软件学报 ISSN xxx
Journal of Software,[doi: 10.13328/j.cnki.jos.007283]

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

基于无服务器计算的多方数据库安全计算系统 *

马旭阳[1], 周小凯[1], 郑浩宇[1], 崔斌[2], 徐泉清[3], 杨传辉[3], 晏潇[4], 江佳伟[1]

[1](武汉大学 计算机学院, 湖北 武汉 430072)

[2](北京大学 计算机学院, 北京 100971)

[3](蚂蚁集团, 浙江 杭州 310013)

[4](香港中文大学 博智感知交互研究中心,香港 999077)

通讯作者: 江佳伟,E-mail: jiawei.jiang@whu.edu.cn

摘 要: 联合多方数据库的安全计算可以在保护数据隐私的情况下,对多个数据库的私有数据进行联合查询或联合建模.这样的联合体通常是一个松散的组织,各参与的数据库可以随时离线,然而现有多方安全计算系统通常采用秘密共享等隐私计算方案,需要参与者保持在线状态,导致系统的可用性差.此外,现有系统对外提供服务时无法预知用户的数量以及请求速度,如果将系统部署在私有集群或者租用云计算平台的虚拟机,面对爆发式请求时系统延迟增大,在请求较少时又造成资源浪费,系统整体的可扩展性差.随着云计算技术的发展,无服务器计算(也称 Serverless Computing)作为一种新的云原生部署范式出现,具有良好的弹性资源伸缩能力.在本工作中,我们提出了基于无服务器计算环境的系统架构和间接通信方案,实现了一套高可扩展、高可用的多方数据库安全计算系统,可以容忍数据库节点掉线,并且在用户请求流量发生变化时自动伸缩系统资源.我们基于阿里云和 OceanBase 数据库实现了系统原型并进行了充分的实验对比,结果显示本系统在低频查询、横向建模等任务上在计算成本、系统性能和可扩展性方面优于现有系统,最高能够节省 78%的计算成本、提升系统性能 1.6 倍,同时也分析了本系统对于复杂查询、纵向建模等任务存在的不足.

关键词: 云原生系统;无服务器计算;多方数据库安全计算

中图法分类号: TP311

中文引用格式: 马旭阳,周小凯,郑浩宇,崔斌,徐泉清,杨传辉,晏潇,江佳伟.基于无服务器计算的多方数据库安全计算系统.软件学报. http://www.jos.org.cn/1000-9825/7283.htm

英文引用格式: Ma XY,Zhou XK,Zheng HY,Cui B,Xu QQ,Yang CH,Yan X,Jiang JW.A Secure Multi-Party Database Computing System Based On Serverless Computing.Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7283.htm

A Secure Multi-Party Database Computing System Based on Serverless Computing

MA Xü-Yang[1], ZHOU Xiao-Kai[1], ZHENG Hao-Yu[1], CUI Bin[2], XU Quan-Qing[3], YANG Chuan-Hui[3], YAN Xiao[4], JIANG Jia-Wei[1]

[1](School of Computer Science,Wuhan University,Wuhan 430072,China)

[2](School of Computer Science,Peking University,Beijing 100871,China)

[3](Ant Group,Hangzhou 310013,China)

[4](Centre for Perceptual and Interactive Intelligence,The Chinese University of Hong Kong,Hong Kong 999077,China)

Abstract: Secure computation for federated multi-party databases enables federated querying or federated modeling tasks on private

* 基金项目: 国家重点研发计划(2023YFB2703604); 湖北省重点研发计划 (2023BAB077, 2023BAB170); 国家自然科学基金(62472327); 中央高校基本科研业务费专项 (2042023kf0219); CCF-蚂蚁科研基金 (CCF-AFSG RF20230106)

马旭阳和周小凯为共同第一作者

收稿时间: 2024-05-27; 修改时间: 2024-07-16, 2024-08-19; 采用时间: 2024-08-29; jos 在线出版时间: 2024-09-13

data from multiple databases while preserving data privacy. Such a federation is typically a loosely organized group where the participating databases can dropout at will. However, existing multi-party secure computation systems usually employ privacy-preserving computation schemes such as secret sharing, which require the participants to remain online, resulting in poor system availability. Moreover, the existing system can not predict the number of users and the request speed when providing services to the outside. If these systems are deployed on a private cluster or rented virtual machines from a cloud computing platform, it will experience increased latency during sudden bursts of requests and resource wastage when the request workload is low, leading to poor scalability. With the advancement of cloud computing technology, serverless computing has emerged as a new cloud-native deployment paradigm that offers elastic resource scaling. In this work, we design a system architecture and an indirect communication scheme within the serverless computing framework to architect a highly scalable and highly available multi-party database secure computation system. This system can tolerate database node dropouts and automatically scale system resources in response to dynamic request workload. We implement a prototype of the system based on Alibaba Cloud and OceanBase database, conducting comprehensive experiments evaluation. The results show that our system outperforms existing systems in terms of computational cost, system performance, and scalability for tasks such as low-frequency queries and horizontal modeling. It can save up to 78% in computational costs and improve system performance by over 1.6 times. We also analyze the shortcomings of our system for complex queries and vertical modeling tasks.

Key words: cloud-native system; serverless computing; secure computation of multi-party databases

随着人工智能和大数据技术的飞速发展,数据的价值越来越受到重视.不同数据拥有者的数据覆盖不同的用户群体,记录不同类型的用户行为,因而催生了数据多方共享计算的需求[1,2].但是,由于用户隐私安全意识的增强以及相关法律法规对数据保护力度的加大,用户数据库之间的直接数据共享变得非常困难.为了在保护数据隐私的情况下进行多方安全计算,联邦学习[3]和联邦数据库[4]被提出.它们通过使用混淆电路、秘密共享、差分隐私、同态加密等方案进行数据的安全共享,设计并实现了多方数据库安全计算系统.

多方数据库安全计算面临多样的数据分布,按照特征空间和样本空间的分布情况主要分为横向数据结构和纵向数据结构[2-5].当多方数据库拥有的数据具有相同的特征空间和不同的样本空间时,可视为将完整的数据矩阵按行切分到多个节点,称为横向数据结构;当多方数据库拥有的数据具有相同的样本空间和不同的特征空间时,可视为将完整的数据矩阵按列切分到多个节点,称为纵向数据结构.

在数据库系统中,多方安全计算包括查询和建模两种任务.对于查询任务,已有一些学者提出使用秘密共享的隐私计算等方案,通过共同计算一组约定的函数来共享中间数据,在保证数据对外不可见的情况下在多方数据库间实现多方安全计算[6-13].对于建模任务,学者们提出共享模型参数或梯度的方法进行模型同步和训练,保证了在数据不离开本地的约束下完成多方安全计算,在一定程度上保护了数据隐私安全[14-25].

现有多方数据库安全计算系统存在一定的局限.第一,系统的可扩展性较差.当前的技术方案将系统部署在私有集群或云虚拟机集群,集群的资源难以在系统运行时进行弹性伸缩,因而难以应对动态变化的业务流量.若配置较少的资源,爆发式请求会导致系统因资源不足而响应变慢;若配置较多的资源为请求峰值提供余量,又会造成资源浪费和高昂费用.此外,Conclave[8]等现有系统受限于隐私计算等加密方案的限制,通常只能支持较少的计算节点.第二,系统的可用性较差.现有系统通常需要在计算节点以及服务节点之间建立持续的网络连接,当节点掉线时系统运行会被阻塞.然而,多方安全计算系统的参与者可能来自不同地域、不同行业,不能保证长期在线,会造成系统的频繁阻塞.第三,支持的任务类型有限.联邦学习系统支持多方安全计算的建模任务,联邦数据库系统支持多方安全计算的查询任务,也有一些工作分别针对横向数据结构和纵向数据结构展开研究.随着数据科学、人工智能技术的飞速发展,不同类型计算任务的需求越来越普遍,多方数据库之间的数据结构越来越多样,目前缺乏能够同时支持查询和建模任务,并适应多种数据结构的多方数据库安全计算系统.

为了解决上述问题,本文探索高可扩展和高可用的多方数据库安全计算系统架构.首先需要考虑的是系统部署的基础设施,当前已有多个云平台提供了无服务器计算支持[26-34].无服务器计算是一种函数即服务的云计算新范式,采用按使用付费计费的计价模型,以事件驱动的方式运行.无服务器函数被触发时自动生成函数计算实例并被分配计算资源,运行完成后自动销毁.基于无服务器计算的系统是一种新颖的云原生系统实践[35],近年来已有实时流计算、数据库、机器学习、分布式系统等领域的科研人员进行了相关的探索[36-40].但是,尚未有工作研究如何基于无服务器计算实现多方数据库安全计算系统.具体而言,采用无服务器计算的方式运行多方

数据库安全计算,在系统设计和实现时需要解决以下问题:

- 高可扩展和高可用的系统架构.用户数据存储在多个数据库节点,查询和建模计算任务都需要在数据库节点之间频繁调度任务.在存算分离的架构下,以无服务器实例作为系统的计算单元,需要对系统的计算任务进行合理的解耦,以允许不同计算任务独立地并发执行,使系统具有良好的可扩展性.已有的系统架构在计算节点之间保持双向的长连接,构成的通信拓扑包含所有的计算节点,当节点出现掉线时系统无法正常运行,系统的可用性存在不足,因此不适合松散的多方数据库协作场景.
- 无状态的通信和协作机制.在设计了系统整体架构后,系统内的数据通信是需要实现的关键技术.由于无服务器的计算实例是无状态的,无服务器计算实例之间、以及计算实例与数据库节点之间无法进行直接通信,需要设计合理的通信机制实现数据交互.同时,无状态的无服务器计算实例在完成当前生命周期后即被收回资源,无法在计算实例中持久化计算的中间状态和数据,因此需要设计合理的数据管理和协作机制来保证系统按照正确的逻辑运行,从而得到可靠的任务结果.

针对上述挑战,我们基于无服务器计算,设计了多方数据库安全计算的系统架构和适配的通信方案,基于阿里云和 OceanBase[41,42]数据库实现了一种弹性伸缩的多方数据库安全计算系统 MpSDB(Multi-party Serverless Database).(代码已发布在 https://github.com/ZhouXiaokay/SecureDatabase)具体而言,本文的主要贡献如下:

- 设计并实现了基于无服务器计算的多方数据库安全计算架构.在无服务计算环境中通过启动无服务器函数来调度计算任务、协同数据库节点,服务节点采用请求驱动的执行模式,即数据库节点主动调用服务节点的函数接口发送数据,从而支持松散的多方数据库协作场景,实现系统架构的高可用.将系统的计算任务解耦为多个函数模块并部署在无服务器计算实例中,可以根据任务请求流量为不同函数模块动态扩展计算资源,实现系统架构的高可扩展.设计了一种事件驱动的模式对数据库节点的本地计算结果进行聚合运算提高了系统对掉线者的容忍度.本系统架构能够接入不同类型的数据源,包括在无服务器计算、云服务器或私有服务器等环境部署的数据库节点.
- 设计并实现了无服务器环境下的多方数据库安全通信与协作机制.设计了间接通信机制,允许无状态的无服务器实例之间,以及无服务器实例与数据库节点之间通过读写共享云数据管道实现数据交互.为实现系统内不同无服务器实例与数据库节点之间的正确协作,设计了无服务器计算实例共享文件和通信交互的严密规则.为了实现身份认证和保护数据隐私,对共享云数据管道进行了空间划分以实现细粒度的权限管理机制,并使用同态加密技术保证数据安全.
- 实现了支持多种类型任务的系统原型并进行了详尽的实验分析.我们基于阿里云和 OceanBase 数据库实现了一套基于无服务器计算的多方数据库安全计算系统并部署在生产环境中,同时支持查询任务和建模任务,支持横向数据结构和纵向数据结构,满足用户多样的计算需求.在实验中,将本系统 MpSDB 与已有的有服务器系统从成本、性能、扩展性等多角度进行对比分析.实验结果显示,得益于快速启动和按使用付费的计价模型,MpSDB 在低频次的查询任务和横向数据结构的建模任务上表现出更好的时间和成本优势;在高频次的查询任务和纵向数据结构的建模任务上,无服务器实例的频繁启动和通信延迟导致系统性能下降,并产生了更高的系统成本.在系统扩展性方面,已有系统难以支持较多的数据库节点,而 MpSDB 可支持 3 个以上数据库节点. MpSDB 具有良好的系统扩展能力,当任务请求流量增加时,任务平均响应时间无明显增加,而有服务器系统的任务平均时间出现明显增加.

本文第 1 节介绍本工作的相关领域工作.第 2 节介绍本系统的整体架构、系统关键技术和相关分析.第 3 节按任务类型分别介绍查询任务和建模任务的运行流程和系统实现的细节.第 4 节展示系统实验结果并进行相关分析.最后在第 5 节对本文进行总结与展望,并讨论本系统存在的局限性.

1 相关工作

在如今的大数据时代,海量的数据存在着巨大的价值,可以用来进行数据分析或训练模型进行预测或分类.一个用户的个人信息数据可能会分布在不同行业、不同机构.不同数据拥有者(例如机构、公司、应用等)

的数据覆盖不同的用户群体,记录不同类型的用户行为,因而催生了数据多方安全计算的需求[1,2].多方数据库安全计算技术可以在不同数据拥有者的数据库之间共享各自的数据,以提高分析效果和决策水平,具有重要的社会意义和经济价值.例如,相同的用户可能在多个银行拥有账户,如果能在银行之间共享这些数据,能够更好地进行信用评估和风险控制,降低金融风险;医疗机构拥有病人的诊疗记录,政府机构知晓市民的社保医保情况,如果能建立数据共享平台,就可以快速精准地定位经济困难的患者,加强对弱势群体的精准帮扶,提高政府施政水平和市民幸福感.但是,出于隐私安全和数据价值的考虑,不同行业、不同机构不愿意或不能分享私有数据,“数据孤岛”的问题越来越严重[3].为了打通“数据孤岛”,在不同数据拥有者之间实现数据的安全多方共享,并在共享数据上进行计算,联邦数据库和联邦学习等系统被提出.

1.1 联邦数据库系统

联邦数据库[4]在20世纪80年代被提出,着重研究如何在不同的异构数据库系统之间协作查询,但没有保护数据隐私.近些年,随着数据隐私越来越受到重视,在联邦数据库上进行多方安全计算受到研究者的关注.

Bogdanov D等人在2008年提出了Sharemind[6],基于秘密共享的技术实现了在数据拥有方之间进行安全的多方计算,保证在“诚实但好奇”(honest-but-curious)的攻击模式下的数据安全,缺点是只能支持三个数据拥有方.Bater J等人在2017年提出了SMCQL[7],实现了在两个数据拥有方的数据库之间进行查询操作,同时不会泄露隐私的数据给任何第三方.然而,SMCQL 数据保护的额外开销较大,并且只能支持两方的安全计算,难以扩展到比较大的数据联邦组织.Volgushev N等人在2019年提出了Conclave[8],该系统能保证数据安全,通过合理地设计任务处理流程,将查询请求的计算任务尽可能在数据拥有方内部完成,以减少数据拥有方之间的数据交互,从而减少数据保护的计算开销,降低数据泄露的风险.但是 Conclave 最多支持三个数据拥有方,适用性仍然受限.Tong Yongxin等人在2021年基于秘密共享的技术实现了OpenHuFu[10]支持空间对象查询,能够适配多种类型的数据库,并支持3个以上参与方,有效提高了系统的适用性.Zhang YY等人在2022年基于秘密共享的技术提出了联邦 θ-连接查询算法优化策略[11],提高了查询效率.但是,这两个系统存在一定的局限性.采用秘密共享的多方数据库安全计算架构不能有效地处理参与者掉线的问题,对于通信量大的任务会产生高昂的计算开销.并且在面对越来越遍普遍的基于多方数据库的建模任务时,这两个系统目前还不能提供支持.

1.2 联邦学习

联邦学习旨在解决终端用户联合训练机器学习模型的问题,其核心目标是保证用户数据仅保留在本地的情况下实现共同建模[3,5].联邦学习可以扩展数据的多样性和丰富性,从而显著提升提高模型的性能.

根据联邦成员持有数据分布的不同,联邦学习可以分为横向联邦学习与纵向联邦学习.在横向联邦学习中,FedAvg[14]是一种广泛应用的聚合算法,服务器随机选取一部分成员,这些成员在本地使用梯度下降算法更新本地模型后,将本地模型发送给服务器进行模型聚合.针对非独立同分布数据,FedProx[15]、FedBN[16]等算法被提出,以实现更好的聚合效果.在纵向联邦学习中,Hardy 等人提出了一种支持三个参与方的纵向联邦逻辑回归算法[17],使用同态加密保护数据隐私.Wu 等人提出了纵向联邦下的树模型训练算法[18],在多个联邦成员之间使用半同态加密和基于秘密共享的安全多方计算来找到树结点的最佳分裂特征和分裂值.一些工作[19,20]提出了基于分裂学习(Split Learning)的纵向联邦学习架构,将整体模型分为本地模型和顶部模型两部分.Romanini D等人提出了一种多方成员的分裂学习架构[21],首先采用隐私集合求交(Private Set Intersection)确定共有实例集,然后执行分裂学习方法,在拥有标签的成员上计算梯度并进行反向传播.

近年来已有一些工作设计并实现了开源的联邦学习系统与架构.He C等人在2020年提出了FedML[22],这是一个灵活、可扩展的联邦学习研究平台,支持跨设备和跨机构的联邦学习.Shi DY等人在2021年提出了一种在信息检索领域进行联邦排序学习的系统架构[23],打通了企业之间的“数据孤岛”.Liu Y等人在2021年提出了一个支持横向数据结构和纵向数据结构的联邦学习框架FATE[24].Chen D在2023年等人引入了OpenFed[25],这是一个端到端的通用联邦学习架构,便于算法在联邦场景下的灵活部署和公平比较.这些系统与架构要求中心服务器和节点之间保持长连接,在训练过程中节点掉线可能导致任务失败.

1.3 无服务器计算

在部署分布式系统时,依赖固定的集群资源可能导致资源弹性不足,同时系统对故障的低容忍度也可能降低系统的整体可用性.幸运的是,随着云计算技术的迅速发展,系统的部署范式有了更多的选择.近年来,无服务器计算(Serverless Computing,也称函数即服务 FaaS)的出现受到了广泛的关注.无服务器计算通过以函数的形式提供服务,并采用事件驱动的方式运行函数代码,为开发者提供了更高的操作灵活性和资源利用效率.当前众多云平台提供了无服务器计算支持[26].AWS(亚马逊云)在 2014 年率先推出了 Lambda[27],其中函数执行时间限制为 900 秒,可分配的固定内存从 128MB 至 10GB 不等.同时,推出了云存储服务,如 AWS S3[28],可用于间接通信.微软在 2016 年发布了 Azure Functions[29],设定函数的最大执行时间为 600 秒,内存分配范围为 128MB 至 1.5GB.谷歌在 2017 年推出了 Google Cloud Functions[30],允许函数运行时间最长达 3600 秒,内存可配置从 128MB 至 16GB.阿里云在 2017 年发布 FC 函数计算服务[31],并在 2021 年将无服务器计算迁移到神龙架构,推出支持 GPU 的函数计算实例,其中函数的执行时间限制扩展至 24 小时,内存分配可从 128MB 扩展至 32GB.除了这些商业平台外,还有诸如 OpenWhisk[32],OpenFaaS[33]和 OpenLambda[34]等开源平台的无服务器计算框架.

无服务器计算凭借其按使用付费的计价模型和良好的弹性伸缩能力吸引了科研人员的兴趣.近年来,实时流计算、数据库、机器学习、分布式系统等领域的研究者已广泛的探索了无服务器计算的应用.Matthew Perron 等人基于 AWS Lambda 和 S3 开发了 Starling[36],允许用户通过设置并发度来平衡数据分析任务的时间和成本.然而,此项工作的数据是集中存储的,并未涉及多方联合场景下数据安全的保护.Dong HW 等人探讨了基于无服务器计算实现可弹性伸缩的云原生数据库带来的优势及挑战[35].Siren[37]提出了一种基于 AWS Lambda 的异步分布式机器学习框架.Cirrus[38]进一步地基于 AWS Lambda 支持完整的端到端机器学习工作流.这些工作验证了无服务器计算在机器学习领域的可行性.FedLess[39]引入了一个在无服务器计算平台的异构结构上执行联邦学习的系统和框架,将无服务器计算拓展到了联邦学习系统领域.LambdaML[40]使用 AWS 分析了无服务器计算机器学习系统的成本-性能权衡,揭示了无服务器计算的优势和不足.然而,这些研究尚未探讨多方安全计算场景下节点掉线等系统可用性问题,且主要支持横向数据结构下的建模任务,难以适应更复杂的数据结构需求.

无服务器计算将资源租赁单元从云服务器缩减到函数,提供了更细粒度的资源配置.在面对爆发式请求时,无服务器函数可以自动扩展,实现高并发任务调用.从成本角度考虑,不必为短暂的高频计算预置超额资源,有效降低了流量波动下的成本.从资源扩展能力看,无服务器计算能有效避免在固定资源分配策略下,高峰期资源不足导致的高延迟甚至任务失败的情况,保证任务的高效、成功执行.这些优点表明,基于无服务器计算的多方数据库安全计算系统可能解决系统可扩展性差的问题.然而,在功能支持和状态管理上,无服务器计算与有服务器计算存在明显差异,这为多方数据库安全计算系统的设计和实现带来了一系列新的挑战.

2 系统架构

MpSDB 是一个基于无服务器计算的多方数据库安全计算系统,能够支持在横向与纵向数据结构下的查询和建模两种任务.本节介绍系统的整体架构和关键技术设计,系统架构如图 1 所示.

2.1 整体架构

本系统由各参与方的数据库节点和运行在无服务器计算环境中的服务节点组成.用户通过任务接口的无服务器函数与系统进行交互,而各参与方的数据库节点则通过共享云数据管道与系统进行交互.

系统用户.用户通过任务接口选择查询或建模两种多方数据库安全计算类型,设置相关的参数后提交任务.系统保证参与方对彼此的数据不可见的情况下完成用户指定的任务.用户通过任务接口获取计算结果.

共享云数据管道.为了解决无服务器计算实例无状态的问题,本系统实现了一个共享云数据管道用于数据交互.我们将共享云数据管道中的存储空间分为用户交互空间和系统任务空间,根据身份标签授予特定空间的读或写权限实现数据的安全隔离.用户交互空间用于存储用户提交的任务请求及任务结果,系统任务空间用于服务节点中各函数计算实例和数据库节点之间的数据交互.

服务节点.服务节点部署在云平台的无服务器计算环境,负责解析任务并调度参与计算任务的数据库节点,将子任务分配给数据库节点并收集子任务结果,通过聚合子任务结果得到完整的任务结果,最后返回给用户.

数据库节点.本系统支持灵活的数据库节点类型,数据库节点可以部署在无服务器计算环境、云平台的租赁服务器或私有服务器上,增强多方安全计算系统的适用性.数据库节点连接本地的数据库,根据服务节点分解的子任务执行本地的数据查询或模型训练,数据库节点从服务节点获取加密的公钥,加密数据后将子任务结果通过共享云数据管道返回给服务节点.这里使用同态加密方法对子任务结果进行加密,同态加密支持密文上的数学运算,数据以密文的形式离开数据库节点并在服务节点上进行聚合计算,从而保护用户数据的隐私安全.

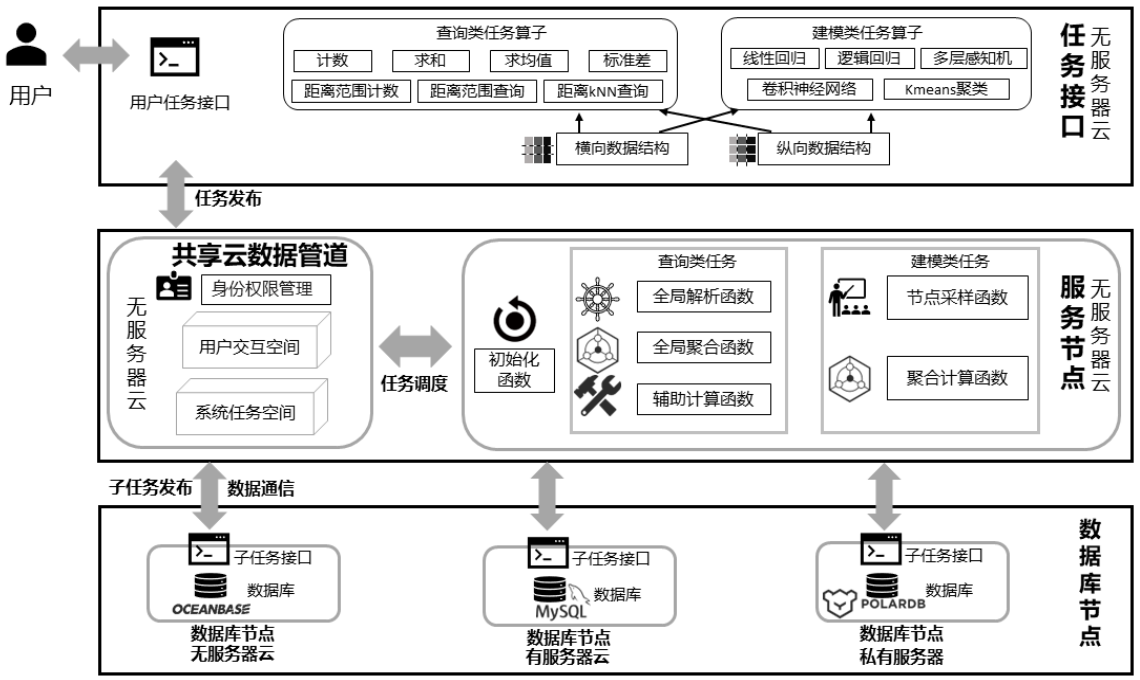

图 1 系统整体架构

2.2 系统关键技术

在本小节中,围绕系统设计中的关键技术进行阐述,包括安全通信和无状态计算.在安全通信中介绍系统内部通信控制、用户数据交互设计和租户级通信空间隔离.在无状态计算设计中介绍节点管理、执行模式和基于任务的计算设计.最后提供数据库节点掉线的容忍度和资源弹性伸缩能力的分析.

2.2.1 安全通信设计

通信安全的挑战.现有系统方案通过长连接实现集群间不同节点的数据通信.但是在无服务器系统中,无状态函数无法建立长连接.现有工作通常采用共享存储的方式进行间接通信,参与通信的双方通过读写约定文件完成数据的传递.但是在多方数据库安全计算的场景下,这种通信方式无法保证参与方之间数据相互不可见.

安全的共享云数据管道.为了加强无服务器系统通信层面的数据保护,我们设计了基于身份验证和权限控制的共享云数据管道.数据库节点在接入系统时被分配 RAM 身份标签,用于完成读写共享云数据管道的验证.为了实现更细粒度的权限控制,我们将共享云数据管道划分为系统任务空间和用户交互空间.

- **严格的系统通信控制.**系统任务空间用于服务节点和数据库节点之间的通信,采用最小化授权的策略.只授予各数据库节点在子任务结果空间的写权限,不授予读权限,保证子任务计算结果彼此不可见的.只授予各数据库节点在服务节点发布子任务空间的读权限,不授予写权限,避免了数据库节点通过恶

意修改发布子任务的文件来干扰系统正常运行,导致任务失败.

- **安全的用户数据交互.**系统在用户交互空间为提出请求的用户创建子空间,设置为仅系统和该用户可读写.这样可以保证任务计算结果仅对提出任务请求的用户可见,避免非法用户窃取数据.
- **多租户的通信空间隔离**.为了实现对多租户的支持,保证通信层面安全的数据隔离,我们在系统任务空间和用户交互空间中都支持租户级别的数据隔离,仅授予数据库节点所在租户空间的读写权限.

2.2.2 无状态计算设计

函数无状态的挑战.现有系统方案基于节点状态建立长连接,保持在线状态并响应节点请求,从而实现节点间的协作计算.但是在无服务器系统中,函数是无状态的,无法建立长连接,需要设计无状态的节点管理,保证系统正确、高效地计算.同时,无状态的函数无法通过长连接驱动执行,需要设计恰当的执行模式,保证函数间正确、高效地交互.无状态的函数独立地完成节点的部分功能,需要根据任务特点进行合理的设计.

无状态的节点管理.本系统中的服务节点和数据库节点之间不存在长连接,需要对节点在线状态进行监测和管理.基于无状态函数的事件驱动特点,数据库节点定期主动上传包含时间戳的状态文件到共享云数据管道.服务节点读取状态文件,判断数据库节点的在线状态.通过节点管理,可以主动丢弃掉线节点,避免长时间等待.

请求驱动的执行模式.不同于现有系统的服务端通过保持在线来响应节点请求,本系统的服务节点在被请求驱动时启动并执行.请求以文件形式发布到共享云数据管道,在服务节点中,全局解析函数被用户的任务请求驱动,全局聚合函数被节点的聚合请求驱动,函数执行完成后自动销毁.这种执行模式克服了对长连接的依赖.

分治式查询计算.在本系统的查询任务中,丢弃对长连接和节点在线状态的依赖,结合中心式架构特点,遵循自上而下的分治思想,设计了解析查询请求-执行子查询-聚合子查询结果的分治式查询计算.设计细节如下:

- **服务节点:**对于查询任务,服务节点按照处理流程被分解为全局解析函数、全局聚合函数和辅助计算函数三个功能单元.全局解析函数负责解析用户的查询请求,生成数据库节点的子任务,是查询任务中服务节点的核心部分,以领导者的角色协调多方数据库节点共同完成计算任务.全局聚合函数负责对子任务的查询结果进行聚合计算.为克服同态加密技术的计算局限,设计辅助计算函数响应全局聚合函数的辅助计算请求,处理其无法完成的计算操作.
- **数据库节点:**数据库节点根据子任务文件的请求信息,基于私有数据进行查询和计算,将本地计算结果进行同态加密后返回给服务节点.本系统为不同部署方式的数据库节点设计了符合其特点的交互方式.部署在云服务器或者私有集群的数据库节点通过主动监听的方式与服务节点交互,部署在无服务器环境的数据库节点设定子任务请求文件的上传为触发事件,以事件驱动的方式与服务节点交互.

数据耦合的建模计算.数据耦合是一种将计算与数据归属相结合的方式.在多方数据库安全计算系统中,为保护用户数据隐私安全,需要以数据耦合的方式设计建模计算.在本系统的建模任务中,丢弃了对长连接和节点在线状态的依赖,并分别针对横向数据结构和纵向数据结构设计数据耦合的建模计算,细节如下:

- **横向数据建模:** 基于 Fedavg[14],FedProx[15]等聚合算法设计了横向数据建模.数据库节点持有数据完整特征,因此持有结构一致的本地模型,在本地进行模型训练后与服务节点进行交互完成多方安全建模计算.服务节点由节点采样函数、聚合计算函数构成.节点采样函数负责挑选本轮的参与者并负责无状态节点管理,在采样阶段过滤掉线的节点.聚合计算函数负责聚合计算节点上传的本地模型参数.
- **纵向数据建模.**基于分裂学习[19,20]的思想设计纵向数据建模.数据库节点持有部分数据特征,因此按照数据特征将模型进行纵向切分,各数据库节点在本地局部模型进行计算后,通过与服务节点交互完成样本损失和梯度的计算.服务节点根据模型计算的需要参与模型部分前向传播和梯度传播的计算过程,通过与共享云数据管道交互保存和恢复模型状态信息.

2.2.3 可用性和扩展性分析

通过设计无状态计算架构,克服了系统对长连接的依赖,提高了系统对节点掉线的容忍能力,并且通过无状态函数扩展实现了资源的弹性伸缩能力.针对所设计的系统架构,可用性和扩展性的分析如下:

- **掉线容忍度分析.**本系统的服务节点是请求驱动的执行模式,因此至少需要一个数据库节点在线,通

过上传本次子任务的计算结果来请求聚合,以驱动服务节点运行.在建模任务中,可以使用掉线的数据库节点上传的历史信息进行聚合.在查询任务中,可以跳过掉线的数据库节点,仅计算在线数据库节点上传的数据.因此,对数据库节点掉线容忍度的上限是 n-1,其中 n 代表数据库节点数量.

- **资源弹性伸缩分析.**系统级的资源自动伸缩由实例的弹性扩展完成,通过事件驱动式的无服务器函数的特性实现,由云平台保证有足够资源可被调度用于扩展实例.本系统支持设置并发实例数、CPU 和内存上限,以调整资源弹性伸缩幅度.并发实例数表示函数实例可并行处理的最大请求数,当前请求数量大于当前实例数量与并发实例数之积时,生成一个新实例.并发实例数越大,系统伸缩幅度越小.

3 系统实现

本系统支持在横向与纵向数据结构下的查询与建模任务.接下来将详细介绍 MpSDB 在多方数据库安全查询和建模方面的系统运行流程和设计评价.其中,系统设计与实现细节融入在运行流程的阐述中.

3.1 多方数据库安全查询

本系统支持横向数据结构和纵向数据结构下的多方数据库安全查询,允许用户跨多个数据库节点进行查询,同时保护原始数据隐私.系统支持的查询算子包含数据统计类(计数、求和、均值、标准差)和空间分析类(范围计数、范围查询和 kNN 查询),可以较好地适应多种数据结构类型和数值类型.

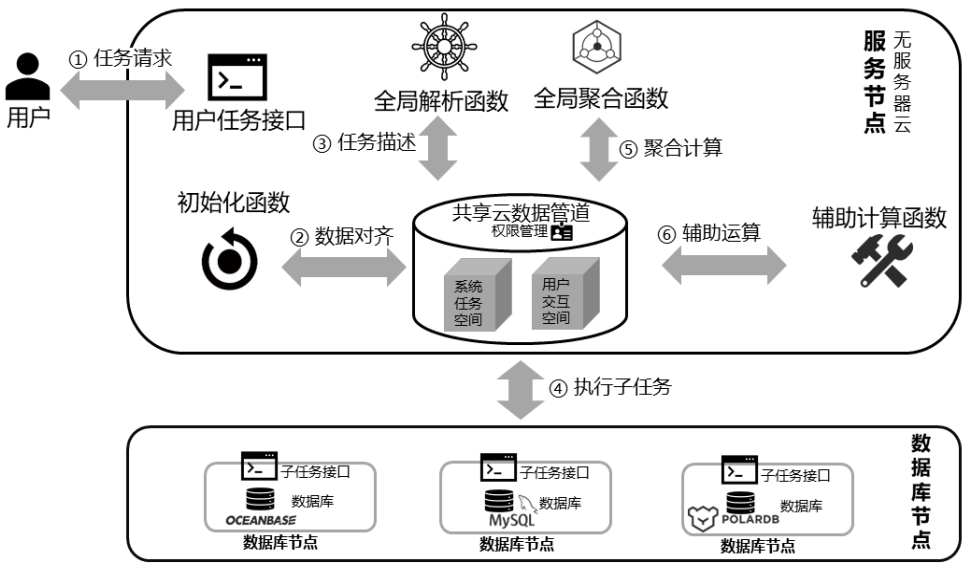

图 2 多方数据库安全查询任务运行流程

如图 2 所示,查询任务的系统运行流程从用户通过任务接口提交查询请求开始:

1. **任务请求:**用户提交的查询请求在通过身份验证后被上传到用户交互空间,触发系统开始执行任务.
2. **数据对齐:**系统通过初始化函数使用隐私集合求交(Private Set Intersection)技术进行数据对齐,确定横向数据结构中的共有特征集和纵向数据结构中的共有实例集.
3. **任务描述:**初始化完成后,服务节点的全局解析函数解析用户请求文件以获取任务参数,根据计算需求为数据库节点分派子任务,并将子任务请求以文件形式上传到系统任务空间.
4. **执行子任务:**数据库节点监测并读取系统任务空间的子任务请求文件,根据子任务文件信息生成查询语句,执行数据库查询和本地计算.使用同态加密技术加密计算结果,并上传到系统任务空间.

5. **聚合计算:**服务节点的全局聚合函数收集数据库节点上传的加密数据,运行相应算法进行密文状态下的聚合计算.在数据收集中,设计了超时判断机制,允许全局聚合函数跳过长时间未响应的数据库节点,使用现有的加密数据进行聚合,从而避免系统崩溃或长时间等待.
6. **辅助运算:** 辅助计算函数由全局聚合函数的辅助计算请求驱动,以克服同态加密技术在密文计算上的局限.辅助计算函数读取请求文件和加密数据进行辅助计算,包括除法、数值判断、开方、排序操作,然后对聚合计算结果进行解密并返回给用户.同时,辅助计算函数负责密钥生成、分发和管理.

上述系统运行流程与设计既保证了安全性,又实现了泛用性.**数据安全上**,数据库节点以密文计算结果交互,确保用户原始数据不离开本地环境,同时,严格的通信权限管理保证数据库节点无法读取彼此上传的密文信息.**系统安全上**,仅服务节点持有对子任务文件等系统控制信息的写权限,数据库节点无法恶意篡改文件干扰系统正常运行.**泛用性上**,数据库节点可根据数据库类型构造相应的查询语句,支持多种类型的数据库接入本系统.

3.2 多方数据库安全建模

本系统支持横向数据结构和纵向数据结构下的多方数据库安全建模,允许用户跨多个数据库节点进行建模计算,同时保护原始数据隐私.系统支持的建模算子包括线性回归、逻辑回归、多层感知机、卷积神经网络和 K 均值聚类,适用于有监督的回归和分类任务及无监督聚类任务.可以较好地适应多种数据结构类型和任务类型,并支持设置模型训练参数,包括学习率、批量大小、训练轮数和聚类簇心数量等.

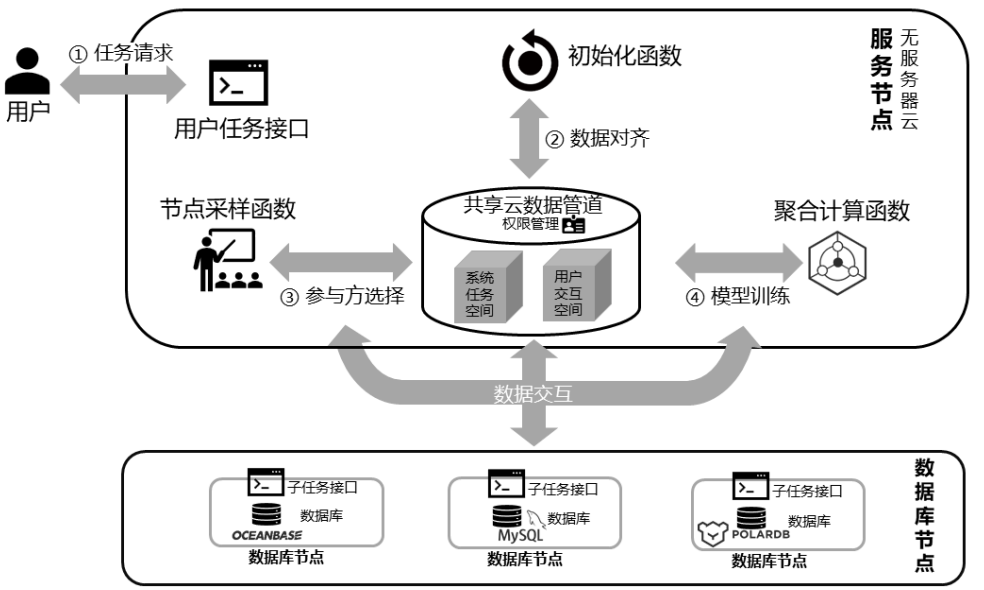

图 3 多方数据库安全建模运行流程

如图 3 所示,建模任务的系统运行流程从用户提交查询请求开始:

1. **任务请求:**用户通过上传任务请求文件来提交建模任务,文件中包括模型类型、模型参数等设置.
2. **数据对齐:**系统通过初始化函数使用隐私集合求交(Private Set Intersection)技术进行数据对齐,确定横向数据结构的共有特征集和纵向数据结构的共有实例集.
3. **参与方选择:**节点采样函数检测数据库节点的在线状态,并从在线的数据库节点中采样出参与本轮联合训练的节点,随后将采样结果上传到共享存储空间.
4. **模型训练:**所有被采样的数据库节点使用其私有数据在本地进行模型训练,并通过与服务节点的数据交互来完成联合训练,未被采样的节点将等待参与下一轮训练.不同数据结构下的具体流程如下:
   a. **横向数据结构:**各个数据库节点持有的数据拥有完整特征空间和标签,因此所有数据库节点的

本地模型采用相同的模型结构.数据库节点完成本地模型训练后上传模型参数.系统实现FedAvg[14],FedProx[15]等多种联邦聚合算法对各节点的模型参数进行聚合计算.数据库节点读取聚合计算结果,更新本地模型参数,并定期上传状态信息用于节点管理.

b. **纵向数据结构:**各个数据库节点持有的数据拥有部分特征空间,因此仅持有被分割的局部模型.这些数据的标签由一个数据库节点持有.按照各数据库节点、服务节点、持有标签的数据库节点的顺序进行模型的前向传播计算,并进行反向传播完成梯度更新.

上述系统运行流程与设计既保证了安全性,又实现了计算正确性.**数据安全上**,数据库节点以本地模型计算结果交互,确保用户原始数据不离开本地环境.同时,严格的通信权限管理保证数据库节点无法读取彼此上传的模型计算结果.**系统安全上**,仅服务节点持有对采样结果文件等系统控制信息的写权限,数据库节点无法篡改文件干扰系统正常运行.**计算正确性上**,数据库节点基于数据结构特点选择合适的模型进行数据耦合式本地计算,适应多种数据结构.同时,设计控制节点交互进程的协作机制,保证无状态函数之间有序的交互和正确的计算.

4 实验分析

4.1 实验设置

4.1.1 实验环境

MpSDB 的部署分为服务节点和数据库节点两部分.数据库节点支持多种接入方式,本实验考虑实际应用场景,租赁云平台的虚拟机部署 50%的数据库节点,另外 50%部署在云平台的无服务器计算环境中.在使用云平台的无服务器计算环境时,服务节点和数据库节点的资源设置为4vCPU和4GB.在使用云平台的虚拟机时,服务节点和每个数据库节点分别租赁一个配置为4vCPU和4GB的通用计算型实例,操作系统为Ubuntu 20.04.6 LTS.由于对比系统中的 Conclave[8]最多支持 3 个参与方,本实验使用 3 个数据库节点测试系统的计算成本和性能,而在扩展性的实验中将数据库节点数量增加到 10.对于无服务器计算系统,使用阿里云 FC 函数计算服务作为无服务器计算环境,使用阿里云 OSS 对象存储服务实现共享云数据管道.实验中所有系统使用阿里云OceanBase 在数据库节点上存储和管理本地数据,但是本系统不限制数据库节点选择的数据库类型.

4.1.2 实验数据

实验共涉及查询和建模两种任务类型,选用 OpenStreetMap[43],imis-3months[44],NYC taxi[45],TPC-H[46]四个数据集作为查询任务数据,选用 Wine Quality[47],Breast Cancer[48],MNIST[49]三个数据集作为建模任务数据.对数据集进行采样并过滤掉冗余信息,得到的各个数据集的说明如表 1 所示.

表 1 实验数据集

数据集	数据类型	数据量	特征数量	任务类型	标签类型
OpenStreetMap	空间坐标 Geometry('POINT(x, y)')	30,000,000	2	查询 空间查询任务	无
imis-3months	空间坐标 Geometry('POINT(x, y)')	30,000,000	2	查询 空间查询任务	无
NYC taxi	表格型数据 数据类型:Float, int	30,000,000	4	查询 统计查询任务	无
TPC-H	表格型数据 数据类型:Float, int	30,000,000	4	查询 统计查询任务	无
Wine Quality	表格型数据 数据类型:Float, int	6,497	11	建模 回归、聚类任务	int 型 3 ~ 9
Breast Cancer	表格型数据 数据类型:Float, int	569	30	建模 分类任务	int 型 0, 1
MNIST	图像集 数据类型:Float	60,000	1x28x28	建模 分类任务	int 型 0 ~ 9

数据集介绍.OpenStreetMap (OSM) 是一个记录地图上对象地理坐标的公开数据集每条记录具有经度

和纬度两个特征,用于查询任务.**imis-3months** 是由 IMIS Hellas S.A.公司收集的记录地理点的公开数据集,每条记录具有经度和纬度两个特征,用于查询任务.我们分别对以上三个数据集进行去重并随机采样 3 千万条记录.**NYC taxi** 是纽约公共出租车旅行票价信息,每条记录具有乘客数,里程,燃油费,总费用四个特征,用于查询任务.**TPC-H** 是一个用于评估和比较数据库管理系统性能的基准测试数据集,设定数据集生成规模为 150GB,即 3 千万条产品记录.我们选择产品的尺寸、数量、售价、批发价四个数值型特征,用于查询任务.**Wine Quality** 是对葡萄酒进行化学分析的公开数据集.该数据集共有 6497 个样本,每个样本有 11 个特征,用于建模任务中的回归任务和聚类任务.**Breast Cancer** 是威斯康星州乳腺癌数据集.该数据集由 569 个样本构成,每个样本的特征数是 30,用于建模任务中的分类任务.MNIST 是一个手写数字图像数据集.该数据集由 60000 张 28x28 像素的图像的训练集和 10000 张图像的测试集组成,用于建模任务中的分类任务.

实验数据集处理.将 Wine Quality 和 Breast Cancer 数据集按照 4:1 的比例划分为训练集和测试集,MNIST 数据集有独立的测试集.将数据集进行纵向和横向的划分:横向数据结构中每个数据库节点的数据具有相同的特征空间和不同的样本空间;纵向数据结构中每个数据库节点的数据具有不同的特征空间和相同的样本空间.

4.1.3 实验任务

本实验需要在横向数据结构和纵向数据结构下进行查询和建模两类任务的测试.下面介绍详细实验设置.

对于查询任务. 在 NYC taxi 和 TPC-H 数据集上运行计数、求和、求均值、求标准差四种统计查询任务,在横向数据结构时,设置查询数据库列名分别为乘客数和尺寸,并指定需要进行的查询操作.在纵向数据结构时,设置数据库列名分别为里程、燃油费、总费用和数量、售价、批发价,并指定需要进行的查询操作.在 OSM 和 imis-3months 数据集上运行范围计数,范围查询,kNN 查询三种空间查询任务,在数据集中随机选择一个坐标作为查询任务的中心点.对于范围计数和范围查询,参照 OpenHuFu 的实验,设置两个数据集上的范围距离分别为 0.012 和 0.05,此时范围内数据占比约为 0.5%.对于 kNN 查询,设置 k 值为 8.由于 OSM 和 imis-3months 数据集只有 2 个特征,无法在 3 个数据库节点时纵向切分,因此仅在横向数据结构下进行实验.

对于建模任务.选择五种机器学习模型:线性回归模型 Linear,逻辑回归模型 LR,多层感知机模型 MLP(第一层 30x150,第二层 150x50),卷积神经网络 CNN(包含两个 5x5 卷积层,一个 0.5 的 dropout 层,320x50 和 50x10 的两个全连接层)和 k 均值聚类 kMeans.在实验数据上,Linear 使用 Wine Quality 数据集,LR 和 MLP 使用 Cancer 数据集,CNN 使用 MNIST 数据集,kMeans 使用 Wine Quality 数据集.在实验设置上,k 均值聚类的 k 设置为 3,表格数据集的 batch size 大小设置为训练集的 10%,CNN 设置 batch size 为常用值 512,使用 SGD 优化算法并在[0.001,0.1]的范围内通过网格搜索得到最佳的学习率.纵向数据结构下随机切分数据集的特征.

实验指标.为了全面地对系统性能进行分析和评估,实验分析中比较系统执行计算任务的运行时间,同时比较运行任务的成本以评估系统的经济性.

4.1.4 对比系统

实验中选择如下四种具有代表性的多方数据库安全计算系统:

- **Conclave[8]系统.**支持横向数据结构的计数、求和、平均值、标准差四种统计查询,部署在云服务器.
- **OpenHuFu[10]系统.**支持横向数据结构的范围计数、范围查询和 kNN 查询,部署在云服务器.
- **MpSDB_IaaS 系统.**以上两种系统只支持横向数据结构的查询.为了评估有服务器环境下纵向数据结构的查询的性能,我们将 MpSDB 进行修改并部署在云服务器,记为 MpSDB_IaaS.
- **FedML[22]系统.**对于建模任务,选择 FedML 作为对比系统,分别针对横向数据结构和纵向数据结构,实现了本系统支持的五种机器学习模型,部署在云服务器.

4.2 无服务器与有服务器系统的计算成本对比

无服务器计算采用按使用计费的模式,而有服务器计算(如云服务器、私有集群)按照租赁时间产生成本,因此系统的成本成为一个有趣的分析对象.根据第 4.3 节中的系统性能数据,相比于其他部署在有服务器环境中的系统,MpSDB_IaaS 系统在各类任务上的运行时间最短,因此选择其作为有服务器系统的代表;由于只有本系统 MpSDB 是无服务器系统,因此选择其作为无服务器系统的代表.

表 2 云计算平台计费单元单价

计费单元	阿里云 FC 函数计算		阿里云 OSS 存储	阿里云服务器
	内存单价 (元/GB*秒)	vCPU 单价 (元/vCPU*秒)	请求单价 (元/万次)	租赁单价 (元/台/小时)
价格	0.000009	0.00009	0.01	0.697

在本实验中,我们关注系统实现形式不同导致的任务成本差异.云服务器的启动会产生费用,计入任务成本.无服务器系统 MpSDB 的服务节点以阿里云的 FC 函数计算实现,共享云数据管道以 OSS 对象存储服务实现,用于系统内部计算的通信.因此,对这两个服务的用量和产生的成本进行统计.这些服务的计费单价如表 2 所示.

由表 2 可知,FC 函数计算的 vCPU 单价是内存单价的十倍,是 FC 函数计算的主要开销.FC 函数计算的请求调用费用不到总费用 1%,因此我们忽略掉这个计费项.OSS 存储作为通信介质,文件被读取后被及时删除,存储费用不到总费用 1%,因此 OSS 存储只计算请求费用.我们按照任务类型分别在下面两个小节中比较计算成本.

4.2.1 查询任务成本对比

首先在固定频次查询下对比和分析任务成本,考虑到查询任务的成本与数据量相关,按照 4.1 节的实验设置分别在 1k、10k、100k、1M、10M 数据量的 NYC taxi、TPC-H、OSM、imis-3months 数据集上将各查询算子重复运行 10 次,统计查询算子在 5 个数据量上的平均成本作为该查询算子的成本.虽然云平台上服务器的最小租用单位是小时,我们在统计有服务器系统的成本时,首先按实际运行时间计算成本,这是有服务器系统最理想的成本.具体而言,启动云服务器并在一个数据集的 5 个数据量上运行 10 次查询算子,统计平均运行时间作为该查询算子的云服务器租赁时长.无服务器和有服务器系统运行查询任务的计算成本如表 3 所示.

表 3 查询任务计算成本

数据分布	查询类型	查询算子	无服务器系统(MpSDB)				有服务器系统(MpSDB_IaaS)	
			FC 函数计算		OSS 存储	任务成本 (元)	云服务器	任务成本 (元)
			内存用量 (GB*秒)	vCPU 用量 (vCPU*秒)	请求用量 (万次)		租赁时长 (秒)	
横向	统计查询	计数	320	320	0.1862	**0.0335**	87	0.0671
		求和	329	329	0.1806	**0.0343**	88	0.0681
		求均值	700	700	0.3700	**0.0730**	96	0.0743
		求标准差	1296	1296	0.6252	0.1346	123	**0.0951**
	空间查询	范围计数	384	384	0.2157	**0.0402**	97	0.0751
		范围查询	382	382	0.2217	**0.0400**	98	0.0755
		kNN 查询	410	410	0.2286	**0.0429**	102	0.0786
纵向	统计查询	计数	338	338	0.1942	**0.0354**	89	0.0689
		求和	349	349	0.2027	**0.0366**	91	0.0704
		求均值	350	350	0.1987	**0.0366**	91	0.0702
		求标准差	393	393	0.2019	**0.0410**	91	0.0701

实验数据表明,在运行较低固定频次(10 次)的简单查询算子时,无服务器系统 MpSDB 表现出更好的成本优势.原因是云服务器启动时间冗长,而任务时间相对较短.有服务器系统 MpSDB_IaaS 的服务器启动时间在总任务时间中占比较大,并且在任务过程中,服务器完成计算后处于等待状态,这些启动和等待的时间也产生租赁费用.相比之下,无服务器系统因启动时间短、资源回收、按使用计费的特点,在低频的简单查询算子上表现出更好的成本优势.对于复杂查询算子(如横向数据结构下求标准差),无服务器系统 MpSDB 存在一定的成本劣势,这是由于复杂查询算子需要更加频繁地启动函数和通信,函数计算因等待通信和函数启动而产生更多的成本.

随后,我们进一步分析随查询频次的增加,成本的变化情况.云服务器的最小租赁时间单位是小时,我们增加无服务器系统 MpSDB 一小时内的查询次数,并记录成本变化.表 3 数据表明,部分查询任务成本相似,因此选取四个具有代表性的查询算子,统计它们在 10M 数据量上的任务成本随查询次数的变化,实验结果如图 4 所示.

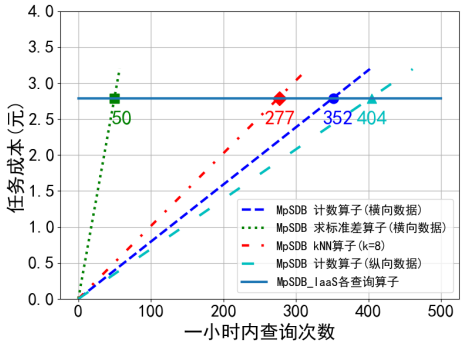

图 4 查询任务成本随查询频次变化图

实验数据表明,无服务器系统 MpSDB 的四种查询算子的任务成本随查询次数的增加而线性增加,这符合无服务器计算资源弹性伸缩的特点,当查询次数较少时,产生的计算成本比有服务器系统更低.有服务器系统 MpSDB_IaaS 以小时为最小单位租赁云虚拟机,因此在一小时内运行任意多次查询算子产生的计算成本相同.

无服务器计算系统 MpSDB 与有服务器系统 MpSDB_IaaS 的计算成本曲线在某个查询次数出现计算成本的交叉.不同的查询算子由于计算复杂度不同,在无服务器计算系统中产生不同的成本,因此达到与有服务器系统相同成本的查询频次也各不相同.例如,在横向数据结构上运行求标准差算子,查询频次小于 50 次/小时时,无服务器系统的成本更低,当查询频次大于 50 次/小时时,有服务器系统的成本更低.

4.2.2 建模任务成本对比

本小节中分析建模任务的系统成本对比.我们按照 4.1 节的实验设置,分别在横向数据结构和纵向数据结构下运行五种机器学习模型.两种系统的数据库节点采样相同的部署方式,因此只统计服务节点的成本,服务节点的计费单元与 4.2.1 节相同.表 4 统计了每个模型的建模任务的计算成本.

表 4 建模任务计算成本

数据分布	查询算子	无服务器系统(MpSDB)				有服务器(FedML)	
		FC 函数计算		OSS 存储	任务成本(元)	云服务器租赁时长(秒)	任务成本(元)
		内存用量(GB*秒)	vCPU 用量(vCPU*秒)	请求用量(万次)			
横向	Linear	105.4	105.4	0.0835	**0.0112**	80.628	0.0156
	LR	118.8	118.8	0.0833	**0.0126**	80.089	0.0155
	MLP	133.5	133.5	0.0860	**0.0141**	81.419	0.0158
	CNN	169.5	169.5	0.0891	**0.0177**	422.793	0.0819
	kMeans	110.5	110.5	0.0705	**0.0116**	81.978	0.0159
纵向	Linear	789	789	0.5563	0.0837	85.355	**0.0165**
	LR	823	823	0.5016	0.0865	85.301	**0.0165**
	MLP	797	797	0.5129	0.0840	83.53	**0.0162**
	CNN	11800	11800	6.7024	1.2352	163.857	**0.0317**
	kMeans	670	670	0.4048	0.0703	84.158	**0.0163**

实验数据表明,无服务器系统在横向数据结构下的建模任务成本更低.这是因为无服务器计算实例可以快速启动,并在等待数据库节点时弹性伸缩并释放资源.在横向数据结构下进行建模任务,无服务器系统服务节点的聚合计算函数和采样参与者函数启动仅需 1 秒左右,而有服务器系统的云服务器启动时间高达 79 秒,启动时间产生了更高的成本.并且,在横向数据结构下运行建模任务,无服务器系统的服务节点的函数只在进行全局聚合时启动,完成聚合后自动销毁,不再产生计算成本;而有服务器系统需要保持服务器运行,产生更多的计算成本.例如,在 CNN 模型的建模任务中,因为模型更复杂,训练数据集更大,本地训练时间比其他模型的本地训练时间多出约 340 秒,有服务器系统运行 CNN 模型的建模任务成本因更长的等待时间而产生更高的任务成本,而无服务器系统运行 CNN 模型的建模任务成本仅因聚合计算量增加而少量增加.因此,在横向数据结构下,数据集

越大、模型越复杂,数据库节点的本地训练时间越久,无服务器系统的成本优势越明显.

与横向数据结构的实验结果不同,无服务器系统在纵向数据结构下的建模任务成本更高.这是因为在纵向数据结构下,采用分裂学习架构,模型计算被分解到多个节点,导致无服务器系统的通信更为频繁,并且服务节点的函数使用量更大.表 4 中实验数据表明,同一个模型在不同数据结构下通信频繁程度和函数计算量存在显著差异.例如, CNN 模型在纵向数据结构下的 OSS 存储请求次数约为横向数据结构下请求次数的 100 倍,FC 函数计算用量约为横向数据结构下 FC 函数计算用量的 100 倍.对比纵向数据结构下两种系统的任务成本,频繁的通信给无服务器系统带来了更高的 OSS 存储文件读写请求成本,并且服务节点需要完成更多的计算任务,造成无服务器系统的函数使用量更大,计算成本更高.总体而言,在纵向结构下,数据集越大,模型训练的轮数越多,通信频率越高,或服务节点的计算量越大时,无服务器系统的成本劣势越明显.

4.2.3 小结

在查询任务上,与有服务器系统相比,无服务器系统因函数快速启动和按使用计费的特点,整体上在较低频次的查询任务上成本更低,其中,横向数据结构下求标准差算子因频繁地启动函数和通信表现出更高的成本.但是,无服务器函数的计算资源单价比服务器的计算资源单价更高,并且通信成本包含了文件读写请求的费用.因此,随查询次数增加而增加,无服务器系统在高频次的查询任务上成本更高.

在建模任务上,与有服务器系统相比,无服务器系统在横向数据结构下的建模任务成本更低,在纵向数据结构下的建模任务成本更高.在横向数据结构下,无服务器系统得益于弹性伸缩的特点,不必在等待数据库节点聚合请求时产生计算成本;在纵向数据结构下,无服务器系统因高频的通信和高额的计算量产生更高的成本.

4.3 系统性能实验

本节比较各系统的系统性能,按照 4.1 节的实验设置提交各类计算任务请求并统计任务时间.在无服务器系统的任务时间中计入了函数计算实例的启动时间,因此在有服务器系统的任务时间中同样计入服务器的启动时间.对所有任务,重复运行 10 次计算运行时间的平均值.下面分别比较查询任务和建模任务的系统性能.

4.3.1 查询任务系统性能

Conclave 系统支持计数、求和、求均值、求标准差四个统计查询算子,OpenHuFu 系统支持范围计数、范围查询和 kNN 查询三个空间查询算子.按照 4.1 节的实验设置,在 NYU taxi、TPC-H 数据集和统计查询算子上比较 MpSDB、MpSDB_IaaS、Conclave 三个系统,在 OSM、imis-3months 数据集和空间查询算子上比较 MpSDB、MpSDB_IaaS、OpenHuFu 三个系统.在不同数据结构的数据集上分别统计它们在 5 个数量级上的任务时间.

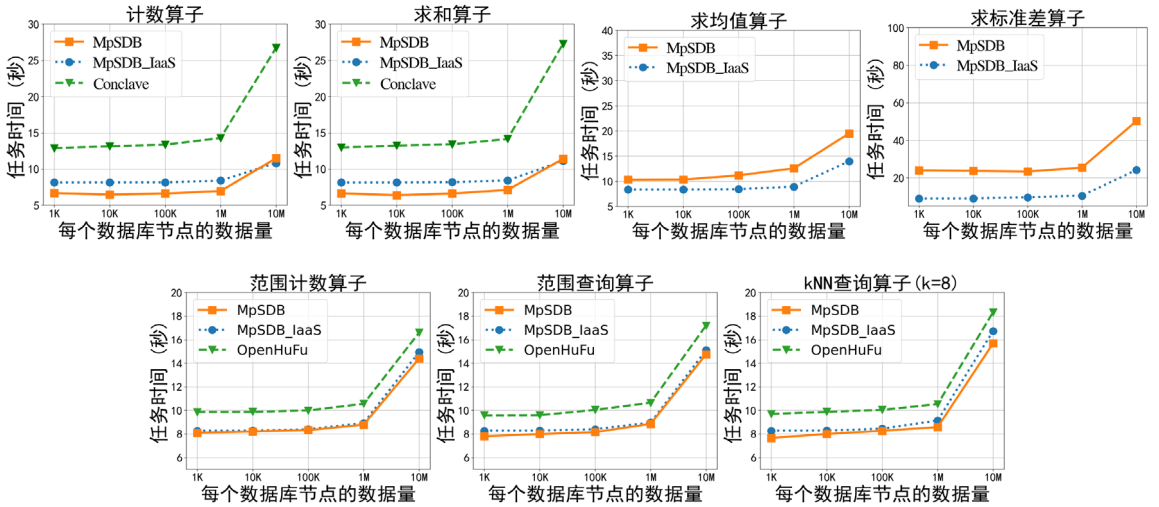

图 5 横向数据结构下查询任务性能比较(NYU taxi、OSM 数据集)

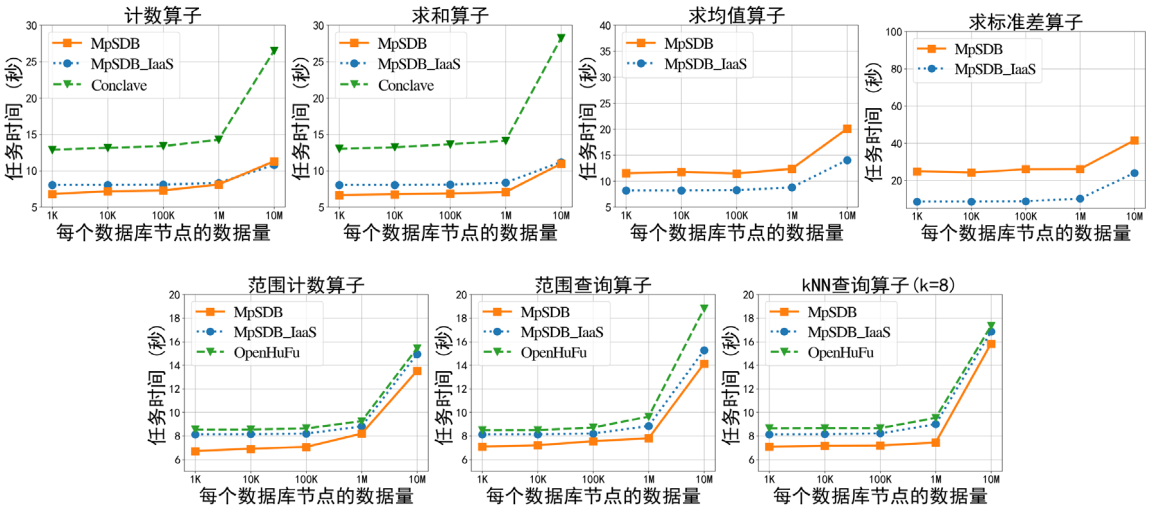

图 6　横向数据结构下查询任务性能比较(TPC-H、imis-3months 数据集)

横向数据结构.横向数据结构下查询计算任务的系统性能对比如图 5 和图 6. Conclave 受限于加密算法的数值界限限制,在求和任务中当数据量过大时任务失败(在 10M 数据量的 NYU taxi 数据集和 1M,10M 数据量的 TPC-H 数据集上).我们将求和算子计算的属性值均改为 1,此时求和结果在数值界限中,可以完成计算,以该情况下算子运行时间作为任务时间并绘制在图 5 和图 6.在求均值和求标准差两个算子的实验上,因为基于秘密共享的隐私计算开销大,导致两个算子的运行时间显著增长,无法在合理的时间内(超过 1 小时)完成计算,因此在图 5 和图 6 的不予绘制.

实验结果表明,与 Conclave 和 OpenHuFu 相比,本系统 MpSDB 表现出更好的系统性能.对比有服务器系统 MpSDB_IaaS,当查询算子较简单时,启动时间占比高,MpSDB 因函数快速启动的优势表现出比 MpSDB_IaaS 更好的性能,但对于复杂查询算子(如求均值、求标准差),需要频繁地启动函数和通信,因而性能低于 MpSDB_IaaS.

在上述图 5、图 6 中,我们观察到在数据量从 1M 增加到 10M 时,算子的查询时间迅速增加.这是由于 x 轴没有采用线性坐标轴的形式导致的.我们以图 6 中横向数据结构下范围查询算子为例,选择[100K, 1M, 10M]数据量,x 轴分别采用线性坐标轴形式和对数坐标轴形式,算子查询时间的变化如图 7 所示.可以观察到,数据库中数据量变大时,查询算子的任务时间会随之增长,整体呈现线性增加的趋势.

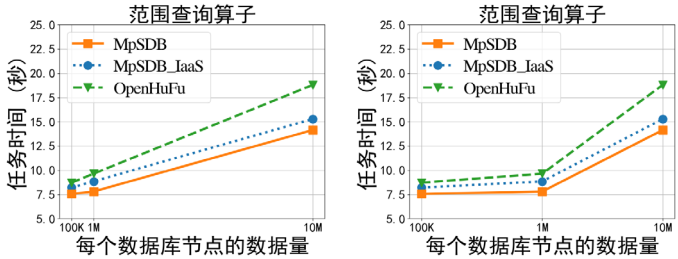

图 7　x 轴形式对算子查询时间变化的影响

在上述图 5、图 6 的 kNN 算子性能比较中,我们将 k 的取值固定为 8.为了分析 k 的取值是否对 kNN 算子性能产生影响,我们令每个数据库节点持有 10M 数据,在[3,20]范围内改变 k 的取值,分别测量系统在 OSM、imis-3months 数据集上的运行时间,实验结果如图 8 所示.可以观察到,kNN 算子的运行时间不随 k 值的增大而显著增加,这是因为查询开销主要与数据库中的数据量有关,k 的取值变化时,不会导致系统的通信量和计算量明显增加,整体开销几乎不变.

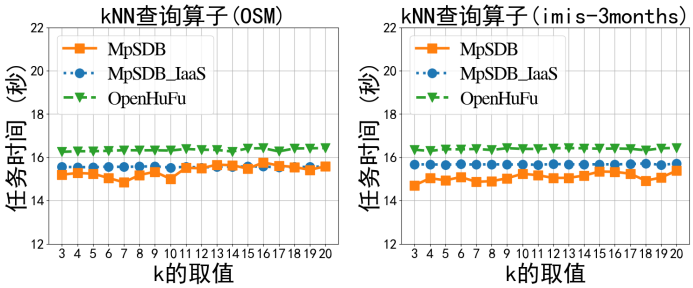

图 8 横向数据结构下 k 的取值对 kNN 查询算子性能的影响

纵向数据结构.由于 Conclave 和 OpenHuFu 不支持纵向数据结构下的联合查询,并且 OSM 和 imis-3months 数据集只有两个特征.因此对比 MpSDB 和 MpSDB_IaaS 在 NYU taxi 和 TPC-H 两个纵向数据集上查询算子的任务时间,如图 9 和图 10 所示.

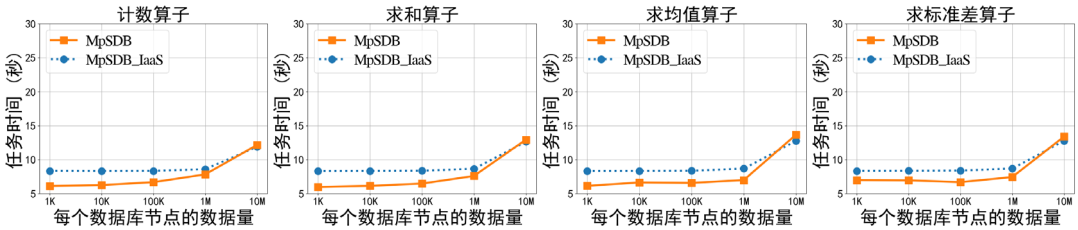

图 9 纵向数据结构下查询计算任务性能比较(NYU taxi 数据集)

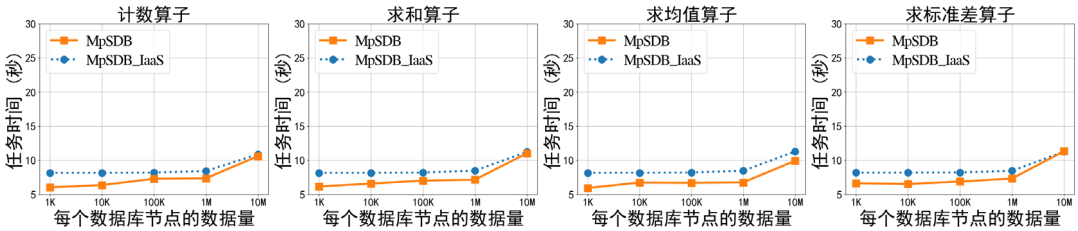

图 10 纵向数据结构下查询计算任务性能比较(TPC-H 数据集)

实验结果表明,MpSDB 在纵向数据结构下表现出比 MpSDB_IaaS 更好的系统性能.在纵向数据结构的查询任务中,系统启动时间占比大,MpSDB_IaaS 的服务器启动时间较长,而 MpSDB 的无状态函数启动时间较短.与横向数据结构不同,纵向数据结构下求均值、求标准差等复杂查询算子可在一个数据库节点上处理一个特征的所有值,不需要频繁地启动函数和通信,因而 MpSDB 在这些查询算子上依然表现出较好的系统性能.

4.3.2 建模任务系统性能

由于 Conclave 和 OpenHuFu 系统不支持建模任务,本节比较 MpSDB 系统和 FedML 系统所支持的建模任务,实验结果如表 5 所示.

表 5 建模任务运行时间(单位为秒)

数据结构	系统	Linear	LR	MLP	CNN	kMeans
横向	FedML	80.628	80.089	81.419	**422.793**	81.978
	MpSDB	**31.111**	**31.167**	**35.808**	425.897	**46.182**
纵向	FedML	**85.355**	**85.301**	**83.530**	**163.857**	**84.158**
	MpSDB	272.124	312.619	246.542	3521.152	219.638

横向数据结构. 实验结果表明,本系统 MpSDB 在横向数据结构的建模任务上表现出一定的性能优势.对于横向数据结构,本系统得益于函数快速启动的优势,在较小数据集上表现出明显的时间优势,例如 Linear/LR/MLP/kMeans 四个模型.当模型和数据集较大时,训练时间增加,本系统快速启动的时间优势不足以抵消函数多次启动和间接通信延迟带来的劣势,表现出较差的系统性能,例如 CNN 模型.

纵向数据结构. 本系统 MpSDB 在纵向数据结构的建模任务上性能明显比 FedML 更差.纵向数据结构下,本系统的通信和函数启动频率增加,大量的通信延迟和函数启动导致任务时间增加.例如,CNN 模型的通信轮数约为 MLP 模型的 10 倍,本系统在 MLP 模型上比 FedML 慢 1.95 倍,而在 CNN 模型上比 FedML 慢 20.5 倍.

时间分解. 为了进一步展示本系统 MpSDB 与 FedML 在不同数据结构下建模任务的优势和劣势,以线性回归模型(Linear)、多层感知机模型(MLP)和卷积神经网络模型(CNN)为例,统计了平均一轮模型计算中任务启动时间、通信时间、计算时间和总时间进行分析,实验结果如图 11、图 12、图 13 所示.

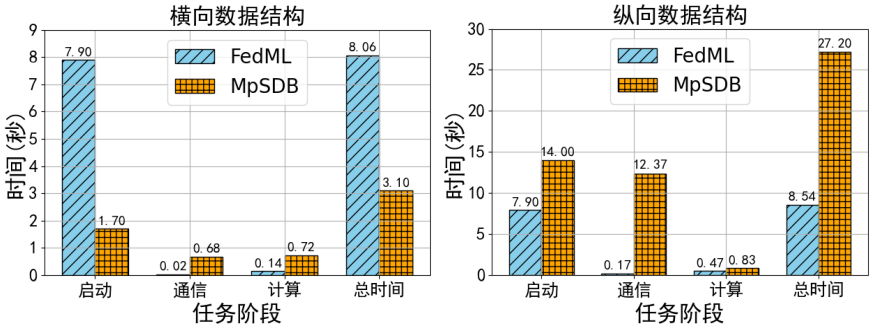

图 11 Linear 模型训练的时间分解

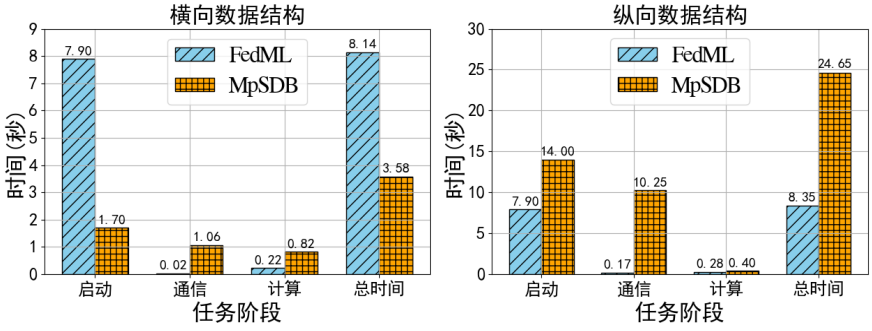

图 12 MLP 模型训练的时间分解

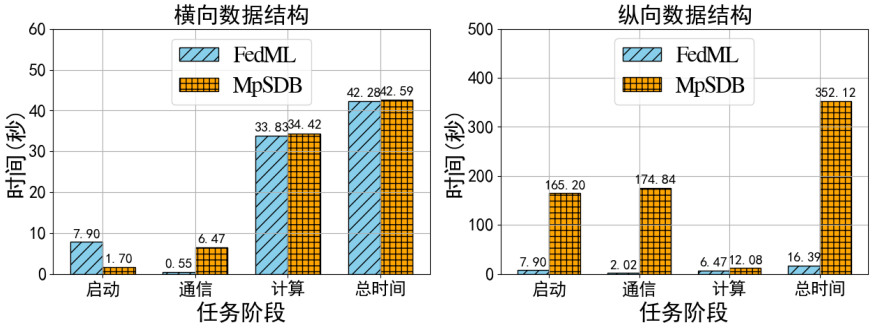

图 13 CNN 模型训练的时间分解

在横向数据结构下,一轮训练中仅进行一次通信,与较长的服务器启动时间相比,较小模型(Linear、MLP)

的无服务器计算函数启动时间的优势弥补了通信开销上的劣势,在总时间上仍然更优.但是较大模型(CNN)的通信开销更大,总时间略长.当减少训练轮数时,FedML 平均一轮的启动时间增加,较大模型(CNN)的总时间将比 MpSDB 系统更长.在纵向数据结构下,根据 4.1 节的实验设置,一轮训练需要频繁地启动函数和通信,导致系统性能下降.总体而言, MpSDB 更适用于横向数据结构下较小模型较少轮数的建模任务.

4.3.3 小结

在查询任务上,本系统 MpSDB 比 Conclave 和 OpenHuFu 表现出更好的系统性能.当查询频率较低时,MpSDB 的函数启动迅速,具有较为明显的时间优势.当数据量较小时, MpSDB 在横向数据结构下的简单查询算子和纵向数据结构下的查询算子上,表现出比 MpSDB_IaaS 更短的任务时间.但 MpSDB 在横向数据结构下的复杂查询算子因频繁地启动函数和通信,表现出比 MpSDB_IaaS 更长的任务时间.在横向数据结构下的建模任务中, MpSDB 因快速启动的特性表现出比 MpSDB_IaaS 更好的系统性能.在纵向数据结构下的建模任务中,频繁地启动函数和通信导致 MpSDB 表现出比 MpSDB_IaaS 更差的系统性能.

4.4 系统可扩展性实验

系统的可扩展性是本工作的重要目标.在本小节中,我们从数据库节点数量和用户任务数量两个角度,以横向数据结构下的查询任务为例进行系统扩展性实验,建模任务的实验结果相似.由于 Conclave 系统最多支持 3 个参与方,在数据库节点数量的实验中,对比 MpSDB、MpSDB_IaaS 和 OpenHuFu 三个系统.在用户任务数量的实验中,由于 MpSDB_IaaS 在有服务器系统中性能最优,选择其与 MpSDB 进行对比实验.

4.4.1 数据库节点数量

分别切分 OSM 和 imis-3months 数据集使得每个数据库节点持有 3M 数据,将节点数量从 2 增加到 10,运行范围计数、范围查询和 kNN 查询(k=8)三种查询算子各 10 次并取平均运行时间,实验结果如图 14 和图 15 所示.

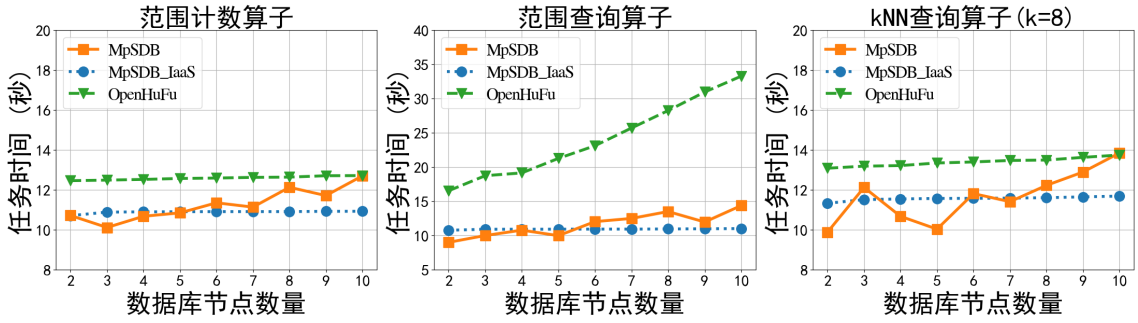

图 14 数据库节点数量的系统扩展性实验(OSM 数据集)

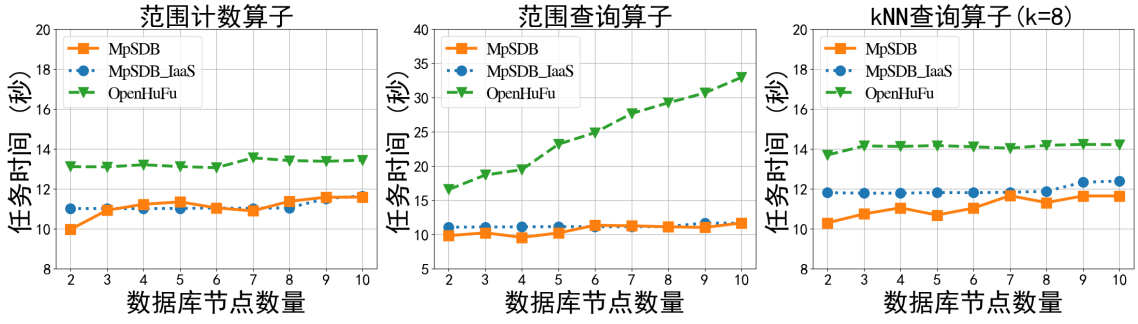

图 15 数据库节点数量的系统扩展性实验(imis-3months 数据集)

范围计数算子与 kNN 查询算子在每个数据库节点上产生的查询结果较少,因此通信开销在总时间中占比

较小,当数据库节点数量增加时,算子的任务时间无显著增加.对于范围查询算子,每个数据库节点产生较多的查询结果,OpenHuFu 使用基于秘密共享的隐私计算技术,数据通信的流程复杂,通信开销在总时间中占主导,因此算子的任务时间随数据库节点数量增加而显著增加;而本系统 MpSDB 的通信过程较简单,通信开销在总时间中占比不大,当数据库节点数量增加时,算子的任务时间无显著增加.

4.4.2 用户任务数量

本小节继续以三种空间查询任务为例,分析在用户流量变化的情况下系统的可扩展能力.在 4.1 节的实验设置下,3 个数据库节点各拥有 10M 的数据量,用户在同一时间并发地提出不同数量的查询请求,统计计算任务运行的平均时间.本实验重点关注系统在用户请求流量变化场景下的响应时间变化情况,因此有服务器系统的任务时间不计入服务器启动时间,实验结果如图 16 和图 17 所示.

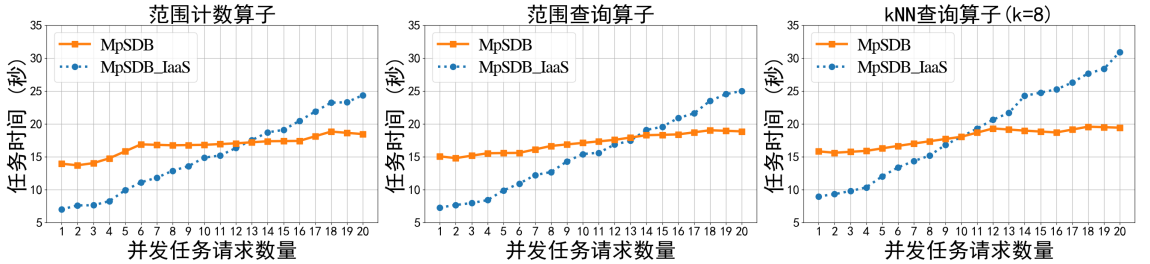

图 16 用户任务数量的系统扩展性实验(OSM 数据集)

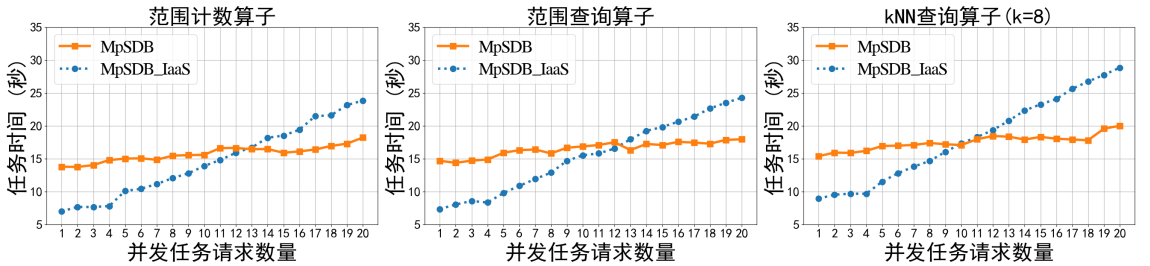

图 17 用户任务数量的系统扩展性实验(imis-3months 数据集)

实验结果表明,随着用户任务请求流量的增加, MpSDB 可以通过资源弹性伸缩有效应对变化的流量,任务时间未呈现出明显的增加,表明本系统具有良好的扩展能力.有服务器系统 MpSDB_IaaS 在并发请求数增大时会达到系统资源瓶颈,随着任务请求流量的增加,任务运行时间逐渐增加,并在并发请求数大于 13 后表现出比 MpSDB 更长的任务时间.在并发请求数较小时有服务器系统的任务时间更短,但是任务时间不包含服务器启动时间,而本系统 MpSDB 计入了函数启动时间.

4.4.3 小结

随着数据库节点数量的增加, MpSDB 的系统性能没有明显的增加,具有良好的扩展性.在用户请求流量变化的场景下,有服务器系统在达到资源瓶颈后计算任务响应时间变长,而 MpSDB 可以通过弹性扩展计算资源保持较稳定的任务响应时间,证明其具有良好的资源弹性伸缩能力扩展性.

5 总结与展望

本文针对多方数据库安全计算面临的系统扩展性和可用性不足的问题,设计并实现了一种多方数据库安全计算系统 MpSDB (Multi-party Serverless Database).MpSDB 实现了基于无服务器计算的多方数据库安全计算架构,通过请求驱动的执行模式支持松散的多方数据库协作场景,提高了系统对掉线者的容忍度,MpSDB 在无服务器计算环境下实现了多方数据库之间安全通信机制,实现多节点之间的数据交互和任务协作.我们将 MpSDB 与现有系统从成本、性能、扩展性等多角度进行实验对比,实验结果表明,MpSDB 在低频次的查询任

务(横向数据结构下复杂查询算子除外)、横向数据结构下的建模任务中表现出系统性能和使用成本的优势.

本系统在无状态函数的启动和通信上存在局限性,分析如下:

- **函数启动代价受云平台限制.**函数启动代价包括异步排队时间和实例启动时间.异步排队时间受云平台内部调度算法和业务流量影响,但不对外提供修改接口,难以进行优化.实例启动时间在本系统设计中已通过细粒度解构进行优化,本系统支持调整实例保留时间以在启动代价和成本间权衡.
- **函数通信开销受云平台限制.**商用云平台的无服务器函数仅对外提供调用接口,无法通过修改源码实现更高效的底层通信.本工作以读写云存储服务实现间接通信.但是,云存储服务只提供读写接口,封闭实现细节,难以进行优化.未来,可以在云平台提供的多种存储服务中探索效率与成本的权衡.

未来将围绕以下问题继续开展研究:

- **实现更多的算子支持.**虽然本系统实现了查询和建模两种任务类型的算子,但在查询任务中,Join 和 Projection 等复杂查询算子的实现仍是难点,未来将深入研究基于无服务器计算的高效实现方案.
- **探索更高效的数据交互.**本系统选择阿里云的 OSS 对象存储服务作为共享存储空间,实现了节点之间的间接通信,这种存储服务的通信成本较低,但通信延迟较高.在未来工作中,将探索不同的存储服务(例如高速存储介质、内存数据库等)以权衡通信效率和成本.

References:

[1] Lindell Y. Secure multiparty computation for privacy preserving data mining. Encyclopedia of Data Warehousing and Mining. IGI global, 2005: 1005-1009.

[2] Cui B, Gao J, Tong YX, Xu JQ, Zhang DX, Zou L. Progress and trend in novel data management system. Ruan Jian Xue Bao/Journal of Software, 2019,30(1):164-193 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5646.htm [doi: 10.13328/j.cnki.jos.005646]

[3] Yang Q, Liu Y, Chen T, et al. Federated machine learning: concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 2019, 10(2): 1-19.

[4] Sheth A P, Larson J A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys (CSUR), 1990, 22(3): 183-236.

[5] Tang LT, Chen ZN, Zhang LF, Wu D. Research progress of privacy issues in federated learning. Ruan Jian Xue Bao/Journal of Software, 2023, 34(1): 197-229 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6411.htm [doi: 10.13328/j.cnki.jos.006411]

[6] Bogdanov D, Laur S, Willemson J. Sharemind: A framework for fast privacy-preserving computations. Computer Security-ESORICS 2008: 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings 13. Springer Berlin Heidelberg, 2008: 192-206.

[7] Bater J, Elliott G, Eggen C, et al. SMCQL: secure query processing for private data networks. Proc. VLDB Endow., 2017, 10(6): 673-684.

[8] Volgushev N, Schwarzkopf M, Getchell B, et al. Conclave: secure multi-party computation on big data. Proceedings of the Fourteenth EuroSys Conference 2019. 2019: 1-18.

[9] 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 with English abstract). http://www.jos.org.cn/1000-9825/6458.htm [doi: 10.13328/j.cnki.jos.006458]

[10] Tong Y, Pan X, Zeng Y, et al. Hu-fu: Efficient and secure spatial queries over data federation. Proceedings of the VLDB Endowment, 2022, 15(6): 1159.

[11] Zhang YY, Li SY, Shi YX, Zhou N, Xu Y, Xu K. Secure multi-party θ-join algorithm toward data federation. Ruan jian xue bao/journal of software, 2023, 34(3): 1109-1125 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6795.htm [doi: 10.13328/j.cnki.jos.006795]

[12] F. Wang, H. Zhu, R. Lu, Y. Zheng and H. Li. Achieve efficient and privacy-preserving disease risk assessment over mul ti-outsourced vertical datasets. IEEE Transactions on Dependable and Secure Computing, 2022, vol. 19, no. 3, pp. 1492-1 504, doi: 10.1109/TDSC.2020.3026631.

[13] Liang Zhu, Jiapeng Yang, Xin Song, Yu Wang, and Yonggang Wei. Real-time entity resolution by forest-based indexing in database systems with vertical fragmentations. Proceedings of the 5th International Conference on Computer Science an d Application Engineering, 2021, 67, 1–5. https://doi.org/10.1145/3487075.3487142.

[14] H. Brendan McMahan, Eider Moore, et al. Communication-efficient learning of deep networks from decentralized data. Pr oceedings of the 20 International Conference on Artificial Intelligence and Statistics, 2017,54: 1273-1282.

[15] Li T, Sahu A K, Zaheer M, et al. Federated optimization in heterogeneous networks. Proceedings of Machine learning an d systems, 2020, 2: 429-450.

[16] Li X, Jiang M, Zhang X, et al. Fedbn: Federated learning on non-iid features via local batch normalization. arXiv prepri nt arXiv:2102.07623, 2021.

[17] Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., Thorne, B. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. arXiv preprint, 2017, arXiv:1711.10677.

[18] Y. Wu, S. Cai, X. Xiao, G. Chen, B. C. Ooi. Privacy preserving vertical federated learning for tree-based models. arXiv preprint, 2020, arXiv:2008.06170.

[19] Vepakomma P, Gupta O, Swedish T, et al. Split learning for health: distributed deep learning without sharing raw patient data. arXiv preprint arXiv:1812.00564, 2018.

[20] Vafl: a method of vertical asynchronous federated learning. arXiv preprint, 2020, arXiv:2007.06081.

[21] D. Romanini, A. J. Hall, P. Papadopoulos, T. Titcombe, et al. Pyvertical: A vertical federated learning framework for multi-headed splitnn. In ICLR, Workshop on Distributed and Private Machine Learning, 2021.

[22] He C, Li S, So J, et al. Fedml: A research library and benchmark for federated machine learning. arXiv preprint 2020, a rXiv:2007.13518.

[23] Shi DY, Wang YS, Zheng PF, Tong YX. Cross-silo federated learning-to-rank. Ruan Jian Xue Bao/Journal of Software, 2 021,32(3):669-688 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6174.htm[doi: 10.13328/j.cnki.jos.006 174]

[24] Liu Y, Fan T, Chen T, et al. Fate: An industrial grade platform for collaborative learning with data protection. Journal o f Machine Learning Research, 2021, 22(226): 1-6.

[25] Chen D, Tan V J, Lu Z, et al. OpenFed: A comprehensive and versatile open-source federated learning framework. Proce edings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,. 2023: 5017-5025.

[26] Wen J, Chen Z, Jin X, et al. Rise of the planet of serverless computing: A systematic review. ACM Transactions on Sof tware Engineering and Methodology, 2023, 32(5): 1-61.

[27] AWS Lambda. Retrieved May 15, 2024 from https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

[28] Amazon S3. Retrieved on May 15, 2024 from https://aws.amazon.com/s3/.

[29] Azure Functions. Retrieved May 15, 2024 from https://docs.microsoft.com/en-us/azure/azure-functions.

[30] Google Cloud Functions. Retrieved May 15, 2024 from https://cloud.google.com/functions.

[31] Alibaba Cloud Function Compute. Retrieved May 15, 2024 https://www.aliyun.com/product/fc.

[32] OpenFaaS. Retrieved May 15, 2024 from https://www.openfaas.com.

[33] OpenLambda. Retrieved May 15, 2024 from https://github.com/open-lambda/open-lambda.

[34] Openwhisk. Retrieved May 15, 2024 from https://openwhisk.apache.org.

[35] Dong HW, Zhang C, Li GL, Feng JH. Survey on cloud-native databases. Ruan Jian Xue Bao/Journal of Software, 2024, 35(2): 899-926 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6952.htm[doi: 10.13328/j.cnki.jos.00695 2]

[36] Perron M, Castro Fernandez R, DeWitt D, et al. Starling: a scalable query engine on cloud functions. Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020: 131-141.

[37] Wang H, Niu D, Li B. Distributed machine learning with a serverless architecture. IEEE Conference on Computer Communications, 2019: 1288-1296.

[38] Carreira, Joao, et al. Cirrus: a serverless framework for end-to-end ml workflows. Proceedings of the ACM Symposium on Cloud Computing, 2019.

[39] Grafberger A, Chadha M, Jindal A, et al. Fedless: secure and scalable federated learning using serverless computing. 2021 IEEE International Conference on Big Data, 2021: 164-173.

[40] Jiang J, Gan S, Liu Y, et al. Towards demystifying serverless machine learning training. Proceedings of the 2021 International Conference on Management of Data, 2021: 857-871.

[41] Yang Z, Yang C, Han F, et al. OceanBase: a 707 million tpmC distributed relational database system. Proceedings of the VLDB Endowment, 2022, 15(12): 3385-3397.

[42] Yang Z, Xu Q, Gao S, et al. OceanBase Paetica: A Hybrid Shared-Nothing/Shared-Everything Database for Supporting Single Machine and Distributed Cluster. Proceedings of the VLDB Endowment, 2023, 16(12): 3728-3740.

[43] OpenStreetMap. 2024. https://www.openstreetmap.org.

[44] imis-3months. 2024. http://chorochronos.datastories.org/?q=content/imis-3months.

[45] Todd W. Schneider. NYC taxi trip data. 2024. https://github.com/toddwschneider/nyc-taxi-data.

[46] TPC-H. TPC-H Version 3. Accessed on September 23rd, 2022. http://www.tpc.org/tpch/.

[47] Forina M, Lanteri S, Armanino C, et al. PARVUS, an extendible package for data exploration, classification and correlation. Institute of pharmaceutical and food analysis and technologies, 2008, 16147.

[48] Mangasarian O L, Street W N, Wolberg W H. Breast cancer diagnosis and prognosis via linear programming. Operations research, 1995, 43(4): 570-577.

[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11):2278-2324.1.

附中文参考文献:

[2] 崔斌,高军,童咏昕,许建秋,张东祥,邹磊.新型数据管理系统研究进展与趋势.软件学报,2019,30(1):164−193.http://www.jos.org.cn/1000-9825/5646.htm[doi: 10.13328/j.cnki.jos.005646]

[5] 汤凌韬, 陈左宁, 张鲁飞, 吴东. 联邦学习中的隐私问题研究进展. 软件学报, 2023, 34(1):197–229. http://www.jos.org.cn/1000-9825/6411.htm [doi: 10.13328/j.cnki.jos.006411]

[9] 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 [doi: 10.13328/j.cnki.jos.006458]

[11] 张媛媛, 李书缘, 史烨轩, 周南, 徐毅, 许可. 面向数据联邦的安全多方 θ-连接算法. 软件学报, 2023, 34(3):1109–1125. http://www.jos.org.cn/1000-9825/6795.htm [doi: 10.13328/j.cnki.jos.006795]

[23] 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]

[35] 董昊文, 张超, 李国良, 冯建华. 云原生数据库综述. 软件学报, 2024, 35(2): 899－926. http://www.jos.org.cn/1000-9825/ 6952.htm[doi: 10.13328/j.cnki.jos.006952]