以在分布式环境中训练模型，还可以保护客户端的数据隐私和安全。在联邦学习环境中，定义存在 N 个数据的所有者 $F_1,\cdots,F_N$，它们拥有各自的数据 $D_1,\cdots,D_N$，假设中心式机器学习使用 $D=D_1\cup\cdots\cup D_N$ 训练出一个模型 M_{SUM}，准确度为 V_{SUM}。联邦学习所有数据拥有者协作训练一个模型 M_{FED}，其中所有数据的所有者 F_i 都不会泄露其数据 D_i，M_{FED} 的准确度用 V_{FED} 表示，其大小接近 M_{SUM}。σ 为非负的精度损失，联邦学习达到 σ 精确损失，如式(1)所示：

$$|V_{FED}-V_{SUM}|<\sigma \tag{1}$$

联邦学习的横向架构是由多个客户端$\{\alpha_1,\alpha_2,\cdots,\alpha_n\}$和一个服务端 β 组成，单个客户端 α_i 负责根据本地的数据 D_i 训练第 j 轮次的模型 M_i^j，并将梯度 G_i^j 加密成 $ECD(G_i^j)$上传至服务端 β，服务端 β 负责将所有的加密梯度进行聚合平均 $F_{avg}\{ECD(G_1^j),ECD(G_2^j),\cdots,ECD(G_n^j)\}$，来获取全局模型 M_i^G，之后返回加密的全局模型参数 $ECD(M_i^G)$给每个客户端$\{\alpha_1,\alpha_2,\cdots,\alpha_n\}$，在这个过程中服务端 β_{honest} 必须保证是信任节点，且客户端本地数据 D_i 不参与网络传播。联邦学习的应用也很广泛，常见于与边缘计算[23]、区块链[24]、迁移学习[25]等技术的融合。

通过对 Raft 共识算法和联邦学习进行分析，本文提出了基于联邦学习的模型选举共识方案——FL_Raft（Federal-learning-Raft），该方案通过优化领导者选举过程，来实现对 Raft 共识算法的改进。

本文的主要工作如下：

1）设计了在 Raft 集群中的联邦学习过程，包括数据集收集与处理、模型训练、模型聚合和评估方案，确保了节点数据的隐私安全，并通过实验证明了该过程的可行性。

2）提出了准领导者选举过程，采用联邦学习训练的模型进行模型选举，新增权益选举来提高选举的正确率，减少去中心化程度，筛选后的准领导者节点通过投票选举成为领导者，减少投票分裂的概率，并提高领导者的可靠性，防止频繁下线问题的发生。

3）通过对 FL_Raft 的安全性、一致性和活性的算法分析来证明其可行性。并从 FL_Raft 的选举时延、可用性、领导者可靠性、共识时延、吞吐量和联邦学习模型的准确率这 6 个方面，通过对比实验验证了 FL_Raft 共识算法的有效性。

2 相关工作

目前有许多针对 Raft 领导者选举部分的研究方案，Wang 等[26]提出结合区块链网络层的双层 Kademlia 路由协议，改进了 Raft 算法中 Leader 节点的选举过程，并将其命名为 K-Raft，但是未考虑引入 K 桶带来的安全性问题。Huang 等[27]设计了 Raft 集群内部成立 PBFT 算法的委员会，用于改进选举过程，使其适用于拜占庭环境，然而该算法的复杂度较高，不适用于网络质量较差的环境。Zhan 等[28]开发了一种随机选择算法，减少了参与投票的节点数，降低了选举时延，然而该算法难以分析运行时间，无法获得错误解决方案的概率。Wei 等[29]在领导者选举过程中加入候选者票数转让机制，缓解了票数分裂问题，最终缩短了选举时间，但这会影响去中心化的程度。

针对领导者选举部分的改进方法各异，近年来机器学习发展迅速，联邦学习结合区块链共识的 BFL 框架的研究层出不穷。Chen 等[30]提出了基于联盟链的联邦学习分布式计算架构，通过去中心化的分布式训练模式和模型压缩来改进 PBFT 算法。Li 等[31]提出了一种具有委员会共识的基于区块链的去中心化联邦学习框架，减少了共识计算量，减小了恶意攻击的概率。Shayan 等[32]提出了一种完全分散的对等多方联邦学习方法，使用区块链和加密技术来协调对等客户端之间的训练数据隐私保护过程。

文献[33]利用联邦学习来评估 Raft 中每个节点的网络稳定性，选取更稳定的节点作为下一个任期的领导者，减小了网络分裂的概率，但该方法仅利用联邦学习改进了网络分裂的情况，并没有充分地利用联邦学习学习其他特性，也未对算法进行安全性分析和改进。

通过以上分析，我们认为 Raft 共识算法与联邦学习结合的方式有以下几个方面需要改进：1）数据集的规范化，收集、存储和处理需要进行统一化处理；2）模型训练和聚合方法的可适配，训练模型、误差函数以及聚合函数的选择需要结合 Raft 共识的特征；3）选举结果的去中心化，减少选举过程中节点之间相互影响的程度。为此，我们设计了 FL_Raft 方案，第 3 章将详细讲解 FL_Raft 方案针对以上 3 点进行的改进。

3 算法设计

3.1 FL_Raft 方案的结构

FL_Raft 共识方案在原有的 Raft 算法的基础上加入了联邦学习模型和权益选举过程，同时添加准领导节点角色，将其作为跟随者成为领导者的过渡节点。当集群中没有领导者时，准领导者改变状态为领导者，以保证集群的稳定运行。算法设定领导者节点为联邦学习服务端，当领导者存在时才会开始联邦学习过程，以防止联邦服务端单点故障问题的发生。

FL_Raft 共识方案的状态转换图如图 1 所示。

准领导者节点的选举行为是由跟随者主动发出的，当集群中不存在准领导者节点时，跟随者会进行模型选举，该过程并不需要领导者的干预，因此即使领导者突然下线，模型选举过程也能够稳定运行。为了防止准领导者选举超时，FL_Raft 保留了原 Raft 的候选者投票选举过程。

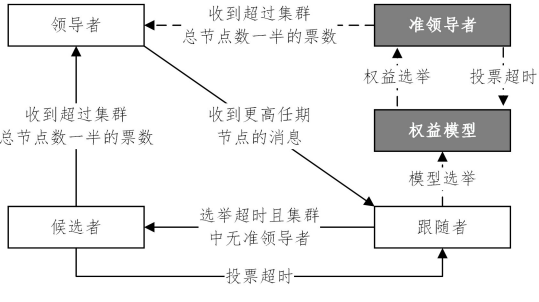

图 1 FL_Raft 共识方案的状态转换图

Fig. 1 FL_Raft's consensus programme state transition

3.2 FL_Raft 联邦学习模块

联邦学习的目的是通过收集集群中每个节点的特征数据，训练出可预测领导者功能的模型。联邦学习的服务端为领导者节点，客户端为集群中正常运行的非领导者节点，采用的训练架构为横向联邦学习。其过程包含 3 个步骤：数据集的收集与处理、客户端模型训练、服务端聚合并广播模型。

3.2.1 数据集的收集与处理

数据集是联邦学习模型训练的基础，它们只保证在节点本地使用，而不参与通信过程。通常它需要包含以下 3 个关键特征。

1) No-IID：客户端训练数据通常是基于特定节点自身的属性特征，因此任何特定节点的本地数据集都无法代表集群整体的分布。

2) 不平衡：每个客户端的运行时间和性能并不相同，从而会导致本地训练数据的数据不均衡。

3) 通信约束：通常来说，节点会存在离线状态，或处于缓慢和昂贵的连接状态，其更多地适用于通信复杂度较低的环境。

通过分析以上 3 点，我们设计了 FL_Raft 数据集的收集、存储、处理及使用方法。

1) 收集。设定节点在较小的时间间隔 $T_{\text{collection}}$，以 $f_{\text{collection}}=1/T_{\text{collection}}$ 的频率记录单条数据。假设当前时间点为 t，客户端编号为 i，则该条记录为 D_{i_t}，全部的记录共同构成客户端 i 的特征数据集 D_i。其中属性值包括任期(Term)、最新的日志索引(Log_Index)、运行时长(Run_Time)、投票时延(Voting_Delay)、状态类型(State)等，除日志消息等隐私数据外，任何与节点相关的属性都可以被收集。

2) 存储。考虑数据集由时间点和特征属性两个维度构成，FL_Raft 设定以时间点 t 与属性值 A 构成二维矩阵 $\mathbb{E}$，E_i 则为单个客户端需要持久化存储的数据集。随着集群的运行和数据的变化，较早时间点的数据对模型的优化作用较小，需要定期删除。

3) 处理。由于集群中节点的状态不断变化，某个时间点仅会记录少量的属性值，而未记录部分以零值替代，故 E_i 普遍为稀疏矩阵。为节约存储空间考虑采用三元组法，对于非零元素 v_{xy}，以 (v,x,y) 构成的三元组存储在顺序表 L_i 中，其中 v 是元素值，x 是 v 所在的行值，y 是 v 所在的列值。

4) 使用。在模型训练前，将 L_i 重构为二维数组 $\mathbb{E}_i$，然后对零值进行预处理，如式(2)所示：

$$v'_{x,y}=\begin{cases} v_{x,y}, & |v_{x,y}|\neq 0 \\ \dfrac{\sum\limits_{k=1}^{X} v_{k,y}}{\sum\limits_{k=1}^{X} f(v_{k,y})+1}, & |v_{x,y}|=0 \end{cases} \tag{2}$$

其中，$v'_{x,y}$ 为处理后的元素值，f 为计数函数(非零值计为 1)，x 为数据集总数。经处理后的数据集 $\mathbb{E}'$ 可适用于模型训练过程，这种做法可提高模型训练的准确率。

3.2.2 客户端模型训练

模型训练发生在客户端本地，仅使用数据集参与训练，并不会在网络中传输任何可能泄露客户端隐私的数据。模型采用逻辑回归，使用 sigmoid 函数式(3)将预测值压缩在区间[0,1]内，其中预测标签为 1 的概率即为 sigmoid 函数的值：$P_{y=1}=\sigma(z)=p$，预测标签为 0 的概率则为 $P_{y=0}=1-p$。

$$\sigma(z)=\frac{1}{1+e^{-w^{T}x}} \tag{3}$$

使用最大似然作为损失函数式(4)，其中 P_{SUM} 为样本概率。

$$F(w)=\ln(P_{\text{SUM}})=\sum_{n=1}^{N}(y_n\ln(p)+(1-y_n)\ln(1-p)) \tag{4}$$

使用 SGD 随机梯度下降算法式(5)优化模型。

$$\theta:\theta+a(y_n-p)x_n;p=\frac{1}{1+e^{-w^{T}x}} \tag{5}$$

为减少通信次数，FL_Raft 增加了每个客户端的计算量，即在每轮通信之间，每个客户端进行多次的本地更新参数计算，并且与每一轮服务器参数更新只需要一次客户端本地更新的 FedSGD 算法进行对比。联邦学习训练的模型精度可以根据集群节点的总体性能来控制，计算能力越强，模型的收敛速度越快，同时联邦学习所使用的训练集大小可以根据集群环境调整，如某一数据量下模型的准确率已经很高，不用再增加数据量来进行训练。

3.2.3 模型聚合

服务端接收来自客户端的模型更新参数 $f_i(w)=f(x_i,y_i,w)$ 后，将会进行聚合过程，FL_Raft 使用最常用的 FedAvg 算法，如式(6)所示：

$$\min_{w\in\mathbb{R}^d} f(w) \text{ where } f(w)\overset{\text{def}}{=}\frac{1}{n}\sum_{i=1}^{n} f_i(w) \tag{6}$$

假设有 K 个客户端参与了聚合过程，其中 $\mathscr{P}_k$ 是客户端 k 上的数据集，其中 $n_k=|\mathscr{P}_k|$，由此式(6)改写为：

$$f(w)=\sum_{k=1}^{K}\frac{n_k}{n}F_k(w) \text{ where } F_k(w)=\frac{1}{n_k}\sum_{i\in\mathscr{P}_k} f_i(w) \tag{7}$$

其中，n 表示参与聚合的数据数量，w 表示模型当前的参数，F_k 表示最大似然损失函数。FL_Raft 在聚合后会得到一个待验证的全局模型 $f(w)$，服务端即领导者节点使用本地数据集对模型进行评估，以此决定是否继续训练模型，若评估

合格，则向每个客户端广播全局模型 $f(w)$，否则继续新一轮的训练。

综上所述，联邦学习的执行步骤如算法 1 所示。

算法 1 联邦学习算法

输入：初始化全局参数 $\mathcal{G}$，收集时间间隔 $T_{collection}$

输出：新的联邦学习模型 f(w)

1. / * 客户端模型更新过程 * /
2. / * 以 $f_{collection}=1/T_{collection}$ 频率收集并存储数据 * /
3. $\mathcal{N}_i$. Collect_Data($T_{collection}$)；
4. / * 读取本地联邦学习训练所需数据集 * /
5. pre_data=Read_FL_Dataset($\mathcal{N}_i$)；
6. / * 预处理节点本地数据集 * /
7. D_i=Pre_Node_Dataset(pre_data)；
8. $p \leftarrow \frac{1}{1+e^{-w^T x}}$；
9. $F_i(w) \leftarrow \sum_{n=1}^{N}(y_n \ln(p)+(1-y_n)\ln(1-p))$；
10. for each round t do
11. $f_i^{(t)}(w)=f_{i-1}{}^{(t)}(w)+a(y_n-p)x_n$；
12. end for
13. $F_i^{(t)}(w) \leftarrow f_i^{(t)}(w)-f^{(t)}(w)$；
14. $n_i \leftarrow |D_i|$；
15. / * 返回 t+1 轮模型和数据集数量 $F_i^{(t)}(w)$，n_i * /
16. / * 服务端模型聚合过程 * /
17. for each round t do
18. $S^{(t)} \leftarrow$(all sample $\sum_{i=1}^{n}$client(i))；
19. for each client $i \in S^{(t)}$ in parallel do
20. $(F_i^{(t)}(w), n_i) \leftarrow$ Client_Update(i, $f^{(t)}(w)$)；
21. end for
22. $n \leftarrow \sum_{i \in S^{(t)}} n_i$；
23. $f^{(t+1)}(w) \leftarrow f^{(t)}(w)+\sum_{i \in S^{(t)}} \frac{n_i}{n} F_i^{(t)}(w)$；
24. end for
25. / * 返回聚合后的全局模型 $f^{(t+1)}(w)$ * /
26. / * 服务端广播并输出全局模型 f(w) * /

3.3 FL_Raft 的准领导者选举模块

Raft 共识算法中跟随者只能通过候选者才能成为领导者，这样的方式不仅非常局限，还会产生两大问题。一是投票分裂问题，多个候选者平分投票，迟迟无法选出领导者，导致一段时间内的集群不可用。二是选举出的领导者具有随机性，无法保证其性能是最优的，若其性能较差，则会出现频繁下线或网络分裂的问题，影响集群的稳定性。准领导者是 FL_Raft独有的一种节点类型，它是经过模型选举和权益选举的跟随者。准领导者的选举是经过模型的预测以及权益的严格筛选后得到的，代表当前集群中最有可能成为下一任领导者的节点，它具备领导者的条件，同时还拥有良好的性能，同时权益选举确保选举结果唯一、正确且具有一定的公平性。下面详细介绍 FL_Raft 的两种选举。

3.3.1 模型选举

跟随者是模型选举的发起者，这里的模型是经过联邦学习得到的。FL_Raft 中模型选举存在触发条件，即跟随者在接收新一轮的全局模型 $f(w)$ 后才可发起模型选举，这样做的目的在于保证跟随者当前的模型是最新状态。

模型选举的执行步骤如下：

1）接收来自领导者广播的全局模型 $f_g(w)$，将其覆盖自己本地的模型 $f_l(w)$。

2）读取本地联邦学习训练所需的数据集 D_i。

3）预处理本地联邦学习训练所需的数据集 D_i，将三元组顺序表 L_i 重构为二维数组 $\mathbb{E}_i$，并且对零值进行填充操作。

4）读取本地联邦学习的模型 $f_l'(w)$，使用预处理后的数据集 $\mathbb{E}_i'$ 执行模型函数。

5）若模型函数预测成为领导者的概率超过 50%，则将其放入权益选举等待队列 Q_p 中，否则放弃此次模型选举，重新执行步骤 1。

算法 2 描述了模型选举的过程。

算法 2 模型选举算法

输入：客户端节点 node

输出：Equity 队列 EQ

1. / * 节点接收领导者的全局模型 * /
2. f(w)←f_L(w)；
3. / * 更新本地模型 * /
4. $f_n(w) \leftarrow f(w)$；
5. / * 预处理数据集 * /
6. for x←1 to X do
7. for y←1 to Y do
8. if $|v_{x,y}| \neq 0$ then
9. $v'_{x,y}=v_{x,y}$；
10. else $v'_{x,y}=\frac{\sum_{k=1}^{X} v_{k,y}}{\sum_{k=1}^{X} f(v_{k,y})+1}$
11. end if
12. end for
13. end for
14. m←f_n(w)；
15. sum=0；
16. for i←1 to n do
17. / * 执行模型函数返回准确率 * /
18. e=Execute_Model(data[i],m)；
19. sum +=e；
20. end for
21. if sum/n >=0.5 then
22. / * 将该节点加入 Equity 队列 * /
23. EQ←node；
24. end if
25. / * 输出 Equity 队列 EQ * /

经过模型选举出的节点，有超过 50%的概率会成为下一任领导者，这类节点通常拥有良好的性能，并且节点的属性条件使其更适合成为领导者，若其成为领导者，可以大幅减少节点下线和网络分裂情况的发生。

3.3.2 权益选举

模型选举虽然可以得到性能较好的节点，但若是只通过这个方式选举领导者，则会存在两个问题：

1)若集群中拥有一批高性能服务器节点参与共识,这批节点很可能通过模型选举,若其成为领导者,那么去中心化程度和安全性将大大降低。

2)模型选举仍然存在一定的漏洞风险,若模型准确率较低或客户端使用伪造的模型,则选举出的节点并不一定符合领导者的要求,一旦当选领导者,就可能会危害集群的正常运行。

由此,FL_Raft 经过模型选举后会得到一个权益选举等待队列 Q_p,队列中的节点称为权益节点,它们需进行权益选举。为保证公平性和安全性,FL_Raft 假设权益选举的发起者为可信的第三方节点。权益选举分为三大部分:

1)权益分配。FL_Raft 将权益设为百分比等级,上限为 100%,代表权益最大值,下限为 0%,代表权益最小值,若一个节点的权益为 0%,则它失去了成为领导者的资格。初始时节点权益为 50%。

2)权益变更。FL_Raft 将根据节点的属性和行为对权益进行变更,假设当前节点权益值为 $p\in[0\%,100\%]$,详细变更情况如表 1 所列。

表 1 节点属性和行为的权益变更

Table 1 Equity changes in node attributes and behaviors

分类	说明	实施
属性	任期不是最新	从队列 Q_p 中删除,$p=p-5\%$
	日志条目不是最新	从队列 Q_p 中删除,$p=p-5\%$
	本地模型轮次 R_{now} 不是最新 R_{new}	从队列 Q_p 中删除,$p=p-(R_{new}-R_{now})*10\%$
行为	上一任期为领导者	$p=p-20\%$
	参与过最新一轮的联邦学习过程	$p=p+5\%$
	未存在上述情况	p 保持不变

注:若节点在权益变更中权益值减小为 0%,则需从队列 Q_p 中删除,这里的最新均指权益队列中的最大值。

3)权益排序。Q_p 中剩余节点根据权益值从大到小排序,排序方式为稳定排序,当权益值一致时,最早加入队列的节点将会排在前面。最终排序第一的节点成为准领导者节点。

算法 3 描述了权益选举的过程。

算法 3 权益选举算法

```
输入:权益 Equity 队列 Qp,最新领导者 Lnew,最新联邦学习节点列表
      LFL
输出:准领导者 Lq
1. /* 初始化权益 */
2. Qp^(0)[R]←0.5;
3. for each round t do
4.   /* 获取最新任期和最新日志索引 */
5.   Tnew←max(Qp^t[T]);
6.   Lognew←max(Qp^t[L]);
7.   for each node n∈Qp^t in parallel do
8.     /* 获取节点权益和模型轮次 */
9.     p←n[R];r←n[MR];
10.    if n[T]≠Tnew || n[L]≠Lognew then
11.      /* 从 Equity 队列中删除节点 n */
12.      delete(n,Qp^t);
13.      p←p−0.05;
14.    else if n==Lnew then p←p−0.02;
15.    else if r≠t then p=p−(r−t)×0.1;
16.    else if n in LFL then p=p+0.05;
17.    end if
18.  end for
19. end for
20. /* 根据权益值对 Equity 队列进行稳定排序 */
21. StableSort(Qp^t);
22. /* 最大权益值节点即准领导者 */
23. Lq←max(Qp^t[E]).
```

上述过程会将任期、日志条目和本地模型不是最新的节点从选举队列中删除;减少上一任领导者的权益值,使其余节点有当选领导者的机会。权益选举一方面筛选了不符合领导者要求的节点,另一方面防止高性能节点频繁成为领导者,从而影响去中心化程度,保证了选举的正确性和公平性。需要说明的是,若本地模型不是最新的,则其节点权益值会减小,参与最新联邦学习的节点权益值增加,目的都是提高节点进行联邦学习的积极性。

3.4 FL_Raft 方案的选举流程

FL_Raft 区别于 Raft 最重要的一点就是加入了准领导者选举过程,与候选者直接投票选举方式不同,准领导者选举先通过模型选举和权益选举来筛选实现,目的在于改善候选者投票的分裂问题以及领导者频繁下线问题,提高集群的吞吐量和稳定性。准领导者的选举是经过模型验证和预测的,且结果具有唯一性和正确性。Raft 共识集群的分布式架构,为联邦学习过程提供了合适的环境,在不涉及节点数据隐私的情况下完成模型的构建,保证了去中心化程度和良好的安全性。

FL_Raft 的选举流程分为三大部分,即联邦学习、准领导者选举、候选者投票,如图 2 所示。其具体执行步骤如下:

1)集群稳定运行情况下,若领导者在线则跳转步骤 2,否则跳转步骤 4。

2)若是第一次联邦学习过程,则服务端首先广播初始模型 $f_0(w)$,$(w=0)$,否则跳转步骤 3。

3)执行正常的联邦学习过程,客户端 $i(i=1,2,\cdots,n)$ 使用本地数据集 D_i 进行模型 $f_l(w)$ 训练,并将更新参数 $U_i(w)$ 发送给服务端 j,在收集到充分的更新参数后服务端 j 进行聚合过程,并将新的全局模型广播 $f_g(w)$ 给集群中所有客户端,客户端更新本地模型为 $f_l'(w)=f_g(w)$。

4)当领导者不在线时,若集群中不存在准领导者,则跳转步骤 5,否则准领导者成为领导者,跳转步骤 6。

5)该步骤分为两点:候选者和准领导者的选举。

(1)候选者。跟随者等待选举超时,将状态设为候选者,为自己投一票,并向集群中的其余节点发起投票请求,其余跟随者若任期和日志条目均小于等于候选者,则向该候选者投一票,在候选者收集到超过集群总节点数一半的票数时,成为领导者,否则在投票超时后重新发起投票,等待新一轮的选举。

(2)准领导者。跟随者 i 利用本地的模型 $f_i'(w)$ 和数据集 D_i 进行模型选举过程，选举出的节点进入权益等待队列 Q_p，然后 Q_p 中的节点进行权益选举过程，排名第一的节点成为准领导者，上述过程也可发生在领导者在线状态，以缩短领导者下线后的选举时间。按照 Q_p 队列中的顺序发起新一轮投票，在收到超过集群一半数量的票数后成为领导者，否则替换准领导者重新发起选举，若选举完仍无法选出领导者，则等待新一轮选举。

6)领导者上线，启动心跳，接收日志并进行共识过程，同时各节点收集训练所需数据集。

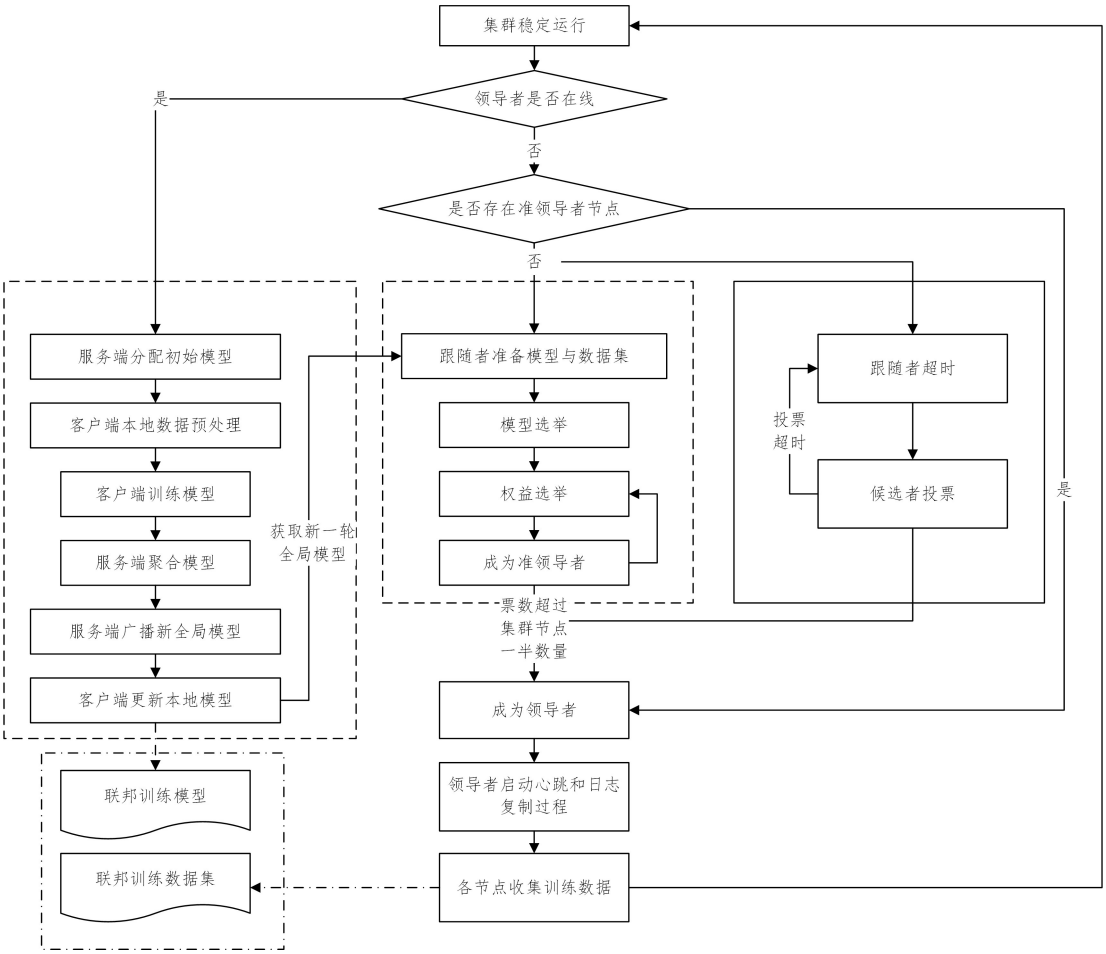

图 2 FL_Raft 共识方案选举流程图

Fig. 2 Flow chart of FL_ Raft's consensus programme election

算法 4 描述了 FL_Raft 单个节点的运行过程。

算法 4 FL_Raft 算法

输入：节点 n，总节点数 N

输出：无

```
1. if L_new≠Null then
1.    /*接收全局模型并更新本地模型*/
2.    if n[MR]==0 then
3.       f(w)←f_L(w);
4.       f_n(w)←f(w);
5.    end if
6. else
7.    if n∈Follower&&L_q==Null then
8.       /*未触发选举超时执行准领导者选举*/
9.       if n[ETO]≠True then
10.         /*执行模型选举和权益选举*/
11.         if L_q==n then
12.            /*执行投票选举票数过半为领导者*/
13.            if v >=N/2 then n==Leader
14.         end if
15.      else
16.         /*执行普通 Raft 选举*/
17.      end if
18.   end if
19.   if n==Leader then
20.      /*领导者启动心跳和日志复制过程*/
21.   end if
22. end if
23. /*收集训练数据*/
24. n. collectionData(T_collection).
```

4 算法分析

FL_Raft 通过结合联邦学习，创新地提出了准领导者的选举方式，并且仍然保留领导者全部的日志复制责任。为保证 FL_Raft 算法的可行性，我们以 Raft 的标准对其进行安全性、一致性和活性的分析。

4.1 安全性分析

FL_Raft 的安全性是在领导选举前以及选举过程中通过增加一些限制来完善的。这些限制包括节点的数据隐私保护、模型的可追溯性以及权益选举确保节点最新的原则。

4.1.1 节点的数据隐私保护

区块链中的交易数据是公开透明的，这很大程度地限制了节点的隐私性。加密算法虽然能一定程度地保证数据不被泄露，但是仍然无法保证数据隐私的暴露问题。对于节点的隐私数据问题，FL_Raft 共识算法采取了联邦学习进行分布式模型训练。联邦学习客户端的数据不会参与通信传输过程，只用于本地训练，负责通信传输的内容仅为模型更新参数，并且参与训练的数据仅为各节点的特征属性，没有涉及日志内容，即使存在恶意节点通过更新参数反推出训练数据，也无法知道节点存储的具体日志内容，因此这并不会造成严重的数据隐私泄露问题。

4.1.2 模型的可追溯性

假设任期 T 的领导者广播了一个全局模型 $f_g(w)$，但是某个客户端由于通信故障，未能接收并更新自己的本地模型 $f_l(w)$，设大于 T 任期的某一客户端没有 $f_g(w)$ 的模型。由于模型选举是有前提的，即必须接收领导者最新轮次的全局模型 $f_g(w)$，此时客户端会检查本地的模型轮次，它会发现 $f_l(w).round=T-1\neq T$，此时无法发起模型选举。若在下一任期该节点又恢复通信，它则需要先向领导者发起请求接收缺失的 T 轮模型 $f_g(w)$，之后才会接收最新轮次的模型，从而发起正确的模型选举过程，因此客户端必须要有从第一轮到最新轮次的模型，这是模型选举成功的基础。综上所述，模型的可追溯性保证了选举的正确性和安全性。

4.1.3 权益选举确保节点最新

这里的最新如 3.2 节中的表 1 所列，指符合队列中的最新原则。任期和日志最新是确保节点符合 Raft 算法成为领导者的条件，模型最新是防止恶意行为。若存在恶意节点构建一个虚假的本地模型 $f_l'(w)$来提高轮次 T，即使通过模型选举，在权益选举时与其他队列中的节点并不一致，也会判定为不合格节点，无法成为准领导者。权益选举相当于是对模型选举的一轮筛选，筛选出不符合领导者条件的节点，并确保选举出的准领导者与集群中部分符合领导者条件的节点保持一致。

4.2 一致性分析

Raft 算法的一致性体现在领导选举、日志复制和安全性，FL_Raft 保留了原 Raft 的日志复制方式，安全性参考 4.1 节，本节将对领导选举过程的一致性进行分析。

Raft 中领导者会周期性地向所有跟随者发送心跳包，以此来重置跟随者的选举超时，其余节点由此来确认领导者的在线状态。如果跟随者选举超时期间没有收到领导者的任何消息，则它会认为领导者已经下线，从而发起新一轮选举。而 FL_Raft 这一过程将发生在准领导者产生之后，即跟随者可以随时进行模型选举和权益选举以产生准领导者，当准领导者触发选举超时后，才会发起新一轮的选举。

性质 1 模型选举和权益选举出的准领导者唯一且正确。

跟随者选举所用的模型是服务端广播的全局模型，因此跟随者进行选举的模型必然是可追溯的，所有跟随者均需要不断更新本地模型来提高选举的成功率，并且在发起模型选举前需要接收服务端最新轮次的全局模型，故可以确保集群中大多数跟随者的本地模型是相同的。权益选举均在可靠的第三方节点进行，各节点执行相同的权益选举过程，可以保证一致性。

在满足性质 1 的基础上，准领导者还是权益最高的节点，其唯一性保证了不会发生投票分裂问题，它也必须要获得超过一半节点数量的投票才能成为领导者，满足大多数节点达成一致性的原则。最终确保选出的领导者是共识一致性正确的结果，且性能优良，不会产生频繁下线问题。

4.3 活性分析

FL_Raft 的活跃性体现在大规模异构节点集群中，通常这类集群中各节点的性能不同，数据服从 No-IID 分布，且集群节点角色处于动态变化中。对于参差不齐的节点和数据，传统的 Raft 算法采用随机选举策略，选出的领导者质量也不尽相同。一个共识集群性能往往体现在领导者的能力大小，因此选出一个高性能的领导者是我们改进集群活性的思路。

FL_Raft 结合联邦学习使发起投票的准领导者节点是经过模型筛选的高性能节点，并且随着集群的运行，数据集会更加完善，模型会更加准确，选举出的领导者更能代表集群的优秀性能，以此维持着共识算法的活性。

5 实验评估

5.1 实验环境

实验使用虚拟机多端口映射方式搭建共识集群环境，运行环境硬件使用 8 核/16GB/64 位的 Intel(R)Core(TM)i7-10750，GPU 为 Geforce RTX 2060，其中虚拟机硬件环境各不相同，用于模拟节点异构环境；软件开发环境采用 Ubuntu18.04/Python3.6.5，联邦重建部分采用 Tensorflow2.0 开发的 Federated Learning API，模型基于 Keras 库开发。集群部分默认配置参数如表 2 所列，配置均可以根据集群的特点进行修改。本文将在搭建的集群环境中对选举时延和可用性、领导者在任时长和可靠性、共识时延、数据吞吐量和模型准确率进行实验，以此验证 FL_Raft 的有效性和可靠性。

表 2 集群默认配置

Table 2 Cluster default configuration

参数	描述	默认值
R_{ip}	节点 ip	192.168.101.1 至 192.168.101.21
R_{port}	共识节点端口	[12000,13000]
L_{port}	集群入口节点端口	9998
F_{port}	联邦服务端口	11000
C_{port}	联邦客户端口	11001
P_r	共识节点日志地址	Data/Node
P_f	联邦服务端数据地址	Data/Federated
P_c	联邦客户端数据地址	Data/Client
T_{heart}	心跳时间/ms	50
T_{elec}	选举超时/ms	200
T_{vote}	投票超时/ms	200

5.2 选举时延与可用性

选举时延是衡量共识算法的一项重要标准，传统 Raft 算法中会经常发生分裂投票问题，导致选举时延较高，FL_Raft 方案结合联邦学习优化了选举过程。实验设定领导者在任时长为 10s，当超时后会停止发送心跳并变为跟随者，集群会发起新一轮的选举，以任期增长作为标准，假设选举前的任期为 E_{t1}，选举后的任期为 E_{t2}，则选举时延 $E_{delay}=T_{elec}+(E_{t1}-E_{t2})\times T_{vote}$，实验在 12 个节点的环境中进行，取 50 次实验的结果，如图 3 所示。

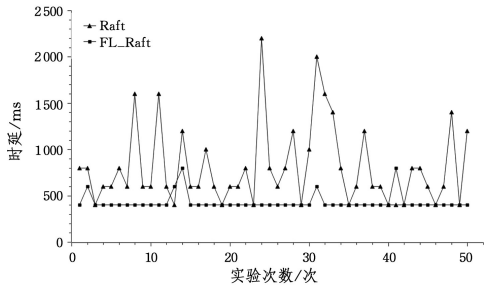

图 3 FL_Raft 与经典 Raft 共识算法的选举时延比较

Fig. 3 Comparison of election delay between FL_Raft and classical Raft consensus algorithm

通过图 3 可以发现，Raft 的选举时延波动性较大，最大达到 2 200 ms，平均选举时延约为 800 ms，而 FL_Raft 的选举时延较为稳定，平均选举时延约为 400 ms。FL_Raft 的低选举时延主要源于准领导者选举过程，减小了分裂投票的概率。

在选举过程中，由于集群中无领导者，因此无法处理新的日志请求，集群此时会处于不可用的状态，故选举时延会间接影响集群的高可用性。高可用性指集群具有较高的无故障运行能力，可用性为平均无故障时间与集群总运行时间的比值。我们设定在集群总运行 50 s 内进行 10 次实验，测定在正常运行情况下 Raft 和 FL_Raft 的集群可用性，实验结果如表 3 所列。

表 3 节点数为 12 的集群可用性

Table 3 Cluster availability with 12 nodes

实验次数	1	2	3	4	5	6	7	8	9	10
总运行时间/s	50									
Raft 故障时间/s	3.4	2.6s	3.6	5	2.8	3.2	2.6	3.6	4	3.4
Raft 可用性/%	93.2	94.8	92.8	90.0	94.4	93.6	94.8	92.8	92.0	93.2
FL_Raft 故障时间/s	1.8	1.6	1.8	2.2	2.4	1.6	1.6	1.6	1.6	1.8
FL_Raft 可用性/%	96.4	96.8	96.4	95.6	95.2	96.8	96.8	96.8	96.8	96.4

由表 3 可知，在集群正常运行的情况下，Raft 的可用性维持在 92%左右，而 FL_Raft 的可用性最高可以达到约 97%。FL_Raft 方案提高了集群可用性，同时节约出的故障时间可以用于处理更多的消息，有助于提高集群吞吐量。

5.3 领导者的可靠性

领导者的可靠性影响着集群的稳定运行，一个网络质量不佳、计算能力弱的领导者无法为集群提供良好的运行环境，随时面临着下线的风险，而频繁地下线又会导致多次选举，影响集群的可用性，而可靠的领导者可以更快速地处理消息，提高集群吞吐量。领导者可靠性为领导者在线时长与总运行时间的比值，我们通过增加每秒日志请求量来对领导者施压，进行 5 组实验，得到在集群稳定运行情况下 Raft 和 FL_Raft 的平均领导者在线时长和可靠性，结果如图 4 所示。

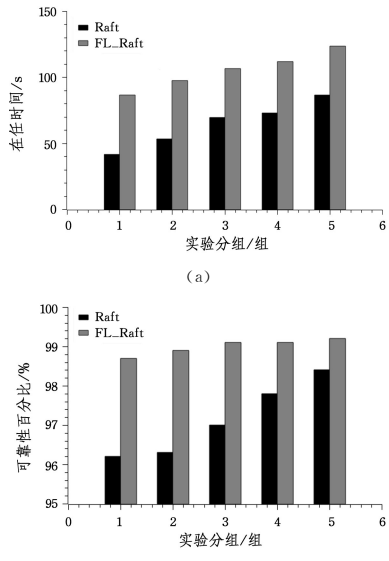

图 4 领导者在任时长和可靠性对比

Fig. 4 Comparison of leader tenure and reliability

由图 4(a)可知，Raft 的平均在任时长为 65 s，FL_Raft 为 105 s，FL_Raft 的领导在任时长比 Raft 高 40%左右；由图 4(b)可知，Raft 的领导者可靠性约为 97%，而 FL_Raft 的可靠性可以达到 99%，同时可以看出 5 组实验中 FL_Raft 集群的领导者更稳定。FL_Raft 通过模型选举和权益选举过程来筛选出集群中较为可靠的领导者，以保证集群的稳定运行。

5.4 共识时延

共识时延指用户向共识集群发送到接收请求的时间间隔，是衡量共识效率的一个重要指标。实验设计用户以随机频率(1～5s 内一次)对集群发送日志请求，记录为时间 T_1，通过再次接收到集群回复的成功响应为止，记录时间为 T_2，则共识时延 $T_{cd}=T_2-T_1$，设定节点数为 15，进行 10 组实验，计算平均共识时延。在相同的网络环境下(传输速率为 100 M/s)，对 Raft，FL_Raft，K-Raft[26] 等 6 种算法的吞吐量进行对比，计算各算法从请求到结果返回的时间差，取平均值测量共识时延，实验结果如图 5 所示。

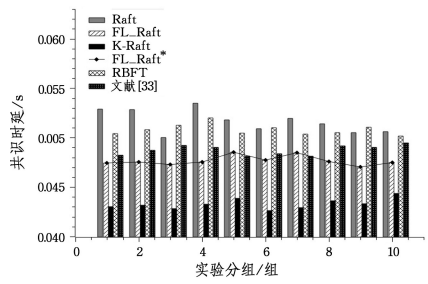

图 5 Raft，FL_Raft 和 K-Raft 等 6 种算法共识时延的对比

Fig. 5 Consensus delay comparison of six algorithm includeing Raft, FL_Raft and K-Raft

在 10 组实验中，共识算法的共识时延的稳定性较好，FL_Raft 的共识时延平均稳定在 47 ms，Raft 为 51 ms，K-Raft 为 43 ms，RBFT 为 50 ms，文献[33]中的算法的共识时延为 48 ms。通过对比后发现，相同的实验环境中 FL_Raft 共识时延较其他算法更低，与 K-Raft 的共识时延十分相近，K-Raft

的低共识时延得益于其并行日志复制方式，我们在今后的研究工作中将深入这部分。FL_Raft 的准领导者选举过程降低了共识时延，加快了领导者处理日志消息的速度。

5.5 吞吐量

吞吐量指集群每秒钟处理消息的数量，为方便计算吞吐量，我们在用户发送的日志消息中添加时间戳 T_{s1} 和唯一哈希 H，当用户再次接收到集群领导者的回复请求时，进行哈希匹配，若匹配成功，则记录接收时间戳 T_{s2}，设定用户单次发送消息中包含 N 个日志，得到 $TPS=N/(T_{s2}-T_{s1})$。实验设置 $N=10$，在相同网络环境下(100 M/s)对 Raft、FL_Raft、K-Raft、RBFT 和文献[33]中的算法进行 10 组实验，取平均值后得到的结果如图 6 所示。

由实验结果可知，10 组实验中，K-Raft 吞吐量的表现最好，平均每秒处理约 230 个日志消息，FL_Raft 为 215 个每秒，Raft 为 190 个每秒，RBFT 为 200 个每秒，文献[33]中的算法为 210 个每秒。FL_Raft 的吞吐量较其他算法均有所提高，与并行复制日志的 K-Raft 很接近。FL_Raft 集群表现出了良好的处理日志效率，这得益于选举过程保证了领导者的可靠性，从而提升了共识性能。

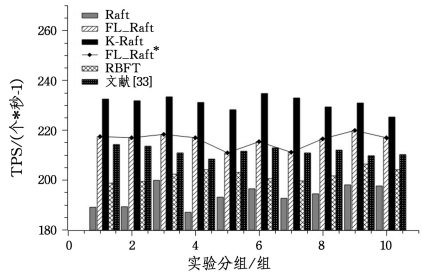

图 6 Raft，FL_Raft 和 K-Raft 等 6 种算法的吞吐量的对比

Fig. 6 Comparison of Raft, FL_Raft and K-Raft throughput

5.6 模型准确率

模型准确率指使用经过检验的数据集去运行模型，将得到的结果与原数据集结果进行比对，其中正确率即为模型准确率。由 4.3 节可知，模型的准确率影响着选举的结果。因此设计在不同数据量和任期的集群环境中，使用收集到的验证数据集对模型进行评估，以检验 FL_Raft 中联邦学习的准确性，实验结果如表 4 所列。

表 4 不同数据量和任期的模型损失值与准确值

Table 4 Model loss and accuracy values for different data volumes and tenure

数据量	0	20	50	100	200	500	1000	2000	3000	4000
损失值	1	0.6998	0.6977	0.6528	0.5389	0.2594	0.1835	0.1396	0.1236	0.1160
准确值	0	0.4997	0.4505	0.5224	0.7624	0.9023	0.9552	0.9601	0.9633	0.9657
任期	1	2	3	4	5	6	7	8	9	10
损失值	0.6930	0.6783	0.6451	0.5403	0.3371	0.2799	0.2263	0.1876	0.1601	0.1415
准确值	0.5059	0.6661	0.7788	0.8215	0.8993	0.9328	0.9548	0.9593	0.9635	0.9650

由表 4 可知，随着数据量的增长，模型的损失值不断下降，准确值不断上升，在 2000 个数据量时趋于稳定，准确率可以保证在 96%以上，极大地提高了选举的正确性。任期的增加意味着数据量的不断完善。从表中可以看出，准确值和任期成正比，在第 7 个任期后准确率趋于稳定，达到 95%以上。我们测算了第 7 个任期的数据量，约在 1000 个左右，和数据量表的结果较为一致，说明了实验的准确性和可靠性。

FL_Raft 中的联邦学习模型为领导者的筛选提供了较大的参考价值，随着集群的稳定运行，选举的准确率会不断提高，保证了共识过程中选举的高效性。若联邦学习模型长时间趋于稳定，则可以减小训练频率以降低节点的计算量。

结束语 本文提出了一种基于联邦学习模型的选举共识方案 FL_Raft，创新性地提出通过收集节点本地数据集来进行联邦训练，用得到的模型和权益值筛选合适的节点成为准领导者，再通过投票选举成为领导者的过程，解决了频繁投票分裂和领导者下线的问题，提高了集群的稳定性和可用性。联邦学习的训练过程保证了节点数据隐私不被泄露，同时模型的可追溯性和权益值的判定也一定程度上保证了选举过程的安全性。实验结果表明，FL_Raft 与 Raft 相比在选举时延、可用性、领导者的在任时长和可靠性、共识时延以及吞吐量方面均有较好的表现。参阅其他主流改进 Raft 的算法，本文的 FL_Raft 共识方案也存在需要改进的地方，如该方案仅改进了 Raft 领导者的选举部分，而日志复制部分也是提高吞吐量的一个切入点，可以通过并行性复制来进一步降低算法的通信复杂度并提高日志的存储效率。实际应用中若出现联邦学习数据量匮乏的情况，则会导致模型不完善，无法针对特殊情况进行准确分析，造成低准确率以及拜占庭环境中的安全性问题等。我们接下来的工作将对这些问题进行优化。

参 考 文 献

[1] NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system [R]. (2008-05-19)[2020-08-19].

[2] ZHENG Z, XIE S, DAI H N, et al. Blockchain challenges and opportunities: A survey[J]. International Journal of Web and Grid Services, 2018, 14(4): 352-375.

[3] KARAGIANNIS T, BROIDO A, FALOUTSOS M, et al. Transport layer identification of P2P traffic[C]// Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement. 2004: 121-134.

[4] AHLSWEDE R, CSISZAR I. Common randomness in information theory and cryptography. II. CR capacity[J]. IEEE Transactions on Information Theory, 1998, 44(1): 225-240.

[5] MINGXIAO D, XIAOFENG M, ZHE Z, et al. A review on consensus algorithm of blockchain[C]// 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2017: 2567-2572.

[6] El IOINI N, PAHL C. A review of distributed ledger technologies[C]// OTM Confederated International Conferences "On the Move to Meaningful Internet Systems". Cham: Springer, 2018: 277-288.

[7] HUANG Y，ZENG Y，YE F，et al. Incentive Assignment in PoW and PoS Hybrid Blockchain in Pervasive Edge Environments [C]// 2020 IEEE/ACM 28th International Symposium on Quality of Service(IWQoS). IEEE，2020：1-10.

[8] ZHENG Z，XIE S，DAI H，et al. An overview of blockchain technology：Architecture，consensus，and future trends [C]// 2017 IEEE International Congress on Big Data (BigData congress). IEEE，2017：557-564.

[9] ONIRETI O，ZHANG L，IMRAN M A. On the viable area of wireless practical byzantine fault tolerance (pbft) blockchain networks[C]// 2019 IEEE Global Communications Conference (GLOBECOM). IEEE，2019：1-6.

[10] JAKOBSSON M，JUELS A. Proofs of work and bread pudding protocols[M]// Secure Information Networks. Springer，Boston，MA，1999：258-272.

[11] KIAYIAS A，RUSSELL A，DAVID B，et al. Ouroboros：A provably secure proof-of-stake blockchain protocol[C]// Annual International Cryptology Conference. Cham：Springer，2017：357-388.

[12] LAMPORT L. The part-time parliament[M] Concurrency：the Works of Leslie Lamport. 2019：277-317.

[13] LAMPORT L. Paxos made simple[J]. ACM Sigact News，2001，32(4)：18-25.

[14] LISKOV B，COWLING J. Viewstamped replication revisited[J/OL]. http：//hdl. handle. net/1721. 1/71763.

[15] CASTRO M，LISKOV B. Practical Byzantine fault tolerance and proactive recovery[J]. ACM Transactions on Computer Systems (TOCS)，2002，20(4)：398-461.

[16] YANG F，ZHOU W，WU Q Q，et al. Delegated proof of stake with downgrade：A secure and efficient blockchain consensus algorithm with downgrade mechanism[J]. IEEE Access，2019，7(1)：118541-118555.

[17] ONGARO D，OUSTERHOUT J. In search of an understandable consensus algorithm[C]// Proceedings of the USENIX Annual Technical Conf(USENIX ATC). 2014：305-319.

[18] VAN RENESSE R，ALTINBUKEN D. Paxos Made Moderately Complex[J]. ACM Computing Surveys，2015，47(3)：1-36.

[19] MAZZONI M，CORRADI A，DI NICOLA V. Performance evaluation of permissioned blockchains for financial applications：The ConsenSys Quorum case study[J]. Blockchain：Research and Applications，2022，3(1)：1-12.

[20] MING X，XU X，JIAN X，et al. A Forensic Analysis Method for Redis Database based on RDB and AOF File[J]. Journal of Computers，2014，9(11)：2538-2544.

[21] LARSSON L，TARNEBERG W，KLEIN C，et al. Impact of etcd deployment on Kubernetes，Istio，and application performance [J]. Software：Practice and Experience，2020，50(10)：1986-2007.

[22] MCMAHAN B，MOORE E，RAMAGE D，et al. Communication-efficient learning of deep networks from decentralized data [C]// Artificial Intelligence and Statistics. PMLR，2017：1273-1282.

[23] HUANG H，DING S，ZHAO L，et al. Real-time fault detection for IIoT facilities using GBRBM-based DNN[J]. IEEE Internet of Things Journal，2019，7(7)：5713-5722.

[24] SHAYAN M，FUNG C，YOON C J M，et al. Biscotti：A blockchain system for private and secure federated learning[J]. IEEE Transactions on Parallel and Distributed Systems，2020，32(7)：1513-1525.

[25] PAN S J，TSANG I W，KWOK J T，et al. Domain adaptation via transfer component analysis[J]. IEEE Transactions on Neural Networks，2010，22(2)：199-210.

[26] WANG R，ZHANG L，XU Q，et al. K-Bucket based Raft-like consensus algorithm for permissioned blockchain [C]// 2019 IEEE 25th International Conference on Parallel and Distributed Systems(ICPADS). IEEE，2019：996-999.

[27] HUANG D Y，LI L，CHENG B，et al. RBFT：Byzantine fault tolerant consensus mechanism based on raft cluster[J]. Journal of communication，2021，42(3)：209-219.

[28] ZHAN Y，WANG B，LU R，et al. DRBFT：Delegated randomization byzantine fault tolerance consensus protocol for blockchains [J]. Information Sciences，2021，559(1)：8-21.

[29] FU W，WEI X，TONG S. An improved blockchain consensus algorithm based on raft[J]. Arabian Journal for Science and Engineering，2021，46(9)：8137-8149.

[30] CHEN Y T，CHEN Q，XIE Y X. A methodology for high-efficient federated-learning with consortium blockchain[C]// 2020 IEEE 4th Conference on Energy Internet and Energy System Integration(EI2). IEEE，2020：3090-3095.

[31] LI Y，CHEN C，LIU N，et al. A blockchain-based decentralized federated learning framework with committee consensus [J]. IEEE Network，2020，35(1)：234-241.

[32] SHAYAN M，FUNG C，YOON C J M，et al. Biscotti：A blockchain system for private and secure federated learning[J]. IEEE Transactions on Parallel and Distributed Systems，2020，32(7)：1513-1525.

[33] KIM D，DOH I，CHAE K. Improved Raft Algorithm exploiting Federated Learning for Private Blockchain performance enhancement[C]// 2021 International Conference on Information Networking(ICOIN). IEEE，2021：828-832.

RONG Baojun，born in 1998，postgraduate，is a member of China Computer Federation. His main research interests include federal learning and blockchain.

ZHENG Zhaohui，born in 1968，professor，Ph. D supervisor，is a member of China Computer Federation. His main research interests include data mining and network security.

（责任编辑：喻藜）