计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(3):116−124 [doi: 10.15888/j.cnki.csa.008983]

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

面向风电场景的联邦学习平台高性能通信优化①

于　航[1], 周继威[1], 张　涵[1], 孔祥锋[2,3], 张玉会[2,3]

[1](中能电力科技开发有限公司, 北京 100034)
[2](中国科学院 信息工程研究所, 北京 100093)
[3](中国科学院大学 网络空间安全学院, 北京 100049)
通信作者: 张玉会, E-mail: zhangyuhui@iie.ac.cn

摘　要: 风能作为清洁能源为改善我国能源结构发挥着越来越重要的作用. 风电场机组及设备的数据可能会包含机组或风场的隐私敏感信息, 这些隐私数据一旦被泄露, 将会为风电场带来巨大的经济风险和法律风险. 联邦学习作为重要的隐私计算手段, 能够保证原始数据不出本地的情况下完成模型的建模和推理, 实现各参与方在互不泄露隐私的前提下实现联合计算, 从而有效应对风电数据分析面临的挑战. 但是, 联邦学习计算过程中存在大量的通信开销, 这成为限制联邦学习技术在风电场景下应用的关键性能瓶颈. 因此, 本文以经典的联邦学习算法 XGBoost 为例, 深入分析了联邦学习计算过程中的通信问题, 提出采用 RDMA 作为底层传输协议的解决方案, 设计并实现了一套高性能联邦学习平台通信库, 有效提升了联邦学习系统的性能.

关键词: 风电; 联邦学习; 通信优化; RDMA

引用格式: 于航,周继威,张涵,孔祥锋,张玉会.面向风电场景的联邦学习平台高性能通信优化.计算机系统应用,2023,32(3):116–124. http://www.c-s-a.org.cn/1003-3254/8983.html

Optimization of High-performance Communication for Federated Learning in Wind Power

YU Hang[1], ZHOU Ji-Wei[1], ZHANG Han[1], KONG Xiang-Feng[2,3], ZHANG Yu-Hui[2,3]

[1](Zhong Neng Power-tech Development Co. Ltd., Beijing 100034, China)
[2](Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China)
[3](School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: As clean energy, wind power plays an increasingly important role in improving China's energy structure. Data on wind farm units and equipment may contain relevant privacy-sensitive information. Once the information is divulged, it will bring huge economic and legal risks to the wind farm. Federated learning (FL) is an important privacy-preserving computing technique, through which model training and inference are completed without transmitting raw data, so as to achieve joint computation among all participants without privacy disclosure and effectively deal with challenges in analyzing wind power data. However, significant communication overheads generated during FL computation have become a major performance bottleneck that has limited the application of the FL technique in wind power scenarios. Therefore, this study takes the typical FL algorithm, namely, XGBoost, as an example and deeply analyzes the communication problems in FL computation. In addition, the study proposes a solution that RDMA shall be utilized as the underlying transport protocol and designs a set of high-performance FL platform communication libraries, which effectively improves the performance of the FL system.

Key words: wind power; federated learning (FL); communication optimization; remote direct memory access (RDMA)

① 收稿时间: 2022-07-18; 修改时间: 2022-09-07; 采用时间: 2022-09-21; csa 在线出版时间: 2022-12-16
CNKI 网络首发时间: 2022-12-19

1 引言

近年来, 风能作为清洁能源为改善我国能源结构发挥着越来越重要的作用, 风力发电得到了迅猛发展. 风电系统整体在向智能化方向发展, 风电机组状态监测的广度和深度不断加强, 生成风电数据呈典型的海量特征. 如何针对风电大数据进行高效、智能的分析 (例如故障诊断与预警), 具有重要的研究意义. 然而, 基于设备状态历史数据建模需要大量的机组历史数据. 设备状态历史数据是拥有者的宝贵资产, 同时这些数据也可能会包含机组或风场的隐私敏感信息, 这些隐私数据一旦被泄露, 将会给数据拥有者带来巨大的经济风险和法律风险.

风机数据如何在隐私安全的环境下发挥最大效益成为风电运营部门面临的主要问题. 一方面, 对于隐私安全保护, 国内外学者进行了大量的研究[1−7], 尤其是近几年随着隐私计算的兴起, 数据保护从传统的存储和传输过程中的数据加密, 进一步拓展到计算过程. 在数据采集、存储、处理、发布 (含交换)、销毁等各个环节对数据隐私进行全面的保护, 大大降低恶意软件、内部攻击和恶意或管理疏忽等安全风险. 另一方面, 现在大数据领域面临专业化的分工, 数据拥有者具有海量的行业数据、健全的数据管理、丰富数据应用环境, 而模型提供方拥有强大算法优势, 如何高效地整合双方的产业优势, 实现数据拥有方和模型提供方的优势结合, 在保护模型提供方的核心知识产权的同时, 积极参与并推动风电大数据产业化, 充分挖掘风电大数据的价值具有重要的意义.

传统集中式机器学习需要共享原始数据, 存在数据隐私泄露风险. 针对这一问题, 联邦学习是实现“数据可用不可见”的重要隐私计算手段, 能够在原始数据不出本地的前提下完成模型训练和推理, 有效保护风电数据的隐私安全. 联邦学习是一种分布式计算框架, 为了协同各方联合完成计算任务, 参与方需要在多方安全协议调度下, 传输计算过程中的中间结果. 因此, 在联邦学习系统中, 参与方之间需要进行大量的数据通信来交互模型更新信息, 这导致通信效率成为联邦学习系统中重要的性能瓶颈. 联邦学习系统的通信效率主要体现在数据通信的延迟和带宽. 通信效率优化的主要目标是降低联邦学习过程中的通信耗时. 例如, 从安全多方协议角度出发, 通过设置阈值等方式, 排除参与方中权重较低的信息, 减少通信数据次数. 或者采用数据压缩策略甚至有损压缩技术, 降低减小通信数据量. 但是无论是丢弃部分低权重数据信息还是高效的有损压缩都会造成联邦训练所得模型精度的降低.

本文提出在联邦学习系统中引入 RMDA (remote direct memory access) 技术提高通信带宽, 缩短通信延迟, 从而提升多方通信效率, 设计并实现了一套高性能基于 RDMA 的联邦学习通信库, 有效缩短风电场景下联邦学习系统中的数据通信时间. 主要贡献如下.

(1) 探讨了联邦学习面临的通信问题, 并详细分析了纵向联邦学习 XGBoost 算法中的通信瓶颈.

(2) 提出了一种基于 RDMA 协议的网络传输方案, 并对协议应用难点提出了优化的解决方案, 设计并实现了高性能的通信函数库.

(3) 基于开源联邦学习框架 FATE, 采用典型的联邦学习应用, 本文所提基于 RDMA 的通信库实现了联邦学习系统 2–4 倍的性能提升.

2 背景知识

联邦学习中的隐私保护协议需要存在大量的网络通信, 而由于传统以太网通信的内核态转换、消息移动与拷贝、CPU 带宽与网卡带宽差距显著等特点, 通信效率已成为制约联邦学习技术发展的性能瓶颈.

2.1 风电场景下的联邦学习应用

在风电场景中, 跨风机或跨场站的数据联合可以有效地帮助提高风机状态检测模型的预测精度. 联邦学习系统具有输入隐私性、计算正确性、去中心化这 3 大特性. 隐私性体现在各参与方间输入数据独立, 信息呈密态流动. 正确性体现在能准确无误地完成约定的计算任务. 去中心化体现在所有的参与方地位平等, 不存在特殊权限或者第三方. 因此, 一个典型的风电场景下应用联邦学习的系统架构如图 1 所示.

当一个联邦学习任务发起时, 枢纽节点会组建相关传输网络及信令控制. 各参与方节点通过枢纽节点进行路由寻址, 选择具有相关类型数据的其他数据拥有方合作, 在安全多方协议调度下使用本地数据完成联合计算, 得到最终正确的反馈结果. 整个过程中本地数据无需泄露给其他任何参与方.

数据的合作流通与网络信息的交互密不可分, 从上述架构图可以看到, 任何一个协作任务的执行会涉及大量网络通信, 而对于复杂的以太网拓扑结构, 又会面临网络延迟、网络丢包、网络拥塞等一系列问题,

它们严重影响着多方安全计算任务的性能. 时间开销成为限制通信效率的瓶颈之一.

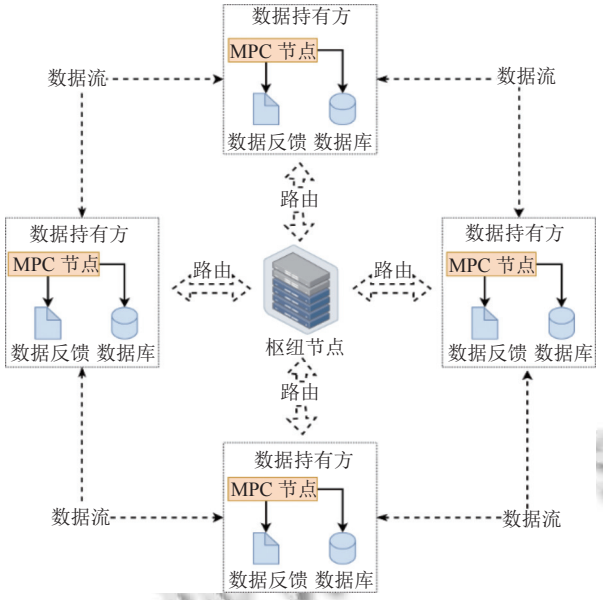

图 1 风电场景下联邦学习架构

2.2 RDMA 通信协议

在传统以太网的 TCP/IP[2] 协议层, 用户若想将数据从自身用户空间发送至远程机器的用户空间需要经过一系列复杂的工作. 首先, 数据发送方需要将数据从用户空间缓冲区 (buffer) 复制到内核空间的 socket buffer 中, 然后在内核空间中添加数据包头进行数据封装, 并经过多层网络协议层的转化, 最终才会被推送至以太网网卡 (NIC) 中进行网络传输. 同样, 消息接收方接收到从远端机器发送而来的数据包后, 同时将数据包从 NIC 的 buffer 中拷贝至 socket buffer 中, 然后再经过一系列网络协议层转化. 解析后的数据被复制到相应位置的用户空间 buffer, 最后完成系统上下文的切换, 数据才能最终被应用程序所调用. 这种通过内核态发送消息的工作模式流程繁琐, 造成了很高的数据移动和数据拷贝开销. 并且随着技术的革新, CPU 片上带宽和网络带宽的差距显著, 进一步制约了网络技术的发展.

为了克服上述问题, 高性能网络开始迅速发展. 由于传统以太网络需要 CPU 完成繁重的网络数据包封装工作, TOE (TCP/IP offloading engine)[3] 最早提出将主机处理器中从数据包处理工作中解放出来, 使其专注于应用执行, 而数据处理工作则交由特定支持 Offloading 操作的网卡完成. U-Net (user-net networking)[4] 更加彻底, 它通过虚拟网络接口的方式, 使得应用程序可以直接通过 MUX 单元访问内核而无需数据移动和拷贝. 通过这种方式, U-Net 将协议栈转移至用户空间, 并且在数据通信路径中彻底地去除了内核态, 这样不仅带来了更高的性能优势, 也带来更高的灵活性. 随后, VIA (virtual interface architecture)[5] 提出了标准化的用户级网络通信模式, 并结合了 U-Net 和远程 DMA (direct memory access) 设备.

随着技术的不断成熟, InfiniBand 传输网络中开始支持 RDMA[6] 的网络协议. 其本质是一种智能网卡与软件架构充分优化的远端内存直接高速访问技术, 它通过将相关协议固化在硬件网卡中, 使得用户能够以零拷贝和内核旁路的方式, 实现高性能的远程数据直接存取的目标. 这样的工作模式也让其具有低延迟、低 CPU 源占用率和高带宽的特性.

图 2 展示了 RDMA 核心技术栈, 主要分为 3 层. 顶层为用户层, 提供了一系列如 send/recv、read/write 等 verbs 接口供用户操作 RDMA 硬件网卡. 内核空间直接处理内核旁路请求, 整个网络传输过程不需要 CPU 参与. 底层硬件层是硬件网卡的实现层, 具有自己独立的页表用来完成虚实地址的转换, 并且保留相关的内存钥匙进行权限检查以保证系统安全性.

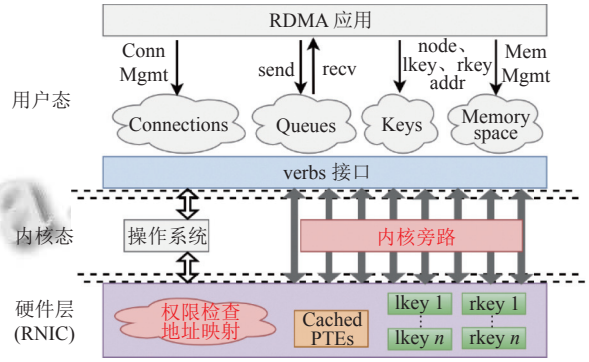

图 2 RDMA 技术栈

基于上述系统架构, 使用 RDMA 协议传输消息的主要工作流程如图 3 所示, 可分为如下步骤.

(1) 首先在参与方 A 与参与方 B 间建立起 RDMA 连接并完成初始化.

(2) 应用程序在内存中注册数据存放区域至 RDMA 网卡, 并获取返回的内存钥匙. 随后 RDMA 网卡即拥有了内存区域的操作权限.

(3) 用户态应用程序产生读写请求, 每个请求包含至少 3 部分信息: address 为待发送数据所在内存地址;

mem_key 为安全权限检查的内存钥匙; data 为对应数据域. 该请求通过 verbs 接口发送至 RDMA 网卡的发送队列 (send queue) 或接收队列 (receive queue) 中, 并同步至远端 RDMA 网卡对应队列中.

(4) RDMA 网卡不断地获取队列中的请求项并解析, 确认内存钥匙无误后, 将数据直接从远端的内存区域写入至本地内存区域 (写入时) 中, 完成一次数据传输, 最终返回相应的确认消息给远端.

在计算网络通信中, 通信延迟主要是指消息处理延迟和网络传输延迟. 前者是指消息在发送和接收阶段的处理时间, 对于小而密的消息类型该部分延迟占据主导地位. 后者是指网络传输链路上的延迟, 对于大消息, 该延迟更加突出. 风电应用场景中, 主要以小消息通信为主, 局域网中消息处理延迟会十分严重.

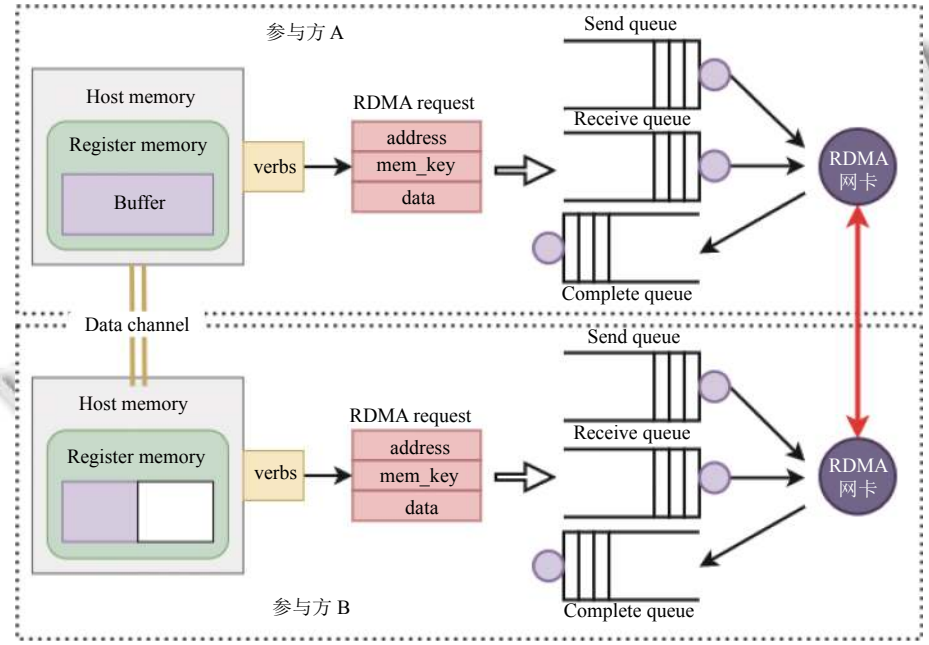

图 3 RDMA 工作流程图

3 问题分析

下面将以联邦学习 XGBoost 算法为例, 具体分析它们所面临的通信挑战.

XBGoost 算法[8] 的核心在于不断进行特征分裂进而生长出一棵完整的决策树, 而不断添加决策树的过程就是不断地学习一个新函数$f_k(x)$去拟合上一轮训练的残差[9,10]. 2021 年 Cheng 等人提出了该算法的纵向联邦学习实现—SecureBoost 框架[11]. 该框架明确指出了将 XGBoost 算法运用至分布式场景时可能面临的安全隐私问题, 并给出了相应的解决方案.

在纵向联邦学习场景[12] 下, 各参与方之间数据样本 ID 存在重叠, 但数据特征不重叠. 在联邦学习场景中, 我们将拥有数据标签的一方称为 guest 方, 其他合作方统称为 host 方. 在联合建模的过程中, 需要进行隐私保护的数据项很多. 首先是风机的特征信息, 这就意味着整体树模型会在各风机数据库间分散存储. 另外, DLG (deep leakage from gradients) 攻击表明攻击者可以从梯度信息中恢复出用户的原始数据, 因此训练过程中的一阶导数、二阶导数等信息需要被加密保护. 由于在加密保护下的梯度信息还需要在后续计算过程中还需要完成梯度聚合操作, 因此, 具有密文计算能力的同态加密在联邦学习框架中被广泛应用.

纵向联邦学习中的决策树节点的分裂过程如图 4 所示, 其主要包含如下步骤: ① Guest 方利用数据标签计算损失函数的一阶导数和二阶导数, 即梯度g,h. 然后该信息会应用同态加密, 并将密文发送至其他参与方. ② 参与方利用密文梯度信息, 遍历自身数据集拆分点并计算拆分收益值, 构建直方图. ③ Guest 方综合所有的直方图, 计算出最优拆分点, 然后将决策结果同步至其他参与方. ④ 拥有决策结果对应数据特征的参与方对当前节点数据集进行拆分并保存, 拆分结果同样同步给所有参与方. ⑤ 所有参与方均按照此结果完成本轮树节点的拆分.

在上述流程图 4 中, 与通信相关的主要过程均

已用虚线箭头标注. 可以看到, 每一个节点的分裂会需要至少 5 次通信, 而一棵树深为 h 的决策树可能拥有2^h-1个节点, 那么一棵完整决策树的建模过程也将达到$5\times(2^h-1)$次通信. 同时, 算法传输过程中的梯度信息往往是一个高维向量, 该信息在经过同态加密算法加密后, 密态数据的大小变为原明文数据的百倍以上, 所以整个决策树的构建过程中将会有大量的数据需要传输. 另外, 实际过程中还存在大量较小的控制类消息传输, 它们也将带来一定的消息处理开销.

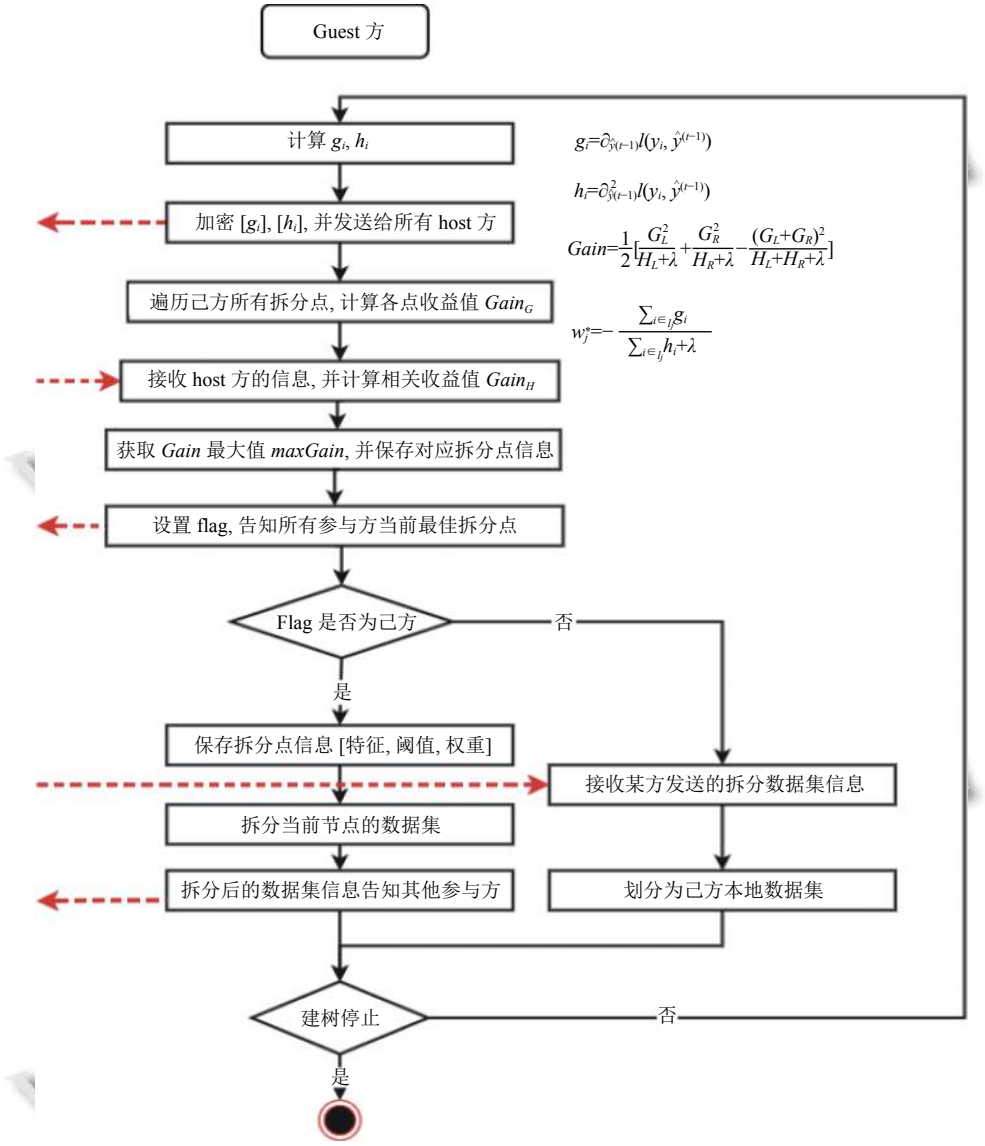

图 4 SecureBoost 流程图

4 优化方案设计

联邦学习场景中经常会面临密文数据导致传输消息大, 多方安全协议导致传输轮数多等问题. 传统以太网通信模式会面临内核态转换、消息移动与复制、数据包封装处理等操作, 它们会带来严重的消息处理开销, 并且碍于 CPU 带宽与网卡带宽的限制, 通信效率低已成为限制联邦学习性能的重要瓶颈. 本工作提出将具有高带宽、低延迟、低 CPU 资源占用率的 RDMA 协议应用在联邦学习计算场景, 利用其内核旁路特性将 CPU 从网络通信中脱离, 从而大大提高联邦学习计算效率.

4.1 RDMA 系统通信库整体架构设计

在 RDMA 协议的使用过程中, 整体系统逐层封装分为底层硬件层、中间转换层以及顶层应用层等 3 个

主要部分. 因此, 系统具有简单易用、通用性高、可扩展性强的特点, 其关系如图 5 所示.

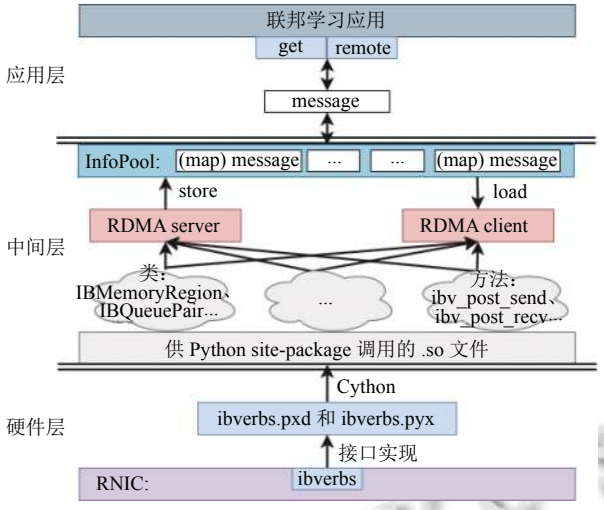

图 5 系统框架

RDMA 硬件层实现: 基于 RDMA 系统库 (RDMA core[13]) 所提供的 verbs 接口对相关通信类和方法进行了实现. 由于底层 verbs 接口是基于 C 语言提供, 而顶层算法层更多的是基于 Python 语言所编写, 因此, 这里选用了 Cython[14] 直接将实现的相关接口编译为了供 Python 包管理模块调用的.so 文件, 在保证便利性的同时, 尽可能地减少跨语言调用的性能开销, 提高了系统的整体效率.

中间消息层封装: 该层是从通用性和可扩展性角度, 对底层接口进行了进一步封装. 在联邦学习场景中, 由于各参与方计算和通信资源不对等, 可能导致消息发送方和消息接收方通信不同步. 通信双方会陷入互相等待的场景, 进一步造成计算资源浪费和时间开销. 为此, 本文引入了消息池策略进行优化, 消息池由一系列 HashMap 组成, 每一类型的消息就是一个 HashMap, 一条消息由 suffix 字段和消息值组成. 当 RDMA server 收到一条消息时, 就会在消息池的集合中找对该消息类型对应的 HashMap, 然后将 suffix 字段作为 Key, 消息值作为 Value 保存条目信息. 当应用程序需要获取某条消息时, 同样会从 HashMap 集合中索引到对应消息类型, 然后通过 suffix 字段作为 Key 获取具体的消息值. 通过将通信和计算过程解耦, 实现多方安全计算程序的异步进行, 减少各参与方互相等待的时间开销.

顶层应用层运用: 多方安全计算领域涵盖面相对广泛, 所涉及的策略和算法种类繁多, 因此顶层的调用接口必须具有简单易用, 可扩展性强的特点. 这里, 本文对照了开源联邦学习框架 FATE[14], 采用声明和实现的方式提供了相应的传输接口. 具体来说, 针对算法中的每一种传输变量, 只需要按照消息的发送方和接收方将其初始化为对应的传输类型, 它们继承至系统的 get 或者 remote 接口, 因此在实现上具有简洁高效的特点. 另外, 用户也可以在继承的过程中, 根据实际需求进一步扩展变量传输接口的相关功能, 在扩展性上也得到了保障.

4.2 内存优化管理

在第 2.2 节中介绍了 RDMA 协议工作的主要流程, 值得注意的是, RDMA 协议在消息传输前需要对一块特定区域的内存进行内存注册, 只有被注册后的内存, RDMA 网卡才能拥有其读写权限. 这在使用过程中将带来一系列的问题.

首先, 内存注册时必须明确告知所需注册内存的位置以及内存大小, 因而给系统带来很大的不便. 在多方安全计算应用程序中, 传输变量往往具有类型多变、消息大、传输轮数多的特点, 因此很难提前预知整个程序执行过程中, 所需传输的消息位置以及大小, 另外这部分消息往往会以密文的形式出现, 这进一步提高了该工作的难度. 一个简单的解决方式是, 在每一轮传输开始前都重新进行内存注册工作. 但它又会带来内存区域注册与释放操作频繁的问题, 正如前文所述多方安全计算任务往往具有通信轮数多和迭代次数多的特点, 这给系统带来较高的资源负担, 并且同一类型的消息可能会被多次重复传输, 而重复内容占据多份内存空间, 导致内存利用效率低.

为了解决上述问题, 充分提高系统内存使用效率, 我们提出一种内存优化使用的方案. 首先, 通过在系统初始化时针对一块较大的内存区域进行内存注册, 该块区域将由后续所有传输变量所共同使用. 然后, 该区域将被划分为许多较小的 buffer 区域, 该区域是 RDMA 传输过程中的最小单位, 它们将被各种类型的传输变量共同重复使用.

在上述内存划分中, 每一次的传输工作都将以 buffer 为基础单元执行, 但实际传输中, 消息的大小并不一定与 buffer 大小相同, 消息的传输过程中对方将无法获取消息的具体大小, 这可能导致消息的序列化和反序列出错, 传输失败. 另外, buffer 的使用与维护也需要仔细考虑.

为了进一步提高内存使用效率, 本文采用了 buffer

环的形式管理内存[15], 如图 6 所示. 在此结构中, 所有 buffer 区域串联成环状结构进行管理维护, 参与方同时拥有发送缓冲环以及接收缓冲环, 每个环又同时拥有一个头指针一个尾指针. 对于发送方来说, 头指针指向的是下一个将要发送的数据 buffer 位置, 发送完成后执行自增操作, 而其尾指针指向那些已经在接收方处理过的缓冲区, 同样处理完成后执行自增操作. 在接收方, 头指针指向的是下一个待接收的消息 buffer 位置, 接收后进行自增操作, 尾指针指向经过处理后可重用的 buffer 位置.

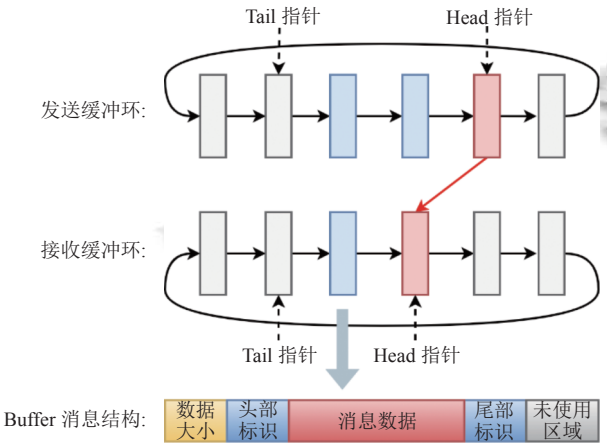

图 6 Buffer 管理模式

另外, 消息的结构也被重新定义, 一个 buffer 里存储的消息应该包含数据大小、头部标识、数据本身、尾部标识以及未使用区域等 5 部分. 接收方在读取该消息后, 会获取数据大小, 如果头部标识已被设置, 则首先根据数据大小计算尾部标识位置, 然后在其后内存区域读取对应大小的数据作为数据接收区. 随后, 继续读取数据并判断是否为尾部标识, 以此判断消息传输是否成功, 否则传输失败. 处理完成后, 清空所有的标识位.

4.3 消息切片

在联邦学习场景中, 为了保护数据隐私, 在数据传输前往往需要对其进行加密操作. 例如, 联邦学习中, 对梯度信息进行同态加密, 其他参与方再在此密文基础上执行计算操作, 完成建模任务. 但若梯度的维度较高, 对加密需要花费很长时间. 另外, 密文数据远大于明文数据, 加密后梯度的传输时间也将大幅度增加. 所以, 数据加密操作和传输操作都将带来很大的时间开销.

传统工作模式中, 数据加密和密文传输送均需由 CPU 串行执行完成, 因此整体任务的执行相对耗时. 然而本方案中, 数据通信过程已经被单独划分, 并交由 RDMA 网卡负责, 那么数据加密和密文数据传输可由不同硬件资源, 从而实现并行化操作.

数据加密和传输, 本身具有一定的时序性, 即对于任意一条数据必须保证其完成加密操作后才能进行数据传输, 因此无法实现并行传输. 为此, 本文采用了消息切片化策略, 通过将高维梯度信息切分为小片信息, 并以片为基本单位实现数据加密和传输操作的并行进行, 提高系统整体资源使用率, 缩短任务的整体时间. 切片处理前后, 消息的传输模式如图 7 所示.

这种方式类似 CPU 指令流水线处理. 假设需要传输的梯度是一个 1 000 维的向量, 在消息切片前, 我们需要对其进行加密操作, 然后才能使用 RDMA 网卡执行消息传输, 整个过程处于串行状态. 如果我们将其切分为 10 个小片段, 每个小片段包含向量的 100 个维度, 之后以片段作为基本单位进行处理. 此时, 当 CPU 在进行第i ($i>1$)片消息的加密操作时, 第$i-1$片消息已经加密完成, RDMA 可以在同一时间段并行的进行第$i-1$片的消息传输. 通过流水化操作, 可以大大缩短消息加密和发送的整体时间.

5 实验评估

5.1 实验平台

本文以两个节点的联邦学习平台为例, 每个节点代表一个风机的数据采集点, 每个节点为 8 核 16 GB 内存物理机器, 配置 Ubuntu 18.04 操作系统. 在软件层上, 部署了隐私计算开源框架 FATE v1.6.0[16], 底层硬件层选用了 Mellanox MCX556A-ECAT 的 InfiniBand 架构网卡. 两台机器部署在局域网环境.

本文实验采用广泛使用的纵向联邦学习 XGBoost 算法, 算法的主要参数配置如表 1 所示. 我们从开源框架 FATE 中选用了 5 个联邦学习经典数据集, 其样本数核特征分布情况如表 2 所示.

5.2 开源框架 FATE

本实验部分基于开源框架 FATE 完成, 它是全球首个联邦学习工业级开源框架, 它使用多方安全计算以及同态加密等多种技术构成底层安全计算协议, 可以支持不同种类的机器学习的安全计算. 因此, 它也可以视为一种统一的执行标准.

用户通过 JSON 格式的 dsl 文件定义一个完整的联邦学习任务流程, 同时利用 JSON 格式的 conf 文件完

成相关参数配置, 之后系统在任务执行时会自动向其他参与方同步任务. 该系统在使用上, 操作简单, 可配置化程度高. 同时它结构化分明, 整个分布式计算和存储由 EggRoll 引擎负责, 这有益于本文工作方案的集成.

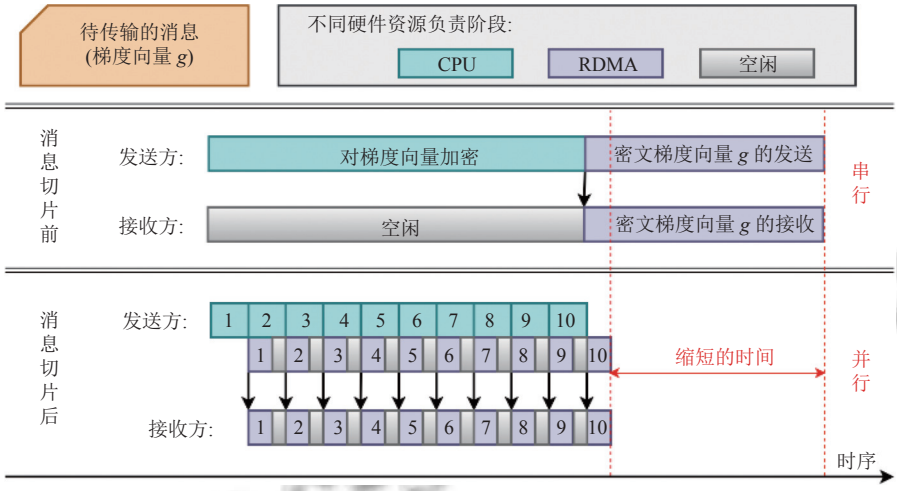

图 7 切片前后消息传输模式对比图

表 1 XGBoost 算法参数设置

参数类型	参数配置
正则项系数	0.01
学习率	0.15
任务类型	Classification
迭代轮数	15
加密策略	InteriveAffine
树的最大深度	3

表 2 数据集设置

数据集	样本数	特征分布	
		Host方	Guest方
ionosphere_scale_hetero	351	15	19
breast_hetero	569	20	10
epsilon_5k_hetero	5000	80	20
default_credit_hetero	30000	10	13
give_credit_hetero	150000	5	5

5.3 通信时间占比分析

在纵向联邦学习 XGBoost 算法上对 5 个纵向分布的开源数据集进行了性能测试. 通过该实验以展现联邦学习任务中任务计算时间与数据通信时间与模型训练总时间的比例情况, 具体实验结果如表 3 所示. 通信时间占比为通信时间比上总时间.

联邦学习建模任务的总时间应该分为 3 个部分: 数据计算时间、消息通信时间以及任务流控制时间. 数据计算时间是指模型训练过程中与计算相关部分, 包括求梯度、模型更新等. 消息通信时间是指隐私消息交互时间, 包括梯度信息的加密传输等. 任务流控制时间是指 FATE 系统进行任务调度的时间, 该部分时间较短且与本文工作关联不大, 因此表格中未给出相关数据.

从实现结果中, 可以发现整个任务通信时间占比严重, 在不同数据集下可以达到 20%–40%. 数据集大小应同时考虑样本数以及总特征数, 在选取的 5 个数据集中, 数据集逐渐增大, 任务各部分时间也呈增长趋势. 在 XGBoost 算法中, 数据集大小主要影响算法直方图的大小, 而计算部分主要发生在直方图构建以及最佳拆分点寻找过程, 通信部分主要受加密后直方图大小影响. 数据集通过影响着直方图构建, 进而影响算法各部分时间, 但由于直方图在传输时会经过一系列加密操作, 再对其密文进行传输. 因此, 数据集大小对通信时间的影响也更加显著.

表 3 XGBoost 算法各部分时间占比情况

数据集	计算时间 (s)	通信时间 (s)	总时间 (s)	通信时间占比 (%)
ionosphere_scale_hetero	536.58	195.96	764.29	25.64
breast_hetero	642.25	339.32	1012.53	33.53
epsilon_5k_hetero	982.83	570.51	1572.96	36.27
default_credit_hetero	975.46	557.53	1549.97	35.97
give_credit_hetero	1249.91	904.74	2182.73	41.45

值得一提的是, 本实验是基于局域网环境完成, 数据通信的时间并未涉及广域网下的链路延迟, 在实际生产环境中, 该部分传输延迟更加严重. 所以, 对联邦学习计算任务通信时间的优化十分必要.

5.4 性能对比分析

为了探究本文工作对联邦学习任务性能的影响,

同样地测试了计算时间、通信时间以及任务总时间这 3 部分. 在此基础上, 将原方案总时间减去加速后任务总时间, 该部分差与加速后任务总时间的比作为任务性能的提升比. 实验结果如表 4 所示.

表 4 XGBoost 算法优化效果对照实验

数据集	原方案 (s)			加速后 (s)			性能提升比 (%)
	计算时间	通信时间	总时间	计算时间	通信时间	总时间	
ionosphere_scale_hetero	536.58	195.96	764.29	163.88	20.16	202.82	276.83
breast_hetero	642.25	339.32	1 012.53	196.91	33.72	261.38	287.37
epsilon_5k_hetero	982.83	570.51	1 572.96	253.96	51.53	333.12	372.18
default_credit_hetero	975.46	557.53	1 549.97	254.36	49.86	337.14	359.74
give_credit_hetero	1 249.91	904.74	2 182.73	361.45	82.47	469.76	364.65

经过优化后, 任务通信时间可以缩短 10 倍左右, 整体性能可以提高 3 倍左右. 性能优化来自于以下几个部分, 首先是采用 RDMA 协议能直接大幅度缩减消息传输时间. 另外, 在 XGBoost 算法中由于树节点的分散存储, 会存在大量小的控制类消息频繁传输, 它们在整体通信时间上占比较低, 但是由于消息的封装处理等工作, 会对 CPU 性能造成较大影响, 从而影响任务整体性能. 而使用 RDMA 协议后, 由于其内核旁路的特性, CPU 将不再负责该部分消息的处理, 任务的整体性能得到一定提高. 最后, 本文工作提出的消息切片策略, 也能够通过计算和通信并行处理的方式提升任务整体性能.

6 总结与展望

随着大数据的进一步发展, 重视数据隐私和安全已经成为世界趋势. 风机等数据如何在隐私安全的环境下发挥最大效益成为风电运营部门面临的主要问题. 联邦学习是应对风电大数据隐私保护的有力手段. 针对联邦学习计算中大量中间数据传输, 造成通信瓶颈的挑战. 本文通过引入 RDMA 协议, 利用其内核旁路的特性提高了联邦学习计算任务的通信效率, 减轻了通信过程对 CPU 造成的资源压力. 最后, 基于开源框架 FATE 对通信库进行了测试, 实验结果显示该通信库对 XGBoost 算法实现了 2–4 倍的性能提升.

本文仅针对特定场景做了初步研究, 未来在多样性算法分析和系统配置上还存在许多值得思考的地方. 另外, 技术的研究离不开实际生产环境, 实验场景与工业场景的迁移运用也值得重点考虑.

参考文献

1 Yao AC. Protocols for secure computations. Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982). Chicago: IEEE, 1982. 160–164.

2 Wright GR, Stevens WR. TCP/IP Illustrated, Volume 2 (paperback): The Implementation. Sebastopol: Addison-Wesley Professional, 1995.

3 Currid A. TCP offload to the rescue: Getting a toehold on TCP offload engines—And why we need them. Queue, 2004, 2(3): 58–65. [doi: 10.1145/1005062.1005069]

4 Von Eicken T, Basu A, Buch V, *et al*. U-Net: A user-level network interface for parallel and distributed computing. ACM SIGOPS Operating Systems Review, 1995, 29(5): 40–53. [doi: 10.1145/224057.224061]

5 Dunning D, Regnier G, McAlpine G, *et al*. The virtual interface architecture. IEEE Micro, 1998, 18(2): 66–76. [doi: 10.1109/40.671404]

6 Recio RJ, Metzler B, Culley P, *et al*. A remote direct memory access protocol specification. RFC 2007. 1–66.

7 郭娟娟, 王琼霄, 许新, 等. 安全多方计算及其在机器学习中的应用. 计算机研究与发展, 2021, 58(10): 2163–2186.

8 Chen TQ, Guestrin C. XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco: Association for Computing Machinery, 2016. 785–794.

9 Luo RQ, Tan X, Wang R, *et al*. Neural architecture search with GBDT. arXiv:2007.04785, 2020.

10 Freund Y, Schapire RE. A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence, 1999, 14(5): 771–780.

11 Cheng KW, Fan T, Jin YL, *et al*. SecureBoost: A lossless federated learning framework. IEEE Intelligent Systems, 2021, 36(6): 87–98. [doi: 10.1109/MIS.2021.3082561]

12 Feng SW, Yu H. Multi-participant multi-class vertical federated learning. arXiv:2001.11154, 2020.

13 RDMA core userspace libraries and daemons. https://github.com/linux-rdma/rdma-core. [2022-02-14].

14 Behnel S, Bradshaw R, Citro C, *et al*. Cython: The best of both worlds. Computing in Science & Engineering, 2011, 13(2): 31–39.

15 Liu JX, Wu JS, Panda DK. High performance RDMA-based MPI implementation over InfiniBand. International Journal of Parallel Programming, 2004, 32(3): 167–198. [doi: 10.1023/B:IJPP.0000029272.69895.c1]

16 FederatedAI/FATE: An industrial grade federated learning framework. https://github.com/FederatedAI/FATE. [2022-02-08].

(校对责编: 孙君艳)