软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(3):1111−1127 [doi: 10.13328/j.cnki.jos.006458]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向多方安全的数据联邦系统*

李书缘[1,2,3], 季与点[4], 史鼎元[1,2,3], 廖旺冬[1,2,3], 张利鹏[1,2,3], 童咏昕[1,2,3], 许 可[1,2,3]

[1](软件开发环境国家重点实验室(北京航空航天大学), 北京 100191)

[2](大数据科学与脑机智能高精尖创新中心(北京航空航天大学), 北京 100191)

[3](北京航空航天大学 计算机学院, 北京 100191)

[4](科学技术部信息中心, 北京 100862)

通信作者: 童咏昕, E-mail: yxtong@buaa.edu.cn

摘 要: 大数据时代, 数据作为生产要素具有重要价值. 因此, 通过数据共享实现大规模数据的分析挖掘与利用具有重要意义. 然而, 近年来日益严格的隐私安全保护要求使得数据分散异质的多方之间不能任意共享数据, 加剧了"数据孤岛"问题. 数据联邦能让多数据拥有方在保护隐私的前提下完成联合查询. 因此, 基于"数据不动计算动"的联邦计算思想实现了一种多方安全的关系型数据联邦系统. 该系统适配多种关系型数据库, 能够为用户屏蔽底层多数据拥有方的数据异构性. 系统基于秘密共享实现了支持多方安全的基础操作多方安全算子库, 优化了算子的结果重建过程, 提高了其执行效率. 在此基础上, 系统支持求和、求均值、求最值、等值连接和任意连接等查询操作, 并充分利用多方特点减少各数据拥有方之间的数据交互, 降低安全开销, 从而有效支持高效数据共享. 最后, 在标准测试数据集 TPC-H 上进行实验, 实验结果说明: 与目前的数据联邦系统 SMCQL 和 Conclave 相比,该系统能够支持更多的数据拥有方参与, 并且在多种查询操作上有更高的执行效率, 最快可超越现有系统 3.75 倍.

关键词: 数据联邦; 数据库系统; 安全多方计算

中图法分类号: TP311

中文引用格式: 李书缘, 季与点, 史鼎元, 廖旺冬, 张利鹏, 童咏昕, 许可. 面向多方安全的数据联邦系统. 软件学报, 2022, 33(3): 1111−1127. http://www.jos.org.cn/1000-9825/6458.htm

英文引用格式: Li SY, Ji YD, Shi DY, Liao WD, Zhang LP, Tong YX, Xu K. Data Federation System for Multi-party Security. Ruan Jian Xue Bao/Journal of Software, 2022, 33(3): 1111−1127 (in Chinese). http://www.jos.org.cn/1000-9825/6458.htm

Data Federation System for Multi-party Security

LI Shu-Yuan[1,2,3], JI Yu-Dian[4], SHI Ding-Yuan[1,2,3], LIAO Wang-Dong[1,2,3], ZHANG Li-Peng[1,2,3], TONG Yong-Xin[1,2,3], XU Ke[1,2,3]

[1](State Key Laboratory of Software Development Enviroment (Beihang University), Beijing 100191, China)

[2](Beijing Advanced Innovation Center for Big Data and Brain Computing (Beihang University), Beijing 100191, China)

[3](School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

[4](Information Center, Ministry of Science and Technology, Beijing 100862, China)

Abstract: In the era of big data, data is of great value as an essential factor of production. It is of great significance to implement its analysis, mining and utilization of large-scale data via data sharing. However, due to the heterogeneous dispersion of data and increasingly

* 基金项目: 国家重点研发计划(2018AAA0101100); 国家自然科学基金(61822201, U1811463, 62076017, 61690202); 北京市科技计划(Z191100002519012); CCF-华为数据库创新研究计划(CCF-HuaweiDBIR2020008B); 软件开发环境国家重点实验室(北京航空航天大学)开放课题(SKLSDE-2020ZX-07)

本文由"数据库系统新型技术"专题特约编辑李国良教授、于戈教授、杨俊教授和范举教授推荐.

收稿时间: 2021-06-30; 修改时间: 2021-07-31; 采用时间: 2021-09-13; jos 在线出版时间: 2021-10-21

rigorous privacy protection regulations, data owners can not arbitrarily share data. This dilemma turns data owners into data silos. Data Federation calculate collaborative query while preserving the privacy of data silos. This study implements a multi-party secure relational data federation system. The system is designed based on the idea of federated computation that “data stays, computation moves”. Its adaptation interface of the system is different kinds of relational database adaptation, which can shield the data heterogeneity of multiple data owners. The system implements the multi-party security basic calculator library based on secret sharing, and the calculator realizes the optimization of the result reconstruction process. On this basis, it supports the query operations such as sum, average, maximum, equi-join and theta-join. Making full use of the multi-party properties to reduce the data interaction among data owners, the proposed system reduces the security computation overhead, so as to effectively support efficient data sharing. Finally, the experiment is carried out on the benchmark data set TPC-H. The experimental results show that the proposed system can support more data owners’ participation and has higher execution efficiency than current data federation systems such as SMCQL and Conclave by at most 3.75 times.

Key words: data federation; database system; secure multi-party computation

随着大数据时代的到来和飞速发展, 数据作为生产要素, 在各产业应用中具有愈发重要的价值, 实现数据共享具有重要意义.《中共中央国务院关于构建更加完善的要素市场化配置体制机制的意见》中指出, 国家要加快培育数据要素市场. 具体强调了要推进数据开放共享, 培育数字经济新产业. 足以说明实现数据共享、挖掘数据价值对未来国家经济社会发展具有重要的战略意义. 数据共享带来的好处是显而易见的, 例如,一名患者在多家医院都有病历与检查报告等存档, 若各医院之间能够共享这些数据, 不仅帮助患者避免重复检查, 还能对患者病史等情况有更为全面的了解; 再如, 政务数据分割存储在不同政府部门的信息系统中, 若打破部门壁垒实现政务大数据的整合利用, 则能优化政务服务.

然而, 现实情况中数据共享受到严重限制. 这是由于数据分散且异质地存在于大量个体之间, 造成了数据汇聚极为困难, 限制了数据的共享利用, 被称为“数据孤岛”问题. 为解决此问题, 产生了相应的数据集成技术[1]. 然而, 近年来数据隐私安全保护已经成为国内外的广泛共识. 例如, 在 2018 年 5 月 25 日, 欧盟出台了《通用数据保护条例》(general data protection regulation, GDPR); 在 2021 年 4 月 30 日, 全国人大常委会公布了《个人信息保护法(草案二次审议稿)》与《数据安全法(草案二次审议稿)》. 以上监管法案均对数据的处理与流通等做出限制, 使得数据共享难以广泛落实应用.

日益严格的隐私保护需求进一步加剧了“数据孤岛”现象, 各数据拥有方只能使用自身所持有的小规模数据, 数据难以通过共享融合汇聚, 不能很好地发挥自身价值. 只有打通数据间孤岛, 破除数据间共享壁垒, 才能真正使数据作为生产要素有效地驱动经济社会发展[2]. 因此, 联邦计算这一概念应运而生. 其中, 联邦这一概念是彼此独立自治的数据拥有方的集合. 联邦中的数据拥有方为保护隐私, 保证原始敏感数据不离开本地. 联邦计算中, 保护隐私的核心思想即为“数据不动计算动”, 各数据拥有方不直接共享数据, 而是先在各方对数据进行计算, 将所得中间计算结果进行汇总, 从而得到最终计算结果. 这一模式将计算拆分至各方, 从而避免了数据从各方本地流出, 因此, 联邦计算“数据不动计算动”的思想可以指导保护隐私要求下的数据共享模式. 在联邦计算中“数据不出本地”这一前提下, 构建上述数据共享模式下的数据联邦系统中还存在以下挑战.

- 数据共享的隐私安全保障难. 有效的数据共享离不开多数据拥有方的参与, 然而让联邦系统实现对多数据拥有方的支持, 对底层安全操作设计提出挑战.
- 数据共享的高效安全查询难. 联邦场景中, 数据共享过程需要保护数据隐私安全, 为此需要对联邦系统进行安全设计, 然而安全计算代价高昂, 保障大规模数据的高效共享同样构成挑战.
- 数据共享的异构多方协作难. 在数据共享场景下, 各数据拥有方存在异构性问题, 包括数据库系统异构、数据模式异构等. 如何在保护隐私的前提下对异构多方完成适配?

目前尚未有系统能够较好地解决上述 3 个挑战. 首先, 在 20 世纪 80 年代出现过的联邦数据库, 这类数据库本质是作为中间件协调多个数据库共同完成某一查询. 然而这些系统侧重解决多方数据库的异构问题与联合完成查询的拆解重写方式, 不关注联合查询中的隐私问题. 直到 21 世纪初, 一些便于开发使用的安全多方计算工具库[3,4]陆续开源. 于是, 自 2017 年, 数据联邦系统开始发展. 不同于联邦数据库, 后者更加强调对数据拥有方的数据隐私安全保护. 一些学者尝试将安全多方计算技术与联邦思想结合, 构建保护隐私的数据联

邦系统[5,6]. 然而这些工作受限于所采用的安全多方计算工具库, 仅能支持 2–3 方数据拥有者参与.

针对上述挑战, 我们实现了一种多方安全的数据联邦系统. 本系统由系统适配、多方安全算子库、查询引擎和交互接口组成, 能够支持多种异构数据库接入, 支持包括任意连接在内的多种多方安全查询操作, 并具备面向用户的统一 SQL 与图形化界面, 从而实现高效的多方安全数据共享. 其主要贡献如下:

- 构建了面向多方安全的数据联邦系统. 该系统能够实现支持 3 方以上的多数据拥有方参与的安全数据共享, 同时其配有适配接口, 能够屏蔽各数据拥有方数据库系统的异构性. 系统还提供用户友好的图形化界面, 支持 SQL 查询并适配多种查询接口.
- 实现了支持加减乘除和比较在内的高效多方安全算子库. 算子库实现基于秘密共享框架, 涵盖的基础操作可以支持基本的数据查询操作. 针对秘密共享中的特点, 实现了高效的结果重构.
- 实现了支持包括任意连接在内的高效多方安全查询引擎. 基于前述基础算子, 查询引擎能够支持求和、求平均、求最值、等值连接和任意连接在内的查询操作. 同时, 在设计过程中充分考虑多方特点, 利用连接过程的传递性等特点, 保障查询操作的高效性.

本文第 1 节介绍本系统的有关背景知识. 第 2 节介绍本系统包括系统构架与系统工作流. 第 3 节和第 4 节分别介绍系统的多方安全算子库和查询引擎. 第 5 节介绍系统适配和交互接口. 第 6 节展示在标准测试集上的系统性能验证结果. 第 7 节介绍相关工作. 最后在第 8 节对本文进行总结与展望.

1 背景知识

在数据隐私安全保护要求日益严格的背景下, “数据孤岛”问题日益严重. 为打通数据孤岛实现数据共享, 一个重要思想就是数据联邦. 具体来说, 数据联邦$\mathcal{F}$可视为 n 个数据拥有方和中心服务器 C 共同组成的集合, 其中, 数据拥有方个数 $n\geqslant 3$. 在数据联邦中, 不同数据拥有方的数据库是异构的. 此外, 就实际应用场景而言, 数据拥有方数量 n 通常大于 3. 例如在流行病学调查过程中, 可能需要医院、地图应用、移动支付和网约车平台等多数据拥有方的合作. 又如出租车联盟中, 参与联盟的出租车公司数量可能在十几到几十家不等. 在某些场景中, 数据联邦中心服务器的计算任务可由数据拥有方轮流承担.

对数据联邦中的第 i 个数据拥有方 P_i, 假设所拥有的数据为 d_i, 各数据拥有方遵循的安全模型假设为半诚信模型(semi-honest)[7]. 在半诚信模型假设中, 参与方是“诚实但好奇(honest-but-curious)”的, 即参与方会诚实地遵守协议和需要执行的代码, 但是会根据执行过程中获得的数据进行推断, 如果能够推断出本应该被保护的信息, 则说明系统是不安全的. 半诚信模型是安全领域被广泛应用的假设, 并且该假设已被现有代表性的多方安全数据库系统采用[5,6].

为实现联邦场景下的数据共享, 数据联邦系统需支持多方安全的联邦查询操作. 定义如下:

定义 1(联邦查询操作). 对某一联邦查询操作 $f:D^n\to R$, 其计算结果与对应的传统数据库查询操作结果 $f':D\to R$ 应一致, 即 $f(d_1,d_2,\ldots,d_n)=f'(d_1\cup d_2\cup\ldots\cup d_n)$. 且在计算中, 不应将数据拥有方 P_i 的数据 d_i 泄露给任意其他数据拥有方.

其中, 基本的联邦查询操作包括联邦求和(f-SUM)、联邦求平均(f-AVG)、联邦求最值(f-MIN/MAX)、联邦等值连接(f-equi-join)和联邦任意连接(f-θ-join)等. 这些查询操作一方面应支持多方异构数据库, 另一方面应保证安全. 即: 各数据拥有方在共同完成计算时, 保证每个数据拥有方的原始数据不出本地. 在该安全保证下进行数据共享, 要求执行查询操作的过程中, 各数据拥有方不能将其原始数据直接发送给其他方进行计算. 保证了各数据拥有方联合在完成查询操作后, 能够得到该查询操作的计算结果, 但不能得到其他数据拥有方的原始数据, 从而保护各方数据隐私.

下面结合上述定义举出医疗领域的应用案例. 例如, 一名患者在多家医院都有病历与检查报告等存档, 若各医院之间能够共享这些数据, 则不仅可以帮助患者避免重复检查, 还可以对患者病史等情况有更为全面的了解. 在该应用案例中, 每一家医院作为数据拥有方具有大量患者医疗数据, 共同构成一个数据联邦. 多家医院为保护患者数据的隐私性, 需进行多方安全数据共享, 如对肝功能检测数据表根据患者身份证号进行联

邦等值连接(f-equi-join)再进行联邦求平均(f-AVG), 即可在患者数据不出本地的前提下, 查询患者以往肝功能检测指标的均值.

上述场景中, 系统设计的主要目标是由于联邦场景带来的数据安全目标和由此带来的计算开销控制的效率目标, 以及多方数据库的异构性带来的可用性目标. 以下安全目标、效率目标与可用性目标这 3 个系统目标分别对应前述提出的隐私安全保障难、高效安全查询难和异构多方协作难这三大挑战.

- 安全目标. 在联邦场景中, 多方相互不可信, 数据不能离开本地. 并且其他数据拥有方可能会根据公开的信息进行推断. 为此, 除了查询语句和查询结果对所有数据拥有方公开以外, 查询用户和所有方都不能根据这些公开信息推断出其他任何额外信息. 例如, 在多方安全求和操作中, 用户和所有数据参与方应该只知道最终求和结果, 但不能推断每方各自参与求和的准确数值. 这样可以避免数据拥有方的数据在计算过程被窃取. 这是多方安全数据库系统中对查询操作的常见要求[5,6].
- 效率目标. 在联邦场景中, 查询操作的计算开销主要有两个来源: 首先, 安全操作保护的对象是数据, 因此保护数据量大, 特别是在连接操作中, 保护对象为数据列, 庞大的数据量导致高昂的保护代价; 其次, 数据联邦场景中的查询操作需要多方参与, 解决各数据拥有方的异构性问题与多方协同问题也会带来开销. 为保证系统运行效率, 需针对上述开销来源设计优化模块. 其中, 对于因为保护数据量大造成的开销, 需要设计有关的高效安全算子等进行效率优化, 提高运行效率; 对于多方协调造成的开销, 应该巧妙利用多方的并行性实现效率提升. 系统的运行效率将影响其在应用中的性能表现, 例如, 在出租车联盟中, 运行效率决定了联盟平台对订单的响应时间; 在流行病学调查中, 系统运行效率决定了病例溯源与密接排查的耗时.
- 可用性目标. 联邦场景中, 各数据拥有方的数据库系统存在异构性, 因此, 本系统应实现面向异构数据库的适配统一. 此外, 为方便多方协调, 应构建面向用户的统一 SQL 查询操作接口和图形化管理界面.

本系统针对上述目标实现, 具体架构和流程设计详见第 2 节.

2 系统概述

2.1 系统架构

本系统架构如图 1 所示.

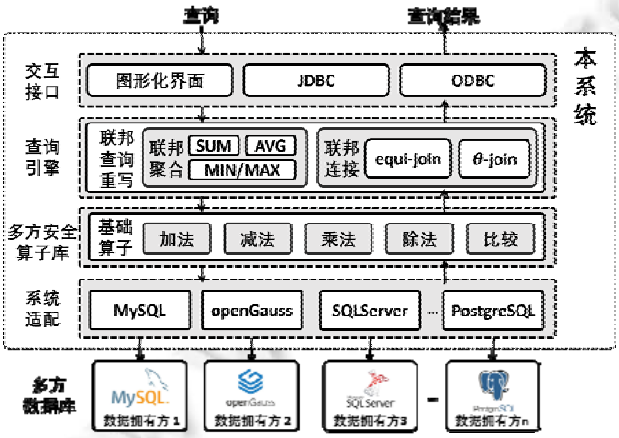


图 1 基于安全多方计算的数据联邦系统框架

系统共分为 4 层, 自底层多方数据库向上分别依次为系统适配、多方安全算子库、查询引擎和交互接口. 底层即数据联邦中不同数据拥有方的数据库, 为屏蔽数据库的异构性, 为不同种类的数据库编写适配模块完

成系统适配, 对应第 1.2 节中的可用性目标设计. 在系统适配上则是多方安全算子库与查询引擎, 这是系统的核心模块, 分别针对第 1.2 节中的安全目标与效率目标而设计, 使系统能够支持安全高效的多方数据共享. 交互接口中的图形化界面与 SQL 查询接口则主要为便于多方用户协调, 和系统适配共同完成了第 1.2 节的可用性设计目标. 各层的具体功能如下.

- 本系统底层面向各数据拥有方, 由各数据拥有方的数据库系统构成. 考虑到各数据拥有方其数据库系统的异构性, 本系统可支持多种数据库系统.
- 系统适配: 本系统适配承接上层多方安全算子库下发的各类查询计算, 适配底层多方异构数据: 首先适配不同数据库系统, 目前已对 MySQL、openGauss、SQL Server 与 PostgreSQL 等多种数据库系统完成适配; 其次适配不同数据模式, 构建面向用户的统一数据视图.
- 多方安全算子库: 本算子库主要包含多方安全的基础算子, 各个基础算子主要实现多数据拥有方共同参与的简单计算, 例如多方安全的求和算子与多方安全的求积算子等. 其计算过程中会通过适配接口调用各数据拥有方查询本地数据库. 多方安全算子库是构建后续联邦查询操作的基础.
- 查询引擎: 本查询引擎承接上层交互接口得到的查询, 对查询进行解析并将其重写为对应的面向联邦的查询操作. 涉及多方联合计算的主要有联邦聚合与联邦连接两类, 其中, 联邦聚合操作基于下层计算层, 支持 SUM、AVG 和 MIN/MAX 这 3 种聚合计算; 联邦连接支持 equi-join 与 θ-join 两种.
- 交互接口: 本交互接口向用户提供图形化界面, 提升系统易用性. 接收用户通过界面输入的查询并将该查询传递给下层查询引擎. 本系统支持用户使用 SQL 查询语句.

如前所述, 本系统中的多方安全算子库和查询引擎属于系统核心, 其具体实现将分别在第 3 节和第 4 节介绍.

2.2 系统工作流程

本系统工作流程如图 2 所示. 用户首先根据系统提供的统一视图输入 SQL 查询. 系统中心服务器接收该查询后将其解析为语法树, 并对 SQL 查询进行谓词下推优化, 即将过滤条件下推至距数据源更近. 然后针对其中涉及多个数据拥有方联合参与的查询操作, 将其重写为多方安全的联邦查询操作方法. 随后, 服务器将重写的联邦查询操作的执行计划发送给各个数据拥有方, 数据拥有方按照重写的流程执行, 主要分为本地计算与多方交互两部分: 首先执行其中对本地数据库的查询操作, 并得到对应的查询结果; 其次, 使用多方安全基础算子, 按照协议执行涉及多方联合计算的部分. 最后将得到的联邦查询结果发送至服务器端, 通过使用秘密共享与安全集合求并等方法, 在保护各联邦查询结果来源的前提下得到最终查询结果, 并将该结果返回给用户.

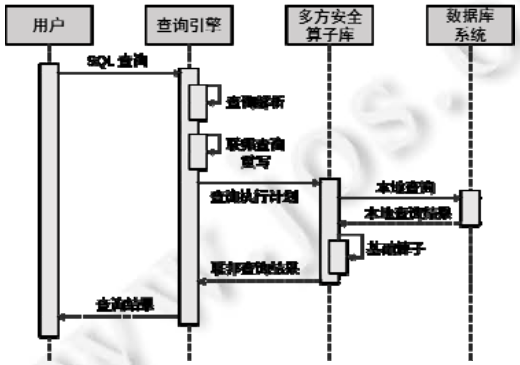


图 2 本系统查询处理流程图

3 多方安全算子库

本系统多方安全算子库支持面向多方的安全基础计算, 涵盖了加减乘除与比较运算, 实现对系统的多种

查询操作的支持. 各算子通过基于秘密共享的安全多方计算技术保障数据安全. 本系统采用秘密共享这一安全多方计算技术而非混淆电路, 主要是因为秘密共享能够较好地实现对多方的支持. 目前已有的数据联邦系统如 SMCQL[5]和 Conclave[6]的开源部分都是基于混淆电路工具库如 ObliVM[3]或 Obliv-C[4]实现的, 均不支持 3 方以上的数据拥有方参与计算. 而基于秘密共享的安全计算协议对多方参与这一要求支持良好. 秘密共享技术工具库主要有 Sharemind[8]和 MP-SPDZ[9]: Sharemind 工具库并未开源, 限制了其应用; MP-SPDZ 工具库的接口封装主要面向计算, 其多方交互模式难以与数据联邦系统兼容. 因此, 需要为本系统定制基于秘密共享的多方安全的基础算子库. 本文分别在第 3.1 节介绍该算子库的基本框架, 在第 3.2 节介绍算子具体执行流程, 并以求和算子为例详细阐述.

3.1 算子库框架

本系统主要基于 *Shamir*(*t*,*n*)模式[10]的秘密共享来设计实现基础算子. 该模式保证当某一秘密分散在 n 个数据拥有方时, 需要至少 t 个数据拥有方参与才能共同重构出该秘密. 基于该模式设计的多方安全算子库框架主要分为 3 个阶段: 秘密分发、本地计算与结果重构. 下面分别针对 3 个阶段进行阐述.

- 秘密分发. 在秘密分发阶段, 首先为各数据拥有方分别分配编号; 随后根据设定的阈值 t, 每方均随机生成一个最高次数项为阈值的多项式, 并将该方的输入数据作为多项式常数项; 接着, 把各数据拥有方对应的编号带入到多项式计算得到其子秘密, 并发送给该编号的数据拥有方.
- 本地计算. 各数据拥有方在收到其他所有方发送的子秘密之后, 根据不同基础算子对子秘密进行对应运算, 运算所得即为最终结果的子秘密.
- 结果重构. 任意不小于 t 个数据拥有方将其在上一阶段中所得最终结果的子秘密发送至中心服务器, 由于最终目标结果是高次多项式的常数项, 且该多项式次数随参与数据拥有方的数量增多而增高, 中心服务器接收的子秘密为该多项式上的点. 所以由中心服务器使用拉格朗日插值法, 根据子秘密求解出该高次多项式, 即可重构出基础算子的最终结果.

以上3个阶段中, 结果重构阶段运行开销最大, 是多方安全算子库的瓶颈所在. 这是因为该阶段使用拉格朗日插值法求解高次多项式, 产生了如下两个问题: 首先, 结果重构阶段需要求解高次多项式, 导致计算开销大, 而在秘密分发阶段与本地计算阶段仅涉及简单的随机数生成与多项式计算; 其次, 常用的拉格朗日插值法求解高次多项式时存在误差问题, 并且误差一般随多项式次数增高而增大. 针对以上两问题, 多方安全算子库根据分治思想改进, 得到如下计算框架: 针对 n 个数据拥有方参与的基础算子, 首先将 n 个数据拥有方划分为若干小组; 然后在每个小组内, 通过以上 3 个阶段得到该小组的计算结果作为基础算子的中间结果; 紧接着, 将各小组得到的中间结果再通过以上3个阶段得到新的中间结果. 重复该过程, 直至得到算子的最终结果. 该框架通过分组的方式减少每次多方安全计算的参与方数量, 从而降低其中多项式的次数, 能显著减少基础算子的误差, 同时提升算子的运行效率.

3.2 算子执行流程

基于以上多方安全算子库框架, 本系统算子库实现了加减、乘除与比较等基础算子, 以加法算子为例进行如下详细介绍. 算法 1 阐述了加法算子的计算过程, 其中, 第 1 行–第 4 行描述秘密分发阶段的执行步骤,各方随机生成 t–1 次多项式, 并代入编号计算子秘密然后分发; 第 5 行和第 6 行描述本地计算阶段的步骤, 对于加法算子, 该阶段每方对子秘密求和即可; 第 7 行描述结果重构阶段的步骤, 根据公式(1)对 t 个 R_i 分别代入公式进行计算, 即可得到最终结果 R.

$$R=\sum_{i=1}^{t}R_i\cdot\prod_{j=1,j\neq i}^{t}\frac{x_j}{x_j-x_i} \tag{1}$$

算法 1. 联邦求和算子.

Input: 数据拥有方 $P_1,P_2,\ldots,P_n$ 的数据 $d_1,d_2,\ldots,d_n$ 与编号 $x_1,x_2,\ldots,x_n$; 阈值 t.

Output: 各方求和结果 R, 其中, $R=d_1+d_2+\ldots+d_n$.

1: **for** 数据拥有方 P_i **do**
2: 随机生成多项式 $f_i(x)=d_i+a_1x+a_2x^2+\ldots+a_{t-1}x^{t-1}$
3: 计算子秘密 S_i,其中, $S_i[j]=f(x_j)$, $1\leqslant j\leqslant n$
4: 分发子秘密, 将 $S_i[j]$发送给 P_j
5: **for** 数据拥有方 P_i **do**
6: $R_i\leftarrow\sum S_j[i]$, $1\leqslant j\leqslant n$, $j\neq i$
7: 汇总任意 t 方的 R_i, 根据公式(1)解得最终结果 R
8: **return** R

图 3 展示了 3 个数据拥有方参与的加法算子运行流程. 每个数据拥有方 的数据为 d_i, 编号为 x_i. 每个数据拥有方首先随机生成两个随机数 a_{i1} 与 a_{i2}, 并得到随机多项式 f_i. 然后将编号代入 f_i 计算, 并将子秘密 $f(x_i)$发送给第 i 方, 完成秘密分发阶段. 在本地计算阶段, 每一数据拥有方将所得子秘密求和, 即 $f_1(x_i)+f_2(x_i)+f_3(x_i)$, 相当于构造了以 $\sum d_i$ 为常数项的多项式 S. 最后, 结果重构阶段即使用 $S(x_1)$, $S(x_2)$, $S(x_3)$求解多项式 S, 解得多项式 S 后, 常数项即为求和算子结果.

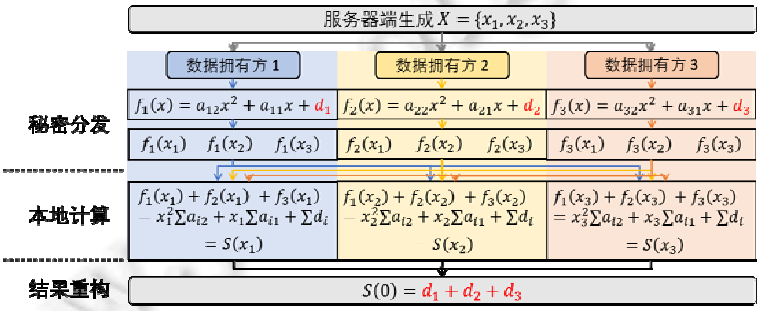


图 3 多方安全算子库求和算子计算流程示意

4 查询引擎

4.1 设计原则

与传统数据库系统不同, 本系统查询面向多个数据拥有方构成的数据联邦, 需保护各方数据安全隐私. 所以在查询处理过程中, 当某一计算过程只涉及数据拥有方自身的数据时, 则可以在该方本地直接明文进行查询, 无须考虑该方数据的隐私安全问题. 而当某一计算有各数据拥有方共同参与需进行多方交互时, 则要根据安全协议执行大量安全操作, 以保护每方数据隐私安全. 所以在本系统中, 多方交互的效率是查询引擎性能的关键所在. 因此, 本系统查询引擎以减少多方交互开销为设计理念, 针对查询处理流程中查询解析、查询计划生成与查询计划执行的三步骤开展设计. 本系统查询引擎设计了相应的解析单元、重写单元与执行单元完成上述流程. 以上 3 个单元的具体设计如下.

4.2 解析单元设计

解析单元主要接收用户通过接口所输入的 SQL 查询, 由中心服务器承担其计算任务. 该单元首先对输入的 SQL 查询进行解析来得到查询语法树. 根据查询引擎的设计理念, 为减少多方交互开销, 对所得语法树执行谓词下推操作. 例如对过滤条件进行下推, 使得过滤条件可以在数据拥有方本地尽早明文执行, 无须考虑安全问题, 从而减少后续涉及多个数据拥有方参与的查询操作中数据量, 以降低后续多方交互时的开销. 图 4 以具体实例展示了解析单元的流程与效果, 通过下推操作减少了多方安全交互涉及的数据量与计算开销.

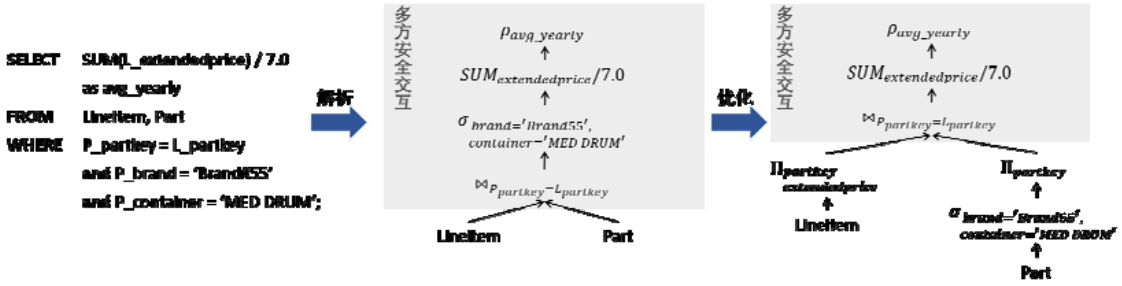


图 4 本系统解析单元工作流程

4.3 重写单元设计

重写单元设计承接前一解析单元输出下推后的语法树，并根据语法树生成查询执行计划. 由于语法树上某些节点的查询操作需要多个数据拥有方的共同参与，同时考虑保护各方的数据隐私安全，因此重写单元根据查询引擎减少多方交互开销的设计理念，主要将涉及多方的查询操作重写为保护多方数据隐私安全的联邦查询操作. 下面以数据库经典连接操作 JOIN 为例，介绍重写单元设计.

- 联邦 JOIN 查询操作

由于本系统底层多方数据库存储对用户透明，因此用户可根据中心服务器提供的统一视图，发起传统连接查询操作 $L.c \triangleright\triangleleft_{condition} R.c$，表示将 L 表与 R 表中基于数据列 c 列满足 *condition* 条件的元组相连接. 而重写单元需面向底层多个数据拥有方，考虑用户视角的 L 表与 R 表实际分散在多方. 由于多个数据拥有方分别持有 L 表与 R 表的一部分，因此在多方联合计算连接操作时，需要其中任意两方均进行一次连接操作. 然后将所有得到的连接中间结果汇总，得到最终的连接结果. 因此，联邦查询操作的全局结果由多个两方连接结果汇总所得. 下面以其中某两方进行连接操作的流程为例，简单介绍在数据联邦场景下如何进行重写.

- 两方 JOIN 查询操作

两个数据拥有方进行保护数据隐私的安全连接时，数据拥有方 P_1 持有 L 表的部分数据 L_1，数据拥有方 P_2 持有 R 表的部分数据 R_2. 直接将整个数据表 L_1，R_2 作为安全协议的输入进行连接，会导致两方交互的开销极大[5]. 因此，根据查询引擎的设计理念，本重写单元针对此高开销问题重写连接操作流程，减少两方计算的通信量. 首先，在数据拥有方本地对数据列 c 列进行明文排序；然后对于数据表 L_1 中数据列 c 列第 i 行的值，与数据表 R_2 中数据列 c 列中第 j 行的值进行安全的比较计算，判断是否符合 *condition* 条件. 即可在不向对方泄露数据值的前提下，判断两数据表中第 i 行与第 j 行对应的元组是否需要进行连接. 若安全比较结果判定符合 *condition* 条件，则可将该元组发送至中心服务器，由中心服务器对两元组进行连接即可；若安全比较结果判定不符合 *condition* 条件，则两元组无须连接. 继续重复以上步骤，将数据表 L_1 中数据列 c 列第 i 行的值与数据表 R_2 中数据列 c 的其他行进行比较，直至数据表 L_1 中数据列 c 上所有值均与数据表 R_2 完成比较.

通过以上流程设计，降低了双方交互的开销，使得两个数据拥有方只有数据表中的一个数据列参与安全比较协议的计算. 本系统根据传递性思想，进一步减少双方连接的交互开销. 由于连接操作的数据列在两方本地是有序的，因此对于数据表 L_1 中数据列 c 列第 i 行的值，无须遍历 R_2 中数据列 c 列的所有值，可通过二分搜索定位. 通过二分搜索减少双方连接时需要进行的安全比较次数，从而降低双方交互开销.

- 全局 JOIN 查询操作

以上重写流程能够安全完成两个数据拥有方之间的连接操作. 然而联邦连接查询中，每一数据拥有方 P_i 均持有数据表 L_i，R_i，因此还需设计如何由多个的两方连接汇总得到全局连接操作结果. 本单元同样根据传递性的思想，减少汇总过程中的多方间交互的通信量. 以等值连接为例，当数据拥有方 P_1 分别与数据拥有方 P_2、数据拥有方 P_3 均进行两方的安全连接操作之后，则数据拥有方 P_1 可根据两次连接结果的交集，推断出该交集部分既存在于数据拥有方 P_2 同时也存在于数据拥有方 P_3. 那么数据拥有方 P_2 与 P_3 进行安全连接操作时，则无须对该部分再重复进行安全比较. 通过该方式组织多方进行连接操作，可减少两方间参与安全操作的计算

量, 减少交互开销.

4.4 执行单元设计

执行单元承接前述单元制定的联邦查询计划, 调度各个数据拥有方执行该计划. 计划中的每一联邦查询操作均在上一重写单元被重写, 则这些联邦查询操作在执行时需调用算子库中对应的多方安全基础算子. 例如, 针对联邦 JOIN 查询操作, 两方进行连接时需调用比较算子来判定是否符合连接的约束条件; 针对联邦 SUM 查询操作, 各数据拥有方本地执行 SUM 操作之后, 需对所得值调用安全求和算子, 该算子计算结果为联邦 SUM 查询操作的结果.

前述单元设计的基础算子与查询操作的流程, 具有良好的可并行性. 在本执行单元中, 调度各数据拥有方在对应流程中并行化部分计算操作, 以提升系统的运行效率. 对于基础算子的执行过程, 由于多方安全算子库中算子均基于秘密共享的框架设计, 因此各方根据分治思想分组进行算子的计算, 所以以组为单位执行算子计算时均可并行; 对于联邦聚合查询操作, 各数据拥有方首先在本地执行对应数据库查询操作, 该本地计算可以各方为单位并行执行; 其次, 各方调用算子库算子进行多方中间结果的聚合, 该步骤可使用前述基础算子的并行化; 对于联邦连接查询操作, 由于全局连接结果由多个两方接连结果汇总构成, 因此每两方执行连接操作的过程均可以两方为单位并行化.

4.5 查询操作的安全性分析

本系统作为面向数据联邦的多方安全数据联邦系统, 需保护各数据拥有方的原始数据不出本地, 即各方执行查询操作流程后可以得知结果, 但不能得知其他方的原始数据. 对基础算子、聚合操作与连接操作的安全性分析如下.

- 基础算子安全性分析. 本系统多方安全算子库基于秘密共享的安全多方计算技术实现, 根据安全多方计算定义, 此类算子对函数 $f(x_1,x_2,\ldots,x_n)$, 保证各方不获取其他方 x_i 的要求下得到函数计算结果[11]. 因此, 在本系统多方安全算子库中的操作能保证各数据拥有方的数据不泄露给其他方, 同时完成计算要求得到计算结果, 符合数据联邦场景下的安全要求.
- 联邦聚合查询操作安全性分析. 本系统支持聚合操作, 此类操作计算流程主要分为本地计算与多方交互两部分: 本地计算部分由各数据拥有方自治数据完成计算, 不涉及数据隐私泄露的安全问题; 多方交互部分操作直接调用基础算子进行聚合计算, 并得到最终聚合结果. 因此, 多方交互部分的安全性由基础算子进行保障, 已在前续段落进行分析.
- 联邦连接查询操作安全性分析. 本系统支持连接操作, 主要将其计算流程拆分为所有方两两进行连接后, 汇总全局连接结果. 在两方进行连接时, 首先对连接基于的数据列调用安全比较算子来判定是否符合连接条件, 在该判定过程中, 安全比较算子保证参与双方在不获知对方数据前提下得到判定结果; 然后, 双方根据判定结果将符合条件的元组发送给中心服务器进行连接, 这些元组作为连接结果也无须保护, 符合系统安全要求. 在汇总全局连接结果时, 由于该计算只涉及各方所得连接结果, 且这些连接结果均包含在最终全局连接结果内, 因此汇总过程也无须进行保护. 而当联邦连接操作后续紧跟聚合操作时, 各方在获得判定结果后不将元组发送给中心服务器进行连接, 而是直接在判定符合连接条件的元组上进行联邦聚合操作, 避免连接结果泄露. 因此, 联邦连接查询操作尽可能地避免泄露原始数据.

5 系统适配与交互接口

5.1 系统适配

本系统针对底层多个数据拥有方间的数据异构问题, 提供面向不同数据库系统的适配接口与不同数据表模式的适配接口. 对底层数据拥有方不同的数据库系统, 本系统已适配如 MySQL、openGauss、SQL Server 与 PostgreSQL 等多种数据库系统的操作接口. 对底层数据拥有方不同的数据表模式, 本系统在视图构建步骤

为各数据拥有方提供数据表模式的上传接口，各方将参与数据联邦共享的数据模式通过接口传至中心服务器.中心服务器根据各方数据模式构建面向用户的统一数据视图，并构建统一视图与各方数据模式间的映射关系，以对用户屏蔽底层数据异构性.

5.2 交互接口

本系统为用户提供易于使用的图形化界面，该界面功能主要有两个：一是向用户展示数据联邦的统一数据视图，供用户基于本数据视图进行查询；二是向用户提供查询操作功能，本系统支持 SQL 语句查询，用户在对应操作框内输入 SQL 查询，即可对多方数据进行联合查询(如图 5 所示).

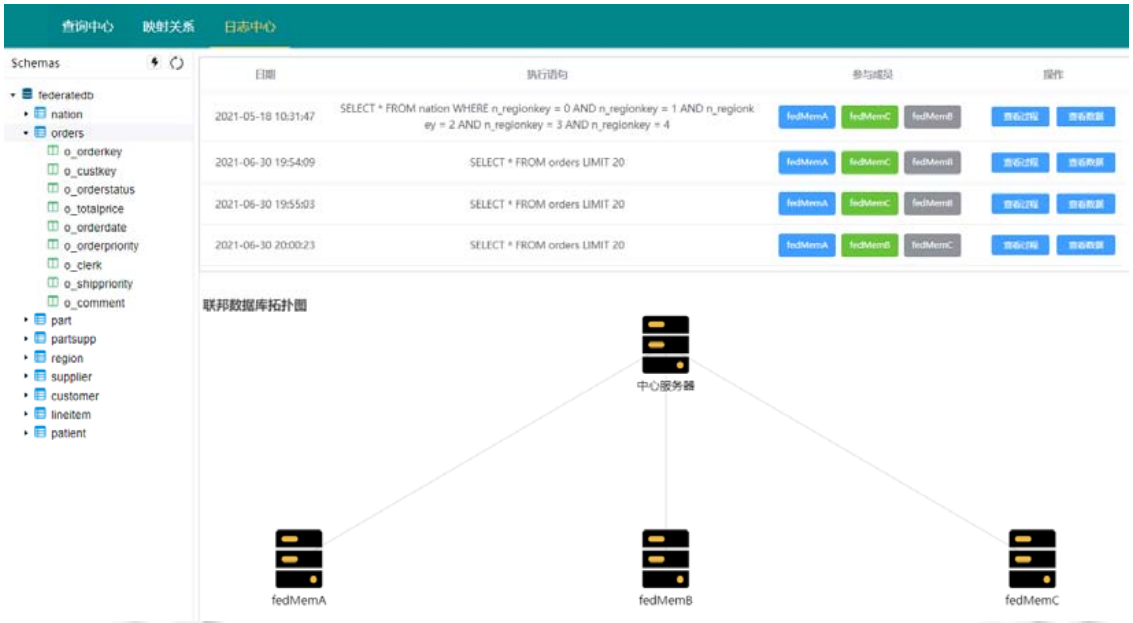


图 5　本系统交互接口

6　系统性能验证

6.1 实验设置

- 实验环境

本次实验需要搭建联邦场景，实验使用多台机器模拟多数据拥有方参与的情况. 每台机器配有 1 个 CPU (型号: Intel®Xeon®Platinum 8269CY CPU T 3.10 GHz)和 32 GB 内存，操作系统为 Ubuntu 18.04.5 LTS (Bionic Beaver). 具体而言，选用一台机器运行中心服务器进程，其他每台机器运行一个数据拥有方进程，并为每个数据拥有方构建其数据库. 在查询操作执行的过程中，各进程协同完成计算. 由于不同的数据参与方真正位于不同的机器上，因此实现了各方数据物理分散存储. 不同于在单机上通过虚拟机模拟联邦场景的方式，数据的分散存储更容易验证“数据不出本地”，实验设置更为真实.

- 测试数据

实验选用测试数据集为 TPC Benchmark™H (TPC-H)[12]. 该数据集由含有各类商品信息的数据表组成. 我们选用其中 1 GB 的数据构造数据拥有方数据库. TPC-H 数据集被广泛用于数据库系统的性能测试分析中[13]，其实验结果具有参考意义.

- 考核指标

实验中，使用查询语句的执行时间作为系统运行效率的量化考核指标.

- 比较系统

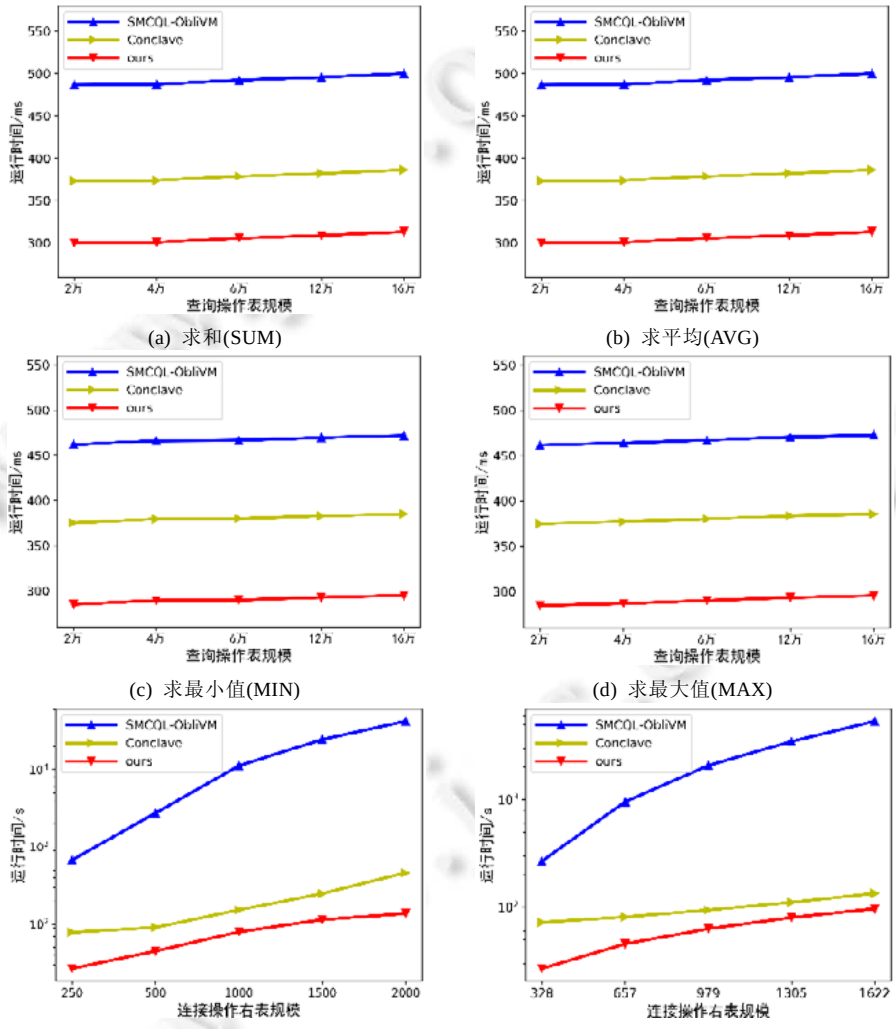
本次实验选取了最具代表性的两个支持多方安全的数据联邦系统 SMCQL[5]和 Conclave[6]. 此外, 为了更进一步验证本系统的高效性和规模可扩展性, 选取安全多方计算工具库 MP-SPDZ[9]作为比较系统.

6.2 系统性能比较

首先分析本系统同代表性多方安全数据联邦系统 SMCQL 和 Conclave 的实际应用性能差异. 具体比较单一查询操作和复合查询语句, 以此全面检验系统的应用性能.

6.2.1 单一查询操作系统性能比较

SMCQL 和 Conclave 系统所支持的最大数据拥有方数量分别为 2 和 3, 经过适当修改, SMCQL 可支持最多在 3 个数据拥有方上的单一查询操作. 因此在参与方个数为 3 的场景下进行对单一查询操作进行比较, 3 个系统在求和(SUM)、求平均(AVG)、求最值(MIN/MAX)、等值连接(equi-join)和任意连接(θ-join)的操作性能比较的比较如图 6 所示, 其中, 连接操作的左表规模为 20 万行, 其中, 表规模指所有数据拥有方参与连接操作的表规模之总和.



(a) 求和(SUM) (b) 求平均(AVG)

(c) 求最小值(MIN) (d) 求最大值(MAX)

(e) 等值连接(equi-join) (f) 任意连接(θ-join)

图 6 本系统、SMCQL、Convlave 单一查询操作比较

从实验比较结果可以看出，随着运行数据规模的增加，不同系统的连接操作耗时明显增加，求和、求平均、求最值操作耗时增幅较小．但我们所实现的系统始终能够保持最优性能，且时间开销最小仅为 SMCQL 和 Conclave 的 0.3%和 29.7%.

6.2.2 复合查询语句的系统性能比较

复合查询语句是单一查询操作的综合，但由于 SMCQL 系统的复杂性和实现问题，不易直接修改使其支持数据拥有方超过 3 的多方场景[13]．因此，本节聚焦于和 Conclave 比较．基于 TPC-H 数据集中的复合查询语句具体如图 7 所示，两系统执行性能比较如图 8 所示．

```
SELECT   SUM(L_extendedprice)/7.0as avg_yearly
FROM     Lineitem, Part
WHERE    P_partkey=L_partkey
         and P_brand='Brand#55'
         and P_container='MED DRUM';
```

图 7 基于 TPC-H 数据集的 SQL 查询

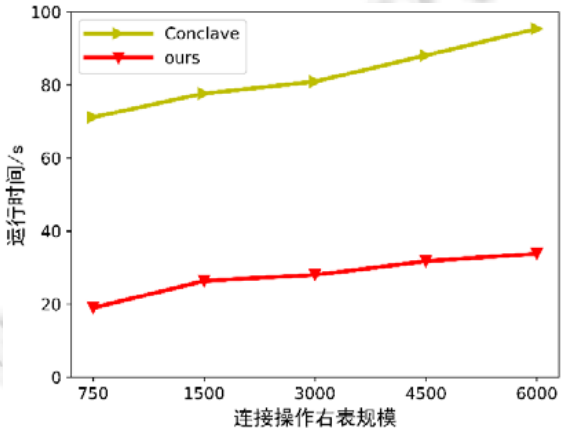


图 8 本系统、MP-SPDZ 的复合语句查询比较(左表规模为 20 万行)

实验结果表明，本系统的效率能够显著优于 Conclave，效率优势最高可达 3.75 倍．

综合上述实验结果，本系统在目前能够支持多方安全的数据联邦系统中达到了最优性能．更进一步地，本系统的另一大优势在于能够支持超过 3 方的数据查询操作，下面的实验将就此展开验证．

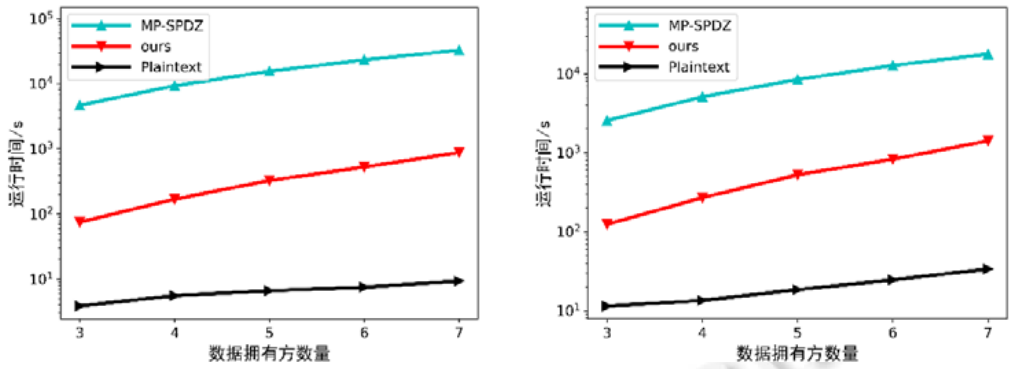
6.3 多方性能分析

在数据拥有方多于 3 方时，SMCQL 和 Conclave 两系统均已无法应用．为此，选择了安全多方计算工具库 MP-SPDZ[9]．作为比较系统，MP-SPDZ 实现了支持多方的安全代数运算，可以作为本系统基础算子的比较对象．实验进一步验证了本系统的查询操作在多方情况下的性能表现．由于求和与求平均操作只是基础算子的简单叠加应用，因此重点验证了等值连接和任意连接的操作性能．

6.3.1 连接操作的多方性能分析

本节实验验证本系统的连接操作在多方场景下的性能表现．本系统的等值连接与任意连接操作时间开销随数据拥有方数量增加的时间开销如图 9 所示．

可以看到，在不同数据拥有方数量的情况下，本系统始终保持较为快速的性能表现，并且能够在和直接明文上计算开销相差不大的情况下有效保护数据安全．

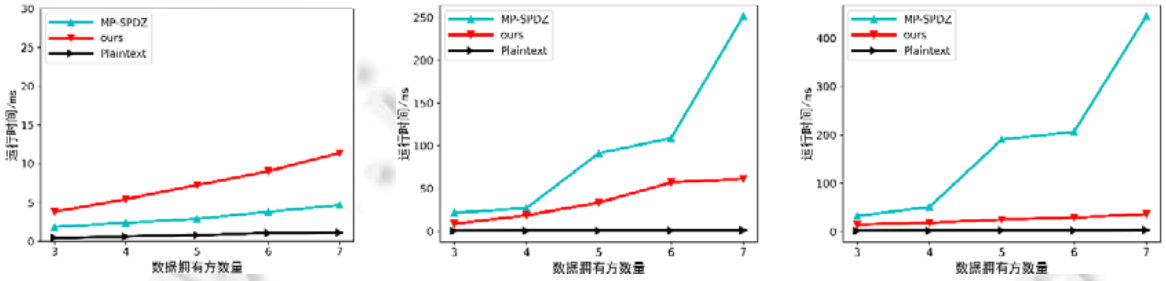


(a) 等值连接(equi-join) (b) 任意连接(θ-join)

图 9 本系统、MP-SPDZ、明文计算的两类连接查询比较

6.3.2 基础算子的多方性能分析

本系统的基础算子加法、乘法和比较运算与 MP-SPDZ、明文上计算的性能比较如图 10 所示. 由于减法和加法之间、除法和乘法之间较易转换, 故省略其比较.



(a) 加法算子 (b) 乘法算子 (c) 比较算子

图 10 MP-SPDZ、明文、本系统多方安全算子库的算子效率比较

可以看到, 3 类基础算子的性能比较中, 随着数据参与方增加, 时间开销随之增加. 对于乘法和比较算子, 本系统的算子执行效率始终快于 MP-SPDZ, 更为接近在明文上计算的性能. 对于加法算子, 虽然略慢于 MP-SPDZ, 但是效率相差较小. 因此, 本系统可以在较少的时间开销代价下实现数据安全保护的要求.

综合上述实验, 本系统在数据拥有方数量增加的情况下依然保持了更好的性能. 这说明本系统不仅能够在执行效率上优于现有的 SMCQL 和 Conclave 系统, 也能在规模可拓展性上显著超越现有系统.

7 相关工作

本文所实现的系统是在联邦场景下的多方安全数据联邦系统. 不同于 20 世纪 80 年代产生的联邦数据库概念, 这类系统聚焦数据库异构问题, 旨在实现异构数据库系统间的协作. 近年来, 随着对用户数据安全保护的加强, 联邦场景下多方安全的数据库系统概念应运而生, 又称为数据联邦系统. 这类系统不仅能够支持异构数据库的协作, 而且注重保护多方协作中数据的安全性问题. 这类系统实现数据安全保护主要依赖安全多方计算技术. 下面分别介绍联邦数据库技术、多方安全数据联邦技术和安全多方计算技术的相关工作.

7.1 联邦数据库技术

联邦数据库系统[14]是在 20 世纪 80 年代提出来的概念, 这类数据库系统并非实体存在的数据库, 它是介于底层多个数据库与上层用户之间的中间件. 它能够屏蔽底层多个数据库间的异构性, 对用户表现为一个统

一的数据库, 是运行在各数据库上的虚拟数据库.

联邦数据库系统底层的多个异构数据库既互相独立又共同协作. 不同于传统的分布式数据库, 联邦数据库需要各个数据库共同协作完成数据的存储查询修改等功能. 联邦数据库系统底层中的各数据库具有一定的自治性, 其拥有方数据库本地独立操作、独立运行. 联邦数据库系统的应用背景源自公司收购后的数据管理. 例如, 当公司 *A* 收购公司 *B* 后, 需要把公司 *B* 的数据纳入管理. 这一过程可能会因为两公司数据库不兼容受到阻碍. 联邦数据库技术能够解决此问题. 通过构建联邦数据库管理系统并将 *B* 公司数据库系统接入, 公司 *A* 无须将公司 *B* 的所有数据迁移至公司 *A* 的数据库系统中, 只需通过联邦数据库系统与公司 *B* 联合处理数据. IBM 公司推出的 DB2 数据管理系统[15]中即支持该联邦数据库.

这类联邦数据库技术中的联邦概念蕴含了多方协同、各方异构自治的特点, 然而并不涉及多方数据中的数据隐私安全问题, 因此和本文探讨的联邦场景及多方安全数据库概念还存在较大差异. 本文聚焦的联邦场景及其有关工作详见第 7.2 节.

7.2 面向多方安全的数据联邦技术

随着国际和国内对用户数据隐私保护要求的日益加强, 传统联邦数据库不考虑数据安全直接开展数据共享的方法已不适用. 为应对数据安全挑战, 面向多方安全的数据联邦技术兴起. 这类系统可以实现多数据拥有方合作完成查询操作的同时, 保证参与计算的敏感数据不泄漏, 从而保护数据安全.

Bater 等人于 2017 年提出了 SMCQL 数据联邦系统[5], 该系统能够支持在两个数据拥有方的数据库上面进行安全地查询. SMCQL 系统把数据表每一列的访问权限划分为公共(public)和私有(private), 不同权限对应不同共享安全需求. 例如, 公共列的值默认可以共享, 因此如果查询中的运算仅涉及公共列数据, 则可以直接明文计算, 和传统数据库计算方式相同; 而如果查询中的运算过程涉及私有列, 则需要在保证原始私有数据不离开本地的前提下展开查询操作, 从而实现数据不泄漏的安全性要求. 该系统虽然能够保证私有数据安全, 但因保障安全带来了巨大的运行时间开销, 且仅能支持两个数据拥有方, 效率和参与方规模的扩展性问题限制了该系统的应用.

Volgushev 等人在 2019 年提出了 Conclave 数据联邦系统[6], 该系统为了控制保障安全带来的时间开销, 提出了上推(push up)、下推(push down)和混合协议(hybrid protocol)等查询优化. 这些优化的核心思想都是尽可能把查询操作的计算在数据拥有方内部完成, 从而减少数据拥有方之间的数据交互, 进一步降低安全保护带来的时间开销. 该系统在效率上较之 SMCQL 有所提升, 但系统最多支持 3 个数据拥有方, 应用场景有限.

目前, 数据联邦系统实现安全操作的思路基本一致, 即把 SQL 语句转化为安全操作原语再执行. 这些安全操作原语使用了安全多方计算技术, 这类技术能够实现多方相互不泄漏敏感数据的情况下进行联合计算, 契合了数据联邦的安全需求. 下面将具体介绍安全多方计算技术的有关工作.

7.3 安全多方计算技术

安全多方计算(secure multi-party computation, SMC)是现代密码学发展中为解决安全计算问题而提出的一类密码协议集合, 它解决了在一些互不信任的参与方之间联合计算一个函数的问题. 该技术最早是由姚期智院士于 1982 年提出的百万富翁问题引出的[11]. 在安全多方计算所关注的问题可形式化表述为: n 个计算参与方分别持有数据 $x_1,x_2,\ldots,x_n$, 协议目的是利用各方秘密数据计算一个预先达成共识的函数 $y_1,y_2,\ldots,y_n=f(x_1,x_2,\ldots,x_n)$. 此时, 任意一方可以得到对应的结果 y_i, 但无法获得其他任何信息. 为解决这一问题, 两大常见的技术有混淆电路(garbled circuits, GC)[16]与秘密共享(secret sharing, SS)[17]两种技术. 两类技术的具体介绍如下.

7.3.1 混淆电路

混淆电路技术是一种将计算函数转换成布尔电路, 对真值表进行加密打乱进行计算的一种技术, 是解决两方之间的安全计算问题的通用框架. 该技术可应用于可验证计算[18]、KDM(key-dependent message)安全性[19]等, 也可在生活中广泛应用于保护隐私的医疗诊断[20]、拍卖机制[21]与信息检索[22]等. 该模型最早是由图灵奖获得者姚期智院士在 1986 年提出的半诚实模型下的姚氏电路[23], 用来解决百万富翁问题. 姚氏电路的主

要工作是将任意功能的函数转化为布尔电路, 电路有两个参与方 Alice 和 Bob, 其中, Alice 根据逻辑电路产生真值表, 再以线缆对应的字符串为密钥, 针对每一个电路门进行两重对称加解密运算, 生成混淆密文表; Bob 调用不经意传输(oblivious transfer, OT)协议[24]获取其输入的字符串形式, 并以此字符串及 Alice 发送的字符串为密钥, 逐行试解混淆密文表. 近年来, 由 Liu 等人和 Zahur 等人分别提出了基于混淆电路的工具库 ObliVM[3]和 Obliv-C[4]. 该工具库构建一套面向开发者的高级程序语言, 能够直接将开发者撰写的模板代码编译为混淆电路, 从而大大降低开发难度. 事实上, 前面所述的 SMCQL 系统就使用了 ObliVM 构建混淆电路以保证安全计算, 而 Conclave 则部分使用了 Obliv-C. 但是这两个工具库采用混淆电路作为安全后端, 在计算效率上表现上略差, 例如, ObliVM 工具库只能负载不足 100 KB 的数据输入量[13].

7.3.2 秘密共享

秘密共享是安全多方计算中的另一种重要方法, 它的思想是: 将秘密拆解分为多个部分(子秘密), 然后将每部分发送给对应的参与者, 使得最后只有授权的参与者集合的子集协同重构出原始的秘密, 而其他任意的非授权的参与者子集无法重构原始的秘密. 秘密共享除了运用在安全多方计算之外, 其最初是用于安全信息存储的问题[25], 还可以用于密钥分发[26]、访问控制[27]等, 在生活中也可应用于电子投票[28]、版权确权[29]和机器学习隐私保护[30]等. 秘密共享最早是由 Shamir[10]和 Blakley[31]引入的. 他们提出的是一种阈值的秘密共享, 即, 只要满足参与者集合的子集的大小大于某个阈值的任何集合都能构造出原始的秘密. Shamir 模式的秘密共享计算流程主要利用拉格朗日插值原理: 首先, 随机选择一个多项式, 将分发者的输入作为多项式的常量项; 然后, 用此多项式来计算分发给各个参与者的子秘密, 通过拉格朗日插值法可以知道满足参与者集合的大小大于指定的阈值时, 就可以重新构造出原始的多项式, 从而得到原始的秘密. 同样也有基于秘密共享方法的多方安全计算工具库, 例如 Sharemind[8]和 MP-SPDZ[9]. 前述数据联邦系统 Conclave 就使用了 Sharemind, 但该工具库最多支持 3 个计算方参与并且未开源, 因此在使用上存在限制.

8 总结与展望

本文针对日益严格的数据隐私安全保护前提下和数据共享需求, 设计并实现了联邦场景下面向多方安全的数据联邦系统. 系统自底向上实现了系统适配、多方安全算子库、查询引擎、交互接口, 能够屏蔽底层数据库异构特性, 向用户提供统一且友好的操作界面, 并实现数据隐私安全保护前提下的数据共享. 具体来说, 本文工作总结如下.

- 实现了系统多方安全算子库与查询引擎. 基于秘密共享框架实现了包括加减乘除比较在内的高效的安全基础算子. 在此基础上构建查询引擎, 支持包括求和、求平均、求最值、等值连接和任意连接的数据库基本查询操作. 查询引擎的设计与实现上充分考虑多方协作特性, 在查询语句的解析、查询计划生成和查询操作执行的全流程中贯彻本地操作优先、减少传输冗余的思想, 最大限度压缩了数据拥有方之间的数据交互, 从而极大地降低了安全开销, 提高系统性能.
- 实现了系统适配和交互界面. 系统适配接口面向多数据拥有方的数据库, 屏蔽其异构性, 而交互界面则面向用户提供统一的 SQL 语句查询接口和图形化交互界面, 方便数据共享过程的统一协调与管理, 提高易用性.
- 实验验证了系统性能. 在标准数据集 TPC-H 上的实验结果表明, 与目前的数据联邦系统 SMCQL 和 Conclave 相比, 本系统能够支持超过 3 个数据拥有方以上的数据联邦场景, 并且在各类基本查询操作和复合查询上都表现出更高的效率. 本系统的高效性和规模可扩展性表明, 其具有比当前主流数据联邦系统更高的实用性.

本系统的未来工作如下.

- 扩展系统支持查询操作范围. 目前, 系统基于秘密共享这一基础多方安全计算思想, 实现了如聚合、连接等操作符, 但与成熟的数据库系统相比, 本系统支持的功能还较为有限. 可在目前系统已有基础上, 进一步拓展窗口等操作.

- 基于现有基础算子实现更为复杂的数据挖掘分析算法. 本系统目前已支持简单的查询操作, 可基于已实现求和、求积等算子构造更为复杂的数据挖掘分析算法, 例如聚类算法、回归算法等, 进一步在保护隐私前提下, 挖掘多方数据所蕴含的价值.

References:

[1] Doan AH, Halevy A, Ives Z. Principles of Data Integration. Elsevier, 2012.

[2] Shi DY, Wang YS, Zheng PF, Tong YX. Cross-Silo federated learning-to-rank. Ruan Jian Xue Bao/Journal of Software, 2021, 32(3): 669−688 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6174.htm [doi: 10.13328/j.cnki.jos.006174]

[3] Liu C, Wang XS, Nayak K, *et al*. Oblivm: A programming framework for secure computation. In: Proc. of the 2015 IEEE Symp. on Security and Privacy. IEEE, 2015. 359−376.

[4] Zahur S, Evans D. Obliv-C: A language for extensible data-oblivious computation. IACR Cryptology ePrint Archive, 2015, 2015: No.1153.

[5] Bater J, Elliott G, Eggen C, *et al*. SMCQL: Secure query processing for private data networks. Proc. of the 2017 VLDB Endowment, 2017, 10(6): 673−684.

[6] Volgushev N, Schwarzkopf M, Getchell B, *et al*. Conclave: Secure multi-party computation on big data. In: Proc. of the 14th EuroSys Conf. ACM, 2019. No.3.

[7] Hastings M, Hemenway B, Noble D, *et al*. Sok: General purpose compilers for secure multi-party computation. In: Proc. of the 2019 IEEE Symp. on Security and Privacy. IEEE, 2019. 1220−1237.

[8] Bogdanov D, Laur S, Willemson J. Sharemind: A framework for fast privacy-preserving computations. In: Proc. of the 2008 European Symp. on Research in Computer Security. Berlin, Heidelberg: Springer, 2008. 192−206.

[9] Keller M. MP-SPDZ: A versatile framework for multi-party computation. In: Proc. of the 2020 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2020. 1575−1590.

[10] Shamir A. How to share a secret. Communications of the ACM, 1979, 22(11): 612−613.

[11] Yao AC. Protocols for secure computations. In: Proc. of the 23rd Annual Symp. on Foundations of Computer Science. IEEE, 1982. 160−164.

[12] 1988. http://www.tpc.org/tpch/

[13] Wang Y, Yi K. Secure Yannakakis: Join-aggregate queries over private data. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 1969−1981.

[14] Sheth AP, Larson JA. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 1990, 22(3): 183−236.

[15] Josifovski V, Schwarz P, Haas L, *et al*. Garlic: A new flavor of federated query processing for DB2. In: Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data. ACM, 2002. 524−532.

[16] Bellare M, Hoang VT, Rogaway P. Foundations of garbled circuits. In: Proc. of the 2012 ACM Conf. on Computer and Communications Security. ACM, 2012. 784−796.

[17] Beimel A. Secret-sharing schemes: A survey. In: Proc. of the2011 Int'l Conf. on Coding and Cryptology. Berlin, Heidelberg: Springer, 2011. 11−46.

[18] Setty S, Vu V, Panpalia N, *et al*. Taking proof-based verified computation a few steps closer to practicality. In: Proc. of the 21st USENIX Security Symp. ACM, 2012. 253−268.

[19] Applebaum B. Key-dependent message security: Generic amplification and completeness. In: Proc. of the 2011 Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Berlin, Heidelberg: Springer, 2011. 527−546.

[20] Chen F, Cheng S, Mohammed N, *et al*. Precise: Privacy-preserving cloud-assisted quality improvement service in healthcare. In: Proc. of the 8th Int'l Conf. on Systems Biology. IEEE, 2014. 176−183.

[21] Kolesnikov V, Sadeghi AR, Schneider T. Improved garbled circuit building blocks and applications to auctions and computing minima. In: Proc. of the 2009 Int'l Conf. on Cryptology and Network Security. Berlin, Heidelberg: Springer, 2009. 1−20.

[22] Kim HJ, Kim HI, Chang JW. A privacy-preserving kNN classification algorithm using Yao's garbled circuit on cloud computing. In: Proc. of the 2017 IEEE 10th Int'l Conf. on Cloud Computing. IEEE, 2017. 766−769.

[23] Yao ACC. How to generate and exchange secrets. In: Proc. of the 27th Annual Symp. on Foundations of Computer Science. IEEE, 1986. 162−167.

[24] Kilian J. Founding crytpography on oblivious transfer. In: Proc. of the 20th Annual ACM Symp. on Theory of Computing. ACM, 1988. 20−31.

[25] Huang W, Langberg M, Kliewer J, *et al*. Communication efficient secret sharing. IEEE Trans. on Information Theory, 2016, 62(12): 7195−7206.

[26] D'Souza R, Jao D, Mironov I, *et al*. Publicly verifiable secret sharing for cloud-based key management. In: Proc. of the 2011 Int'l Conf. on Cryptology in India. Berlin, Heidelberg: Springer, 2011. 290−309.

[27] Naor M, Wool A. Access control and signatures via quorum secret sharing. IEEE Trans. on Parallel and Distributed Systems, 1998, 9(9): 909−922.

[28] Schoenmakers B. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Proc. of the '99 Annual Int'l Cryptology Conf. Berlin, Heidelberg: Springer, 1999. 148−164.

[29] Zhu Y, Yang YT, Sun ZW, Feng DG. Ownership proofs of digital works based on secure multiparty computation. Ruan Jian Xue Bao/Journal of Software, 2006, 17(1): 157−166 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/157.htm [doi: 10.1360/jos170157]

[30] Tan ZW, Zhang LF. Survey on privacy preserving techniques for machine learning. Ruan Jian Xue Bao/Journal of Software, 2020, 31(7): 2127−2156 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6052.htm [doi: 10.13328/j.cnki.jos.006052]

[31] Blakley GR. Safeguarding cryptographic keys. In: Proc. of the Int'l Workshop on Managing Requirements Knowledge. IEEE Computer Society, 1979. 313−313.

附中文参考文献:

[2] 史鼎元, 王晏晟, 郑鹏飞, 童咏昕. 面向企业数据孤岛的联邦排序学习. 软件学报, 2021, 32(3): 669−688. http://www.jos.org.cn/1000-9825/6174.htm [doi: 10.13328/j.cnki.jos.006174]

[29] 朱岩, 杨永田, 孙中伟, 冯登国. 基于安全多方计算的数字作品所有权证明. 软件学报, 2006, 17(1): 157−166. http://www.jos.org.cn/1000-9825/17/157.htm [doi: 10.1360/jos170157]

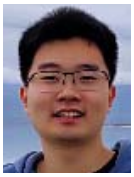
[30] 谭作文, 张连福. 机器学习隐私保护研究综述. 软件学报, 2020, 31(7): 2127−2156. http://www.jos.org.cn/1000-9825/6052.htm [doi: 10.13328/j.cnki.jos.006052]

李书缘(1998－), 女, 博士生, CCF 学生会员, 主要研究领域为大数据分析处理, 隐私保护.

张利鹏(1997－), 男, 硕士生, CCF 学生会员, 主要研究领域为联邦数据库, 隐私保护.

季与点(1991－), 男, 博士, 工程师, 主要研究领域为时空数据分析处理, 数据压缩, 数据挖掘.

童咏昕(1982－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为联邦学习, 时空大数据分析处理, 智慧城市, 众包计算, 群体智能, 隐私保护.



史鼎元(1998－), 男, 硕士生, 主要研究领域为联邦学习, 时空大数据分析处理, 众包计算, 群体智能, 隐私保护.

许可(1971－), 男, 博士, 教授, 博士生导师, 主要研究领域为算法与人工智能.

廖旺冬(1996－), 男, 硕士生, 主要研究领域为大数据分析处理, 隐私保护.