软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(1):164−193 [doi: 10.13328/j.cnki.jos.005646]

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

新型数据管理系统研究进展与趋势*

崔 斌[1], 高 军[1], 童咏昕[2], 许建秋[3], 张东祥[4], 邹 磊[5]

[1](北京大学 信息科学技术学院,北京 100871)

[2](软件开发环境国家重点实验室(北京航空航天大学),北京 100083)

[3](南京航空航天大学 计算机科学与技术学院,江苏 南京 211106)

[4](电子科技大学 计算机科学与工程学院,四川 成都 611731)

[5](北京大学 计算机科学技术研究所,北京 100871)

通信作者: 崔斌, E-mail: bin.cui@pku.edu.cn

摘 要: 随着各类新型计算技术和新兴应用领域的浮现,传统数据库技术面临新的挑战,正在从适用常规应用的单一处理方法逐步转为面向各类特殊应用的多种数据处理方式.分析并展望了新型数据管理系统的研究进展和趋势,涵盖分布式数据库、图数据库、流数据库、时空数据库和众包数据库等多个领域.具体而言:分布式数据管理技术是支持可扩展的海量数据处理的关键技术;以社交网络为代表的大规模图结构数据的处理需求带来了图数据库技术的发展;流数据管理技术用来应对数据动态变化的管理需求;时空数据库主要用于支持移动对象管理;对多源、异构而且劣质数据源的集成需求催生出新型的众包数据库技术.最后讨论了新型数据库管理系统的未来发展趋势.

关键词: 分布式数据库;图数据库;流数据库;时空数据库;众包数据库

中图法分类号: TP311

中文引用格式: 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

英文引用格式: 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). http://www.jos.org.cn/1000-9825/5646.htm

Progress and Trend in Novel Data Management System

CUI Bin[1], GAO Jun[1], TONG Yong-Xin[2], XU Jian-Qiu[3], ZHANG Dong-Xiang[4], ZOU Lei[5]

[1](School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

[2](State Key Laboratory of Software Development Environment (Beijing University of Aeronautics and Astronautics), Beijing 100083, China)

[3](College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)

[4](College of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China)

[5](Institute of Computer Science and Technology, Peking University, Beijing 100871, China)

Abstract: With the emergence of novel computing techiniques and applications, the traditional database manamgement systems face challenges, and undergo significant shifts from the single data model processing to multiple data model processing. This paper presents a

* 基金项目: 国家自然科学基金(61832001, 61572040, 61822201, 61622201, 61602087)
Foundation item: National Natural Science Foundation of China (61832001, 61572040, 61822201, 61622201, 61602087)
本文由“软件学科发展回顾特刊”特约编辑梅宏教授、金芝教授、郝丹副教授推荐.
本文作者同等贡献,其中,张东祥主要负责第 2 节,邹磊主要负责第 3、第 4 节,许建秋主要负责第 5 节,童咏昕主要负责第 6 节
收稿时间: 2018-07-03; 修改时间: 2018-08-21; 采用时间: 2018-09-25; jos 在线出版时间: 2018-11-22
CNKI 网络优先出版: 2018-11-23 07:17:57, http://kns.cnki.net/kcms/detail/11.2560.TP.20181123.0717.002.html

comphrensive survey on the recent progress and future direction in the novel data management systems, including distributed databases, graph databases, streaming databases, spatial-temporal databases, and crowdsourcing databases. Specifically, the distributed techinqiues play a key role to improve the scabablity of large scale data processing. Graph data management techniques are driven by the big graph management requirement in applications like social network. Stream data management techiniques are also developed to process dynamic data. Spatial-temporal databases are mainly applied in the management of mobile objects. Last but not least, the processing of multiple sources, hetergonenous and low quality data motivates the advance of crowd-sourcing techniques. This study also surveys other hot research directions and foresees the future work.

Key words: distributed databases; graph databases; stream databases; spatial-temporal databases; crowd-sourcing databases

21 世纪以来,随着计算机技术,尤其是互联网和移动计算技术的发展,大量新型应用应运而生.这些应用不仅对人类的日常生活、社会的组织结构以及生产关系形态和生产力发展水平产生了深刻的影响,也使得人们能够获取的数据规模呈爆炸性增长.“大数据”这一词汇被发明出来,用以概括这种态势.

目前广泛认为大数据具有所谓的“4V”特征,即规模大(volume)、变化快(velocity)、种类杂(variety)和价值密度低(value).为了有效地应对大数据的上述“4V”特征,各类新型数据管理系统也逐渐涌现出来.表 1 中我们列举了部分典型的新型数据管理系统.

Table 1 Category of novel data management systems

表 1 新型数据管理系统分类

系统类型	代表性系统	主要解决的问题
分布式数据管理系统	MongoDB,Redis,Cassandra,Spanner,Oceanbase	数据的规模大(volume)
流数据管理系统	STREAM,Aurora,TelegraphCQ,NiagaraCQ,Gigascope	数据的变化快(velocity)
图数据管理系统	Pregel,Giraph,PowerGraph,GraphChi,Xstream,Giraph+	数据的种类杂(variety)
时空数据管理系统	SpatialHadoop,Simb,OceanRT,DITA,SECONDO	数据的种类杂(variety)
众包数据管理系统	CrowdDB,CDB,Deco,Qurk,DOCS,gMission	数据的价值密度低(value)

数据规模大在诸多数据处理场景中都有所体现.例如社交媒体应用中的用户关系数据,若用图数据模型进行建模,其涉及的节点数可高达几亿.为了处理这类大规模的数据,一个朴素的想法是分而治之,即,将数据分布式地存储在多台机器上分别处理.据此,人们提出了各类分布式数据管理系统.

数据变化快这一特征具体体现在数据实时到达、规模庞大、大小无法提前预知,并且数据一经处理,除非进行存储,否则很难再次获取.在金融应用、网络监控、社交媒体等诸多行业领域,都会产生这类变化极快的数据.为了解决这一问题,人们提出了流数据处理系统.

针对数据种类杂的特征,人们采取“各个击破”的手段,针对各类数据分别提出专门的数据管理系统,图数据管理系统和时空数据管理系统是典型代表.图数据模型是一种具有高度概括性的数据模型,其典型应用包括社交媒体数据的建模和知识图谱等.时空数据在人们的日常生活中也十分常见,例如各类地图应用在提供导航服务时,都需要对大量的时空数据进行高效的处理.

大数据的价值密度通常较低,例如社交媒体中大量的图片数据在未经标注之前,并不具备显著的价值.众包正是解决该问题的有效手段之一.众包通常是指“一种把过去由专职员工执行的工作任务通过公开的 Web 平台以自愿的形式外包给非特定的解决方案提供者群体来完成的分布式问题求解模式”,是完成大规模的对计算机较为困难而对人类相对容易的任务的有效手段,例如数据标注.为了有效地对众包过程中的数据和众包参与者群体进行有效管理,人们提出了众包数据管理系统.

如表 1 所示,我们可以看到:人们已经提出了分布式数据管理系统、图数据管理系统、流数据管理系统、时空数据管理系统和众包数据管理系统以应对大数据的“4V”特征带来的挑战,并设计了大量代表性数据管理系统,在实际应用中已经取得了较好的效果.下面我们将针对不同系统类型分别阐述上述技术及相关系统的进展,并展望后续发展趋势.本文最后对新型数据管理系统在数据模型、计算模型和体系结构等方面的挑战和机遇进行了探讨.

1 分布式数据库

1.1 引 言

在大数据时代下,移动互联网、智能设备以及物联网技术的发展,使得全球数据量呈现爆发式增长,远远超出传统的单机版数据库的处理能力.近几年,分布式数据库一直是工业界和学术界的研究重点.分布式数据库应该具备强一致性、高可用性、可扩展性、易运维、容错容灾以及满足 ACID 属性的高并发事务处理能力.但在实际设计中,一方面受限于 CAP 理论[1],即,在必须支持分区容错性(partition tolerance)的前提下,系统实现只能侧重一致性(consistency)和可用性(availability)的一个方面而无法同时满足;另一方面,支持 ACID 事务属性及高并发事务处理一直是分布式关系数据库的难点.针对这些挑战,现有的解决策略大致可分成 3 类:(1) 将现有商业关系数据库(如 Oracle、SQLServer、MySQL、PostgreSQL)在分布式集群或者云平台上进行小规模扩展和部署;(2) 放弃关系数据库模型和 ACID 的事务特性,选择灵活的 schema-free 数据模型及高可用性和最终一致性的 NoSQL 数据库;(3) 融合关系数据库和 NoSQL 优势的新型数据库(NewSQL).

1.2 主要研究问题

主流的分布式数据库基本上围绕数据强一致性、系统高可用性和 ACID 事务支持等核心问题展开研究工作,这些性质与系统的扩展性和性能密切相关,甚至相互制约,往往需要根据具体的应用需求进行取舍.

- 数据强一致性:银行交易系统等金融领域往往有数据强一致性和零丢失的需求.当更新操作完成之后,任何多个后续进程或线程的访问都要求返回最近更新值.如果在这个分布式系统中没有数据副本,那么系统必然满足数据强一致性要求,原因是只有独本数据,才不会出现数据不一致的问题.但是分布式数据库系统的设计需要保存多个副本来提高可用性和容错性,以避免宕机时数据还没有复制,导致提供的数据不够准确.如何低成本地保证数据的强一致性,是分布式数据库系统的一个重要难题;
- 系统高可用性:在分布式数据库中,系统的高可用性和数据强一致性往往不可兼得.当存在不超过 1 台机器发生故障的时候,要求至少能读到一份有效的数据,往往需要牺牲数据的强一致性来保证系统的高可用性.相当一部分 NoSQL 数据库采用这个思路来支持互联网场景下的大规模用户并发访问请求,它们通过实现最终一致性来确保高可用性和分区容忍性,弱化了数据的强一致要求.为了解决数据不一致问题,不同的分布式数据库设计各自的冲突机制.另外,有效的容错容灾机制也是保障系统高可用性的坚实后盾;
- ACID 事务支持:ACID 指的是事务层面的原子性(atomicity)、一致性(consistency)、隔离性(isolation)和持久性(durability).如何有效地支持 ACID 事务属性,一直是分布式数据库的难点,涉及到很多复杂的操作和逻辑,会严重影响系统的性能,很多 NoSQL 数据库都是放弃支持事务 ACID 属性来换取性能的提升.近年来,新型数据库(NewSQL)的出现给分布式数据库的发展带来新的方向,它的目标是提供与 NoSQL 相同的可扩展性和性能,同时支持事务的 ACID 属性.这种融合一致性和可用性的 NewSQL 已经成为分布式数据库的研究热点.

1.3 国内外研究现状

1.3.1 基于分布式集群或云平台的关系数据库

MySQL 集群是一种常见的开源分布式数据库,它基于无共享的(shared-nothing)数据存储模式,采用读写分离的主从模式(master-slave)来实现高可用性.该设计方法也被主流的云平台关系数据库采纳和应用,包括亚马逊的 Amazon RDS(Amazon relational database service)[2]、谷歌的 Google Cloud SQL[3]、微软的 Azure SQL Database[4]以及国内的阿里云 RDS[5]、腾讯 CDB(cloud DataBase)[6]和网易的蜂巢 RDS[7].与传统数据库相比,这些云数据库往往同时支持 MySQL、SQL Server 及 PostgreSQL 等数据库引擎,具有低成本、易运维、可伸缩、高可用等优势,并提供容灾、备份、恢复、监控、迁移等数据库运维全套解决方案.

基于分布式集群或云平台关系数据库的主要缺陷是很难低成本地保持数据的一致性,每次将数据写到

master 节点后,都要及时同步 slave 节点,所以往往牺牲性能来保障强一致性.另外,如果 master 节点宕机,会直接导致业务不可写,也会影响整个系统的高可用性.为解决这一问题,MySQL 集群自带的数据同步策略从最初的异步复制进化为MySQL 5.7版本的半同步复制,但效果依旧有限.各大企业也纷纷设计MySQL补丁来保证数据一致.Amazon Aurora将事务引擎和存储引擎分离,redo日志从事务引擎中剥离,归并到存储引擎中,属于典型的shared disk架构,通过存储层共享来解决一致性问题.Galera Cluster采用多主架构(multi-master)来实现真正的多点读写,即集群中每个节点都可读写,无需读写分离.集群不同节点之间数据同步是基于 Galera replication 中间件,避免了 MySQL 集群主从节点之间的复制延迟.

国内的云关系数据库也对数据一致性的提升做出了原创性贡献.阿里巴巴的 AliSQL 利用了分布式一致性协议(Raft)[8]以保障多节点状态切换的可靠性和原子性.为了保证多节点之间的 binlog 的强一致性,腾讯的PhxSQL 使用分布式一致性协议 Paxos 实现 master 管理.同时,用 BinlogSvr 来支持 MySQL 的异步复制协议以支撑微信后台的账号系统、企业微信及 QQ 邮箱等.网易 RDS 则是采用虚拟同步复制技术,确保所有主机的更新事务在提交前都首先在从机上落盘,保证主从切换后数据完全一致.在节点上使用了并行复制技术,大幅度提高了从机回放主机事务的速度,复制延迟消失,保证了在秒级完成主从切换.

1.3.2 NoSQL 数据库

由于事务处理过程对 ACID 属性的严格要求,云关系数据库的可扩展性相对有限.为提升系统存储和处理海量数据的能力,NoSQL 从底层数据模型进行考虑,放弃关系模型,也不保证支持 ACID 事务处理.它采用schema-free 的数据模型,可以根据不同应用需求衍生出多种类型的分布式数据库.按照存储模型来分类,可将NoSQL 数据库分为列式存储(如 HBase[9]和 Cassandra[10])、键值存储(如 Redis[11]、MemcacheDB[12]、DynamoDB[13]、SimpleDB[14])和文档存储(如 MongoDB[15]和 CouchDB[16]).

一个分布式系统最多同时满足一致性(consistency,简称 C)、可用性(availability,简称 A)和分区容错性(partition tolerance,简称 P)中的两项,可根据相应的设计目的进行选择.对 NoSQL 而言,分区容错性是不能牺牲的,因此只能在一致性和可用性上加以取舍.如果在这个分布式系统中数据没有副本,那么系统必然满足强一致性条件,因为只有独本数据,不会出现数据不一致的问题,此时 C 和 P 都具备.但是,如果某些服务器宕机,那必然会导致某些数据是不能访问的,那 A 就不符合了;反之,如果在这个分布式系统中数据是有副本的,那么如果某些服务器宕机,系统还是可以提供服务的,即符合 A,但是很难保证数据的一致性.因为宕机时可能有些数据还没有复制到副本中,那么提供的数据就不准确了.一般情况下,对于一致性要求比较高的业务在访问延迟时间方面就会降低要求,适合选择 CP 模式;对于访问延时有高要求的业务在数据一致性方面会降低要求,适合选择 AP 模式.

- CP 模式

CP 模式要求分区容忍性,同时对数据一致性要求较高,即,能够保证所有用户看到相同的数据.当网络通信出现问题时,暂时隔离开的子系统可继续运行(分区容忍性),但是不保证某些节点故障时,所有请求都能被响应.如果主节点宕机,可能需要选举新的主节点,同步节点间数据以及进行数据回滚等操作,这会使系统的可用性降低.常见的 CP 模式系统有 BigTable[17]、HBase[9]、Redis[11]、MongoDB[15]和 MemcacheDB[12].

BigTable是一个GFS[18]之上的索引层,采用两级的B+树索引结构.GFS保证至少写入一次成功的记录并由BigTable记录索引.为了保证BigTable的强一致性,同一时刻同一份数据只能被一台机器服务,且BigTable中的Tablet Server没有对每个Tablet备份.HBase是Apache Hadoop中的一个子项目,属于BigTable的开源版本,是满足强一致性的分布式数据库,主要用来存储非结构化和半结构化的松散数据.HBase利用Hadoop HDFS[19]作为其文件存储系统,借助 MapReduce[20]计算模型来处理海量数据,并使用 Zookeeper 作为分布式协同服务.HBase使用了事务机制中常见的一致性实现方式WAL(write-ahead logging)[21].

Redis集群通常是主备,主节点负责写入和读取,而slave节点只是用来备份.当主节点失败时,slave节点有机会被提升为主节点.MongoDB采用类似于Redis的主从集群方式,主节点作为单点写操作服务,然后同步到slave节点,可以通过设置 autoresync 发现 slave 节点的数据不是最新,则自动地从主服务器请求同步数据.MongoDB

使用基于 Raft 协议选主策略,一旦主节点发生故障,整个 MongoDB 会进行交流,然后选择一个合适的 slave 节点快速实现故障恢复.

MemcacheDB 是一个新浪网基于 Memcached 开发的分布式 Key-Value 存储持久化开源项目,它使用 BerkeleyDB[22]作为存储引擎,通过为 Memcached 增加 Berkeley DB 的持久化存储机制和异步主辅复制机制,使 Memcached 具备了事务恢复能力、持久化能力和分布式复制能力,非常适合超高性能读写速度和持久化保存的应用场景.

- AP 模式

AP 模式主要以实现最终一致性来确保可用性和分区容忍性,但却弱化了对数据的一致要求,大部分 NoSQL 系统都属于 AP 模式范畴.

Amazon Dynamo[13]是一个分布式键值存储系统,它采用去中心化、松散耦合方式,由数百个服务组成面向服务架构,不支持复杂的查询.Dynamo 利用一致性哈希来完成数据分区,给系统中的每个节点随机分配一个 token,这些 token 构成一个哈希环.执行数据存放操作时,首先计算 key 的哈希值,然后存放到顺时针方向第 1 个大于等于该哈希值的 token 节点上.这种算法的优点是:节点的增删只会影响哈希环中相邻的节点,对其他节点没有影响.

Cassandra[10]由 Facebook 开发并于 2008 年开源,系统架构与 Dynamo 一致,均为基于 DHT(分布式哈希表)的完全 P2P 架构,具有高度可扩展性和高度可用性,没有单点故障.Cassandra 使用由 Dynamo 引入的架构特性来支持 BigTable 数据模型,并采用 MemTable 和 SSTable 方式进行存储.在 Cassandra 写入数据之前,需要先记录日志(CommitLog),再将数据开始写入 ColumnFamily 对应的 MemTable 中.

Amazon SimpleDB[14]是一个可大规模伸缩、用 Erlang 编写的高可用数据存储,类似于 Amazon S3,但两者的一致性有很大的区别.Amazon S3[23]一致性模型为弱一致性,只支持最基本的 Put/Get/Delete 操作,且每个操作之间是互相独立的.SimpleDB 除了支持 Get/Put/Delete,还需要支持强一致读以及基于条件更新或者删除的乐观锁机制.Amazon S3 直接使用“Last Write Wins”的方式解决冲突,因为集群内部机器时钟不一致的概率很低.另外,发生同一条记录被多个客户端更新且同时发生机器故障等异常情况的概率也很低.SimpleDB 需要支持一些条件更新或者删除,从而支持乐观锁机制以实现最终一致性.

Apache CouchDB[16]是一个面向文档数据管理的开源数据库,使用 JSON 存储半结构化的数据,查询语言为 JavaScript 并封装 MapReduce 和 HTTP 作为 API,适合 CMS、电话本、地址本等的应用.其最显著的特性是支持多主复制,没有锁机制,通过使用 MVCC(多版本并发性控制)[24]实现最终一致性.Tokyo Cabinet[25]的开发者是日本人 Mikio Hirabayashi,主要被用在日本最大的 SNS 网站 mixi.jp 上,也曾是键值数据库领域的热点.其他的高可用性 NoSQL 数据库还有 Voldemort[26]、Riak[27]等(见表 2).

Table 2 Comparison of representative NoSQL systems

表 2 代表性 NoSQL 数据库比较

	MongoDB	HBase	Cassandra	Redis
数据模型	Document	Column	Column	Key-Value
存储模型	普通存储	HDFS	普通存储	内存
数据一致性	最终一致性保证	强一致性保证	最终一致性保证	无一致性保证
数据压缩	支持	支持	支持	不支持
事务支持	不支持	不支持	不支持	不支持

1.3.3 NewSQL 数据库

近年来,以 Spanner[28]为代表的新型数据库(NewSQL)的出现,给数据存储和分析带来了 SQL、NoSQL 之外的新思路.NewSQL 指的是提供与 NoSQL 相同的可扩展性和性能,并同时能支持满足 ACID 特性的事务.这保留了 NOSQL 的高可扩展和高性能,且支持关系模型.融合一致性和可用性的 NewSQL 可能是未来大数据存储新的发展方向.

Spanner 是第一个将数据分布到全球规模的系统,并且在外部支持一致的分布式事务,设计目标是横跨全球

上百个数据中心,覆盖百万台服务器.不同于 BigTable 版本控制的键值存储模型,Spanner 演化为时间上的多维数据库.旧版本数据根据可配置的垃圾回收政策处理,应用可以读取具有旧时间戳的数据.Spanner 显著的特点是外部一致读写和在某一时间戳的全度跨数据库一致读取.对于最典型的读写事务,Spanner 使用常见的两阶段锁策略(2PL)来控制并发,并实现了一个所谓的外部一致性.F1[29]是 Google 公司提出的建立在 Spanner 基础上的用于广告业务的存储系统.F1 实现了丰富的关系型数据库的特点,包括严格遵从的 schema、强力的并行 SQL 查询引擎、通用事务、变更与通知的追踪和索引.其存储被动态分区,数据中心间的一致性复制能够处理数据中心崩溃引起的数据丢失.Google F1 提供了一种可能性:OLTP 与 OLAP 融合的可能性,这是在其他数据库中从未没有实现过的.

国内数据库在 NewSQL 领域的代表性系统包括阿里巴巴的 OceanBase[30]和腾讯的 DCDB[31].OceanBase 是支持海量数据的高性能分布式数据库系统,实现了数千亿条记录和数百 PB 数据的跨行跨表事务.数据多副本通过 Paxos 协议同步事务日志,多数派成功事务才能提交.缺省情况下,读、写操作都在主副本进行,保证强一致.存储采用读写分离架构,没有主从结构,集群节点全对等,每个节点都具备计算和存储能力,无单点瓶颈.腾讯 DCDB 又名 TDSQL,是一种兼容 MySQL 协议和语法且支持自动水平拆分的高性能分布式数据库,即:业务显示为完整的逻辑表,数据却均匀地拆分到多个分片中.每个分片默认采用主备架构,提供灾备、恢复、监控、不停机扩容等全套解决方案,适用于 TB 或 PB 级的海量数据场景.这几年,TDSQL 不断进步,研发了很多新特性,诸如多级分区、热点更新、隐含主键、分布式事务等,不仅有力地支撑了事务型数据库应用,而且在体系结构上也朝 Spanner 架构上迈进,是一个名副其实的 NewSQL 系统.

TiDB[32]作为 NewSQL 开源社区的代表,是 PingCAP 公司基于 Google Spanner/F1 论文实现的开源分布式 NewSQL 数据库,能够实现分布式事务以及跨数据中心数据强一致性保证.TiDB 最底层用 Raft 来同步数据.每次写入都要写入多数副本,才能对外返回成功,即使丢掉少数副本,也能保证系统中还有最新的数据.TiDB 的事务模型采用乐观锁,只有在真正提交时才会做冲突检测,如果有冲突,则需要重试.由于分布式事务要做两阶段提交,并且底层还需要做 Raft 复制,一个非常大的事务会使得提交过程非常慢,并且会卡住下面的 Raft 复制流程,所以在设计上,TiDB 和 Spanner 对事务的大小进行了限制.

1.4 总结与展望

本节从 CAP 理论出发,对分布式数据库的数据一致性、系统可用性和 ACID 事务属性进行了综述,并针对国内外数据库的发展状况,介绍了现有商业关系数据库如何在云平台上进行扩展和部署,NoSQL 数据库如何支持 schema-free 数据模型、高可用性和最终一致性,以及新型数据库(NewSQL)如何提供与 NoSQL 相同的可扩展性和性能,同时能够支持满足 ACID 特性的事务.

在大数据环境下,NoSQL 分布式数据库与传统分布式数据库的最终目标都是对用户提供完善的数据存储和查询功能,并且在运营上能够实现可伸缩和高可用等特性,并提供容灾、备份、恢复、监控等功能.两者最大的区别在于传统分布式数据库追求数据强一致性,并且需要提供 ACID 事务支持,导致其在峰值性能、伸缩性、容错性、可扩展性等方面的表现不尽如人意,很难满足海量数据的柔性管理需求.NoSQL 则是以牺牲支持 ACID 为代价,换取更好的可扩展性和可用性.NewSQL 是一种相对较新的形式,旨在将 SQL 的 ACID 保证与 NoSQL 的可扩展性和高性能相结合.

未来几年,融合关系数据库和 NoSQL 优势的 NewSQL 将继续在分布式数据库领域大放光彩,并成为一个重要的研究热点.以 OceanBase 和 DCDB 为代表的国内 NewSQL 系统也将在海量复杂业务的推动下持续发展和优化,并作为国家大数据发展战略提供有力支撑.这也意味着,我国有可能在下一波数据库技术潮流当中占领先机,进入第一梯队.

2 图数据库

2.1 引 言

近年来,随着社交网络与语义网的发展,基于互联网的图数据规模越来越大.截止到 2017 年底,微信已经有了将近 10 亿的活跃用户,这些用户相互关联与通信,仅在 2016 年春节期间,用户之间就互相分发了 32 亿个红包[33].在语义网的 Linked Open Data 项目中,已有超过 1 184 个 RDF 图数据集,合计超过 800 亿条边[34].针对这些规模巨大的图数据,设计与实现高效的图数据管理系统成为一个很重要的研究热点.

现阶段,工业界和学术界已经设计并实现了不少大规模图数据管理系统.按照对图数据管理的抽象程度,可以将其分成如下两类.

- 低层次抽象的提供编程接口的图数据管理系统:这类系统会针对图数据管理中的基本操作设计并实现相应的编程接口,用户利用这些编程接口来实现相应的管理功能;
- 高层次抽象的提供描述性查询语言的图数据管理系统:这类系统设计图数据管理描述性查询语言,用户将相应的管理需求用描述性查询语言表达,系统解析这些描述性查询语句并生成相应的查询计划以进行执行处理.

表 3 中,我们列举了本文即将介绍的图数据管理系统以及它们的分类.

Table 3 Category of graph data management systems

表 3 图数据管理系统分类

系统类型	代表性系统	编程模型	描述性查询语言
低层次抽象的提供编程接口的图数据管理系统	Pregel,Giraph,PowerGraph,GraphLab,Quegel,PAGE	点计算	×
	Trinity	基于内存云的键值对	×
	GraphX	图并行计算/数据并行计算	×
高层次抽象的提供描述性查询语言的图数据管理系统	Neo4j	×	Cypher
	EmptyHeaded	×	自定义查询语言
	gStore	×	SPARQL

2.2 主要研究问题

针对大规模图数据处理,主要有以下几个常见问题.

- 图搜索:给定一个图,从一个点出发沿着边搜索其他所有节点.常见的图搜素方法有宽度优先、深度优先和最短路径等.图搜索是图计算问题的基础,用于衡量图计算的 Benchmark Graph500[35]就是以宽度优先搜索性能作为评测机器的图计算能力的标准;
- 基于图的社区发现:社区发现是社交网络分析中一个重要的任务,用于分析网络图中的密集子图.这对于理解社交网络中的用户行为和朋友推荐等都具有非常重要的应用价值,典型的社区发现算法有 *K*-core[36]、*K*-truss[37]以及 *K*-clique[38];
- 图节点的重要性和相关性分析:计算图中某个节点的重要程度,例如在网页链接图中分析网页的重要程度,代表性工作是 Pagerank[39];衡量图上两个节点的相关性,例如社交网络中两个人之间的关系,代表工作包括 SimRank[40]和 Random Walk[41]等;
- 图匹配查询:给定数据图和查询图,图匹配查询找出所有在数据图上与查询图同构的子图,这个问题常用于描述针对图结构的查询.图匹配查询的应用包括化学分子库中的分子拓扑结构查询[42]、在一个社交网络图中的特定社交结构查询[43]等.面向 RDF 知识图谱数据的 SPARQL 查询语言就是基于子图匹配的查询语义[44,45].

2.3 国内外研究现状

2.3.1 低层次抽象的提供编程接口的图数据管理系统

提供低层次抽象编程接口的图数据管理系统包括 Pregel[46]及其衍生[47−51]、Trinity[52]、GraphX[53]等;系统

会将常见的图运算中的基本操作抽象成编程接口.虽然这类系统屏蔽了包括图数据的内部数据结构表示、分布式环境下的通信处理等底层系统问题,但是由于是低层次抽象的编程接口,用户还需要将具体的图计算任务转换成系统提供的低层次抽象编程接口逻辑.

这类系统中典型的是“点计算模型”,即允许用户定义每个点的计算任务.最早的系统是谷歌提出的一种分布式图数据管理系统 Pregel[46],该系统基于 BSP 分布式计算模型[54]进行设计,图数据的每个点为基本计算核心,机器之间通过消息传递来实现同步.Pregel 将图运算用一系列的超级计算步(superstep)来描述,在运算每一次超级计算步时,每一个顶点都能接收来自上一次超级计算步的信息,然后将这些信息传送给下一个顶点,并在此过程中修改其自身的状态信息(例如以该顶点为起点的出边状态信息)或改变整个图的拓扑结构.Pregel 系统将不同机器的通信进行了封装,用户需要通过编程来描述点计算函数进而实现自身的需求.

Pregel 的基于 BSP 的点计算模型得到了工业界以及学术界的广泛关注与认可.很多人在 Pregel 上开发了效率更高的系统,包括 Giraph[47]、PowerGraph[48]、GraphLab[49]、Quegel[50]、PAGE[51]等.

除了基于“点计算”的图计算系统,Trinity[52]是微软研发的一个基于内存的分布式图数据管理系统.Trinity 认为:随着时代的发展,一方面内存越来越大且越来越便宜;另一方面,图数据上的基本操作非常复杂,将数据存储在外存会导致操作不便,所以利用内存进行图数据管理才是更好的选择.因为单机内存规模总是有限的,所以 Trinity 使用了内存云的技术,也就是将多台机器的内存封装起来,使得用户能够同时使用多台机器的内存,而且无需知道底层细节.Trinity 的基本数据结构是键值对,可以通过邻接表的形式存储与管理数据图,用户通过编程调用内存中的邻接表来实现自身需求.

GraphX[53]将图计算任务分成两种:图并行计算任务(graph-parallel computation)和数据并行计算任务(data-parallel computation).所谓图并行计算任务,主要是指基于 BSP 点计算模型来实现的迭代计算任务,如 PageRank;所谓数据并行计算任务,主要是指图上代数运算,如构建一个图、合并两个图、跨越多个图等等.GraphX 的作者认为:现有的图计算系统(如 Pregel[46]、GraphLab[49])通过限定编程框架的形式来提高图并行计算任务的执行效率,但是这些系统并不适合数据并行计算任务.基于上述观察,GraphX 在分布式计算平台 Spark[55]的基础上构建了 GraphX,以同时处理图并行计算任务和数据并行计算任务.GraphX 以图为第 1 类组成对象,以属性图为数据模型.所谓属性图,就是每个点和每条边都可以关联一个属性值表.GraphX 定义了很多图上的操作.既包括一些图并行计算任务,如 Pregel 等,也包括一些数据并行计算任务,如 map、filter 等.

2.3.2 高层次抽象的提供描述性语言的图数据管理系统

所谓高层次抽象的提供描述性查询语言的图数据管理系统包括 Neo4J[56]、EmptyHeaded[57,58]、gStore[44,45,59]等.这些系统为了方便用户对图数据的使用,在构建图数据管理系统的基础上,设计或者采用了一些描述性查询管理语言.用户可以将自身的需求表达成描述性语句,然后系统将这些任务语句解析成执行计划,最后由系统按照执行计划进行处理进而得到计算结果.因为这类系统用描述性查询语言作为用户和系统的交互中介,所以这类系统具有较好的用户友好性.

Neo4J[56]是一个由美国 Neo Technology 公司开发的基于 Java 平台的开源图数据管理系统,具有如下 4 个特点:支持满足 ACID 特性的事务操作、很好的可用性、很高的可扩展性、支持高效率遍历查询.Neo4J 的描述性查询语言是 Cypher[60],适合于开发者和在数据库上进行查询的数据专业操作人员.针对实际中各种应用需求,Cyper 定义不同的方法来描述与表达.Cyper 的许多关键字受 SQL 的启发,如 like 和 order by,它是一个申明式的语言,焦点在如何从图中找回(what to retrieve),而不是怎么去做.在不公布实现细节的前提下,用户关心如何查询优化.

EmptyHeaded[57,58]是由斯坦福大学开发的图数据管理系统,这个系统首先将图上的计算任务转化成边的连接操作,然后利用现有关系数据库关于多路连接的最新研究成果[61]找出最优的多路连接查询执行计划.在查询执行阶段,EmptyHeaded 利用 SIMD 技术来提高查询执行效率.EmptyHeaded 提出了自己的描述性查询语言,主要整合了联合查询、聚集操作和迭代运算,支持常见的子图匹配、PageRank 计算、最短路径计算等.

着语义网的发展,越来越多的数据被表示成 RDF(resource description framework,即资源描述框架)[62]形式

发布到网络上.在RDF模型下,网络资源及其关系也可以被表示成一个图,方便用户利用图技术进行数据表示与管理.针对 RDF 数据,已经有推荐的描述性结构化查询语言 SPARQL(simple protocol and RDF query language)[63],可以实现大规模 RDF 的数据管理.

针对 RDF 知识图谱的数据管理,可以采用基于关系数据库的方法,也可以采用图计算的策略.利用 RDF 知识图谱的图结构特性来管理数据,代表性系统有基于图的 RDF 知识图谱数据管理系统 gStore[44,45,59]和支持自然语言问句的 RDF 知识图谱检索系统 gAnswer[64].

2.4 总结与展望

本文分类阐述了两类图数据管理方法,对图数据计算任务进行了不同程度的抽象,进而提出了不同的交互方式.研究人员也提出了一些新的研究问题,包括在异构计算环境下的图数据管理问题,例如,如何利用新型高性能计算芯片 FPGA 进行图数据处理.异构计算环境下的图数据计算问题是一个开放性研究课题,同时,在传感器、社交网络等环境下的图数据管理问题具有多源且实时更新的特点,面对多源的流式图数据管理也是图数据管理所面临的新的挑战.

3 流数据管理

3.1 引 言

智能手机的普及和移动互联网的发展极大地加速了数据的生成过程,令数据呈现出爆炸式的增长,并给大数据的实时管理带来了前所未见的难题和挑战.例如,微信的月活跃用户已超过了 10 亿,用户之间的交互则会带来更大规模的数据,包括语音、视频、图片以及相关的文本等.数据的规模和复杂性还在高速增长,如社交网络每天以亿级别的发文[65]、轨道交通应用形成的大规模定位与轨迹信息以及网络通信中的数据传播等.为了处理实时增长的大规模复杂数据,流数据的管理和相关系统的研究一直是学术界和工业界的热点问题,包括早期的关系型数据为主的数据流管理系统、近期在工业界普遍使用的流式计算系统以及目前广泛关注的对图数据流管理系统的探索.

3.2 主要研究问题

流数据有众多不同的定义,但统一起来可以用随时间不断增长的数据模型来概括.除了基本的数据查询统计等操作外,主要有 3 方面的研究问题——流数据采样、持续性数据查询和流数据并行计算.

- 流数据采样.基于有限的存储来管理无限的动态数据是流数据管理中的基本挑战之一,应对这一挑战的最经典的思路则在于流数据上的高效采样.将高速更新的流数据采样到有明确规模边界的有限存储中,通过对采样数据的计算和挖掘来反映流数据所蕴含的重要信息.一方面,需要研究不同流数据场景下采样策略的选取,进而能够利用有限的资源尽可能地反映原流数据的特征信息;另一方面,需要结合计算需求,精准分析采样数据上的计算与挖掘结果相对于精确解的近似程度,控制计算结果的偏移范围;
- 持续性数据查询.流数据模型所对应的最核心的现实场景是实时监控.对不断生成的现实数据进行高效的计算挖掘,能够及时获取现实世界中的重要信息.例如银行对实时的交易数据进行监控,及时规避欺诈风险和追踪洗钱等违法行为.因此,给定基于结构特征、统计特征的数据查询模式,实时地监控流数据中匹配的目标,一直都是研究的热点.一方面,需要保存已计算的中间结果来减少重复性的计算,另一方面,又需要避免中间结果维护带来过高的额外开销;
- 流数据并行计算.应对流数据高速生成的一个重要策略就是利用数据和计算的独立性进行并行处理,提高系统吞吐量.系统日志数据、银行流水数据以及大量的移动应用产生的用户数据等在其初期的归整处理上都可以利用数据独立性进行流水线式的并行处理.在更复杂的数据计算和分析过程中,针对计算独立性和流场景的一致性要求,设计锁机制来实现计算分析的并行化.

3.3 国内外研究现状

本节将通过流数据管理研究的 3 个不同阶段分别阐述流数据管理系统目前的研究脉络:首先,简单了解早期以关系型数据为主的数据流管理系统(DSMS);然后,详细介绍近期针对大规模复杂数据的流式计算系统;最后讨论目前兴起的对图数据流管理系统的探索.

3.3.1 数据流管理系统

数据流管理系统(DSMS)是指管理持续性数据流的计算机软件.不同于传统的数据库管理系统(DBMS),数据流管理系统支持持续性查询,每个查询从注册开始有效直至撤销结束,不仅仅执行一次.在有效期间内,随着数据流的不断更新,持续性查询的结果也会更新.传统的数据库管理系统一般假定有足够的外存用来持久化数据,并且可以进行随机访问.而数据流管理系统中主要强调用有限的内存来处理无限的数据流,并且只能顺序访问,这也是数据流管理最独特的特征以及最大的挑战.

目前,数据流管理系统并没有统一的系统框架,但在查询语言上,绝大多数都采用类似于传统数据库管理系统中 SQL 的声明式语言来表达查询,包括持续性查询语言(continuous query language,简称 CQL)[66]、流 SQL[67]等.流查询语言也会支持窗口表达.在图示化的查询语言中,每个细分的查询由一个“查询盒”表达,各个细分查询的关联与组合通过“查询盒”[68]间的箭头连线表达.这个结构可以理解为流计算框架(诸如 Storm 等)中数据处理与传输架构的前身.

目前,主要的数据流管理系统有 STREAM[69]、Aurora[68]、TelegraphCQ[70]、NiagaraCQ[71]以及 Gigascope[72]等.STREAM 是斯坦福大学研发的基于关系模型的多功能数据流管理系统,它聚焦在数据流计算时的内存管理以及近似查询.Aurora 是一个以工作流为导向的数据流管理系统,用户可以通过“查询盒”来定义查询计划,每个“查询盒”含有基础的操作命令,“查询盒”之间的数据流指向决定了各个步骤结果的传输框架.TelegraphCQ 是一个由伯克利大学开发的自适应数据流管理系统,用于支持不同场景的数据流应用.NiagaraCQ 是针对动态 Web 内容进行持续性 XML-QL 查询的数据流管理引擎(XML-QL 是 XML 查询语言的一种扩展,用来支持大 XML 文档上的数据抽取,能够跨多个不同的 DTD 解释 XML 数据以及集成多个不同源的 XML 数据).它对 XML 数据的抽取、查询与监控分别由 3 个主要组件来支撑:搜索引擎、查询引擎以及触发管理.Gigascope 是面向网络数据流监控的分布式数据流管理系统,可以用来支撑网络流量分析、入侵检测、路由配置分析等,也能够进行网络搜索、性能监控等.

表 4 给出了数据流管理系统的对比情况.这些系统的核心初衷在于对静态数据管理系统的流模型扩展,因此,这些系统在查询语义和执行计算的数据处理逻辑方面与传统的数据管理模型有很大的重叠,可以认为是在传统数据管理系统的语义和架构上的扩展,以支撑数据流场景的持续性查询.目前,大规模高速生成的数据结构复杂,基于关系模型的数据流管理系统难以应对这种大数据场景.

Table 4 Comparisons of data stream flow management systems

表 4 数据流管理系统对比

数据流管理系统	查询语言	数据类型	架构	时间窗口	持续性查询
STREAM	CQL	关系元组	单机	✓	✓
Aurora	SQuAl	关系元组	单机	✓	×
TelegraphCQ	StreaQuel	关系元组	单机	✓	✓
NiagaraCQ	XML-QL	XML 对象	单机	×	×
Gigascope	GSQL	关系元组	分布式	✓	✓

3.3.2 流计算框架

流计算系统是目前学术界和工业界广泛使用的进行大规模数据处理的计算系统.目前,主流的流计算框架主要有 5 个:Storm[73,74]、Spark Stream[75]、Samza[76,77]、Flink[78,79]以及 Kafka Stream[80].流计算框架具有两个重要概念:交付保证(delivery guarantee)[81]、流处理类型中的实时处理流和微批量处理流.

交付保证是系统在处理新来的数据项时提供相应层次的处理保障,分为 3 种:第 1 种是“至少 1 次”,也就是

说,即便出现系统宕机等错误,也仍然能够保证新来的每个数据项被处理 1 次,可能会出现多次重复处理;第 2 种是“最多 1 次”,也就是说,新来的数据最多会被处理 1 次,在宕机等错误情况发生时,有可能数据会被丢弃而导致没有被处理;第 3 种则是“恰好 1 次”,也就是要求最严格的交付保证,确保无论发生什么情况,新来的数据项有且仅有 1 次处理.

流处理类型[82]的实时处理流是指每个新来的数据项都会被立即处理而无需等待后续的数据项,代表流框架有 Storm、Samza、Flink 以及 Kafka Stream;微批量处理流是指数据项并不是到达之后立即处理,而是等待一段很短的时间使其聚成一定单位大小的单个小批量数据后再处理,对应的流框架有 Spark Stream 以及 Storm-Trident(Storm 的一个扩展,一个以实时计算为目标的基于 Storm 的高度抽象.它在提供处理大吞吐量数据能力(每秒百万次消息)的同时,也提供了低延时分布式查询和有状态流式处理的能力).

Storm 是一个 Twitter 开源流系统,也是最早出现的开源流式计算框架.在初始化时,需要用户定义一个实时计算框架,其结构是一个有向图.图中的点是集群中的计算节点,而边则对应整体计算逻辑中数据的传输,这个图框架也被称为拓扑.在一个拓扑中,传输的数据单元是一系列不可修改的键值对(tuple),键值对从 spout(消息源)点中输出形成流数据并传输到 bolts(消息处理者)点中进行计算,进而产生出新的输出流.bolt 输出流也可以传输给其他 bolts 节点,形成流水线式的计算处理流.Storm 也有不足之处,它并不支持状态管理以及窗口、聚集等操作,支持的交付保证为“至少一次”而不是高要求的“恰好一次”.Storm 的容错机制是 Ack 机制,通信过程中需要的额外开销可能会影响流计算的吞吐量.

Spark Stream(又称为 Structured Stream)其实是 Spark[83,84]核心 API 的一个扩展,其对流式处理的支持其实是将流数据分割成离散的多个小批量的 RDD 数据(RDD 是 Spark 的数据单元),然后再进行处理.这些小批量的数据被称为 DStream(D 为 Discretized,即离散化的意思).Spark Stream 采用的是 Lambda[85,86]架构,即同时运行批量处理和实时流处理的架构,其中,批量处理用来确保计算的正确性,而实时流处理则是为提高吞吐量.当实时流处理计算结果与批量处理的计算结果不一致时,则会校正错误,因为有批处理的存在,所以自然而然就实现了容错机制. Spark Stream 支持的是“恰好一次”的交付保证,而且与 Storm 一样,Spark Stream 并没有状态管理.

Samza 系统处理的流数据单元是类型相同或相近的消息,这些消息在产生之后是不可修改的.新产生的消息将被追加到流中,而流中的消息也不断地被读取.每条消息可以有相应的键值,这些键值可以被用来对流中的所有消息进行分割.Samza 的工作过程是按需计算转换(或过滤)一组输入流中的消息数据,并将计算结果以消息数据的形式附加到输出流中.因此,运行工作负载可能包含多个工作组,工作组之间可能有流数据的依赖关系,进而将整体计算抽象成一个数据流图框架.Samza 的一大特色在于对状态管理的良好支持,可以用来支撑流数据的连接操作,其状态管理的引擎主要是 RocksDB[87]和 Kafka Log.

Flink 与 Storm 相似,其整个数据处理过程被称为 Stream Dataflow,既定的数据流动框架类似于 Storm 的拓扑.Flink 提取数据流的操作 Source Operator、数据转换(map,aggregate)的操作 Transformation Operator 以及数据流输出的操作 Sink Operator 与 Storm 架构中 Spout 与 Bolts 之间、Bolts 和 Bolts 之间的数据流是高度对应的,区别在于容错机制:Flink 的容错机制是 Checkpoint,通过异步实现并不会打断数据流,因此 Checkpoint 的开启与关闭对吞吐量的影响很小;Storm 采用的是 Ack 机制,开销大而且对吞吐量的影响明显.另外,Flink 提供的 API 相对 Storm 更高级.Storm 的优势主要是具有比较成熟的社区支撑和经过较长期迭代之后的稳定性.目前,Flink 成熟度较低,仍有部分功能需加以完善,如在线的动态资源调整等.Flink 提供的交付保证是“恰好一次”,优于 Storm 的“至少一次”.Flink 广泛受到大公司的亲睐,包括 Uber 和阿里巴巴,其中,阿里巴巴开发了基于 Flink 的流计算系统 Blink,并应用在电商流数据计算中.

Kafka Stream 是 Apache Kafka[80,88]中的一个轻量级流式处理类库,用来支撑 Kafka 中存储数据的流式计算与分析.利用这个类库计算的结果既可以写回 Kafka,也可以作为数据源向外输出.目前的流式计算系统基本都支持以 Kafka Stream 的输出作为数据源,如 Storm 的 Kafka Spout.Kafka Stream 作为一个 Java 类库而非系统,是流计算的重要工具之一.它可以非常方便地嵌入任意 Java 应用中,也可以任意方式打包和部署,除了 Kafka 之外,并没有任何外部依赖.与流计算系统相比,使用 Kafka 所受到的逻辑限制较少,开发者能够更好地控制和使用

Kafka Stream 上的应用.Kafka Stream 提供的是“恰好一次”的交付保证,并且能够利用 RocksDB 进行状态管理.目前,大公司在 Kafka Stream 上的实践较少,相关技术有待进一步成熟.

这些流系统最主要的相似性是针对数据独立性设计分布式并行计算策略,即:针对流式的输入,利用集群进行协作计算,而整体的数据依赖框架一般是一个有向无环图.集群的整体协作更像是流水线的协作,计算框架中的数据依赖所形成的数据流动方向基本上是单一既定的.除了数据处理的先后关系外,这些流系统对不同工作组之间的数据独立性往往要求更高,可实现较好的并行效果.

已有针对这些流系统的基准比较(benchmarking)[89],然而系统的参数各不相同,同一系统在不同参数下的性能也会大相径庭,所以 benchmarking 的结果可信度不足(见表 5).目前,从流系统支持的功能上来看,Flink 的功能是最完备的,有状态管理、高吞吐量和低延迟、支持最严格的“恰好一次”交付保证、可调控内存使用等,并且不会出现容错机制影响吞吐量的情况(如 Storm).虽然 Flink 的社区积累较少,相关 API 不够成熟,但在 Uber、阿里巴巴等公司的使用和推广之下,目前逐渐成为广受欢迎、功能强大的流计算框架.

Table 5 Comparison of stream computing systems

表 5 流计算系统对比

流计算系统	架构	交付保证	流处理类型	状态管理	吞吐量	延迟	成熟度
Storm	分布式	至少一次	实时	×	低	很低	高
Spark Stream	分布式	恰好一次	微批量	×	高	中等	高
Samza	分布式	至少一次	实时/微批量	✓	高	低	中等
Flink	分布式	恰好一次	实时	✓	高	低	低
Kafka Stream	分布式	恰好一次	实时/微批量	✓	高	低	中等

3.4 图数据流管理系统的探索

目前的流数据除了规模大和增长快之外,还有结构复杂的特点,而图模型能够以简易的形式表达出丰富的语义,因此,图模型与流数据模型融合而成的图数据流模型应运而生[90].图数据流的多种定义可以用无限增长的边序列来概括(孤立点的现实意义有限),对应的研究问题除了传统的图计算问题之外,还有针对时间先后信息定义的研究问题,如满足时序约束的路径、子图查询等[91].常见的时序约束有时间先后、时间间隔.例如,阿里巴巴通过网购交易数据中的环形子图的实时监控来追踪通过网购进行恶意信用卡套现的行为[92].如图 1 所示:一个信用卡恶意的套现模式中,套现者向商户发起信用卡虚假购买,银行将真实的资金支付给商户之后,商户通过中间人将资金回流到套现者储蓄卡中,实现恶意套现.整个流程中,参与的对象和资金的流向构成了图的结构,而每个行为环节有其明确的时间先后关系.因此,针对图数据流的管理能够解决很多现实中的重要问题.

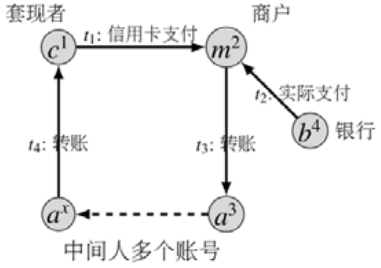

Fig.1 Malicious cash arbitrage model of credit card[92]

图 1 信用卡恶意套现模式[92]

在图数据流的管理方法上,核心的思路仍是在于利用已计算出来的结果来加速当前的计算,并且需要将中间结果维护上的时间和空间代价控制在可接受的范围内[93].以流数据上的子图匹配为例,如果采用静态算法构建复杂的索引的思路来加速查询,则需要针对复杂的索引设计高速更新的算法.然而往往对查询的加速容易增加更新的代价,在无索引的极端情况下,针对子图的匹配需要完全重算;而在另一种极端情况下,即构建复杂的

索引时,往往需要高额的更新时间甚至整个索引重构.因此,图数据流下的计算首要考虑的是计算结果的维护与计算加速的权衡.此外,在图数据流的高速更新场景下,多线程的并发计算仍然具有重要的意义.然而,图数据中不同部分的关联程度较高,如单条边的删除能够导致大量路径特征的改变等,因此,在图数据流下进行并行算法设计和并发访问控制等具有严峻的挑战.

基于目前已有的图数据管理的探索,可以总结出图数据流管理系统所需要解决的三大重要问题.

- 第 1 个是对图数据流中数据的基本操作的支撑,包括边序列的存储、增删改查以及已获取图数据的基本访问操作,如节点度数、邻居等;
- 其次是针对图数据流上的更复杂的挖掘和查询支撑,包括边流行为分析、路径计算以及更复杂的子图结构匹配等.对于复杂查询和挖掘的支撑,所设计的索引一方面需要考虑对计算的加速保证,另一方面需要考虑在高速更新场景下对索引的更新维护代价;
- 第 3 个问题则是事务管理和并发、并行调度等机制的设计,旨在提高系统的吞吐量和缩短响应时间.

3.5 总结与展望

已有的数据流管理系统主要是在传统的数据流管理模型和架构上作了持续性查询的简单扩展,两者在语义和计算逻辑方面有相似之处,大部分数据流管理系统来自高校科研团队,而且这些数据流管理系统已经难以处理大规模复杂数据流的查询和计算.主流的流计算框架采用分布式的计算方式,利用数据独立性的特点进行并行计算,与传统的数据管理模型有很大区别.对数据的格式要求不高,能够处理大规模的多种复杂数据流,有大量的社区支持以及大规模企业的实践与推广,大部分是来自开源社区而鲜有是学术界的科研团队开发的.目前,对大规模生成的复杂数据的计算处理基本上依赖于流计算框架.由于对数据的独立性要求较高,流计算框架不适于处理数据之间有高度关联关系的模型,因此,目前针对图数据流管理系统的研究受到了学术界和工业界的高度关注.

4 时空数据库

4.1 引 言

时空数据库是管理空间、时态以及移动对象数据的数据库系统,与传统的关系型数据相比,时空数据具有多维度、多类型、动态变化、更新快等特点,关系型数据库不能很好地处理此类数据,需要新的有效数据管理方法.近年来,时空数据库在地理信息系统、城市交通管理及分析、计算机图形图像、金融、医疗、基于位置服务等领域有着广泛的应用.根据时空数据特点,时空数据库大致包括以下 3 种.

- 空间数据库:主要处理点、线、区域等二维数据,数据库系统需提供相应的数据类型以支持数据表示、存储、常见拓扑运算操作和高效查询处理,同时需要与传统的关系数据库系统融合以扩展数据库系统处理能力,支持不同类型数据的查询处理;
- 时态数据库:管理数据的时间属性,包括有效时间(valid time)、事务时间(transaction time)等.时间可以为时间点或者时间区间:如果是时间区间,数据库管理系统将以开始和结束时间两个属性或一个区间属性进行存储.不同的应用场景下,时间属性会有相应的特点(例如周期性);
- 移动对象数据库:管理位置随时间连续变化的空间对象,主要有移动点和移动区域:前者仅是位置随时间变化,后者还包括形状和面积的变化.移动对象具有数据量大、位置更新频繁、运算操作复杂等特点.近年来,随着定位设备的不断普及,例如智能手机,采集这类数据越来越容易.同时,与地图兴趣点(例如酒店、餐馆等)相结合,使得移动对象具有语义信息,带来各种新的应用,例如基于位置服务、最优路径规划等.

4.2 主要研究问题

- 数据模型和查询语言

数据模型包含数据类型和运算操作两个方面.时空数据类型包含多个,有些为定长记录存储(例如点、区间),

有些为变长记录存储(例如区域、移动点).运算操作定义时空拓扑运算(例如相交 intersect、重合 overlap),包含语法和语义两个层面:前者描述输入输出参数类型,后者定义抽象层含义.时态和移动对象数据库均处理具有时间因素的对象,数据类型和运算操作都涉及随时间变化的数值,增加了复杂性,主要体现在如何表示数值的动态变化以及拓扑运算定义和求解方法上.移动对象除了要考虑自身的时空属性外,还需要结合对象所在的受限制空间环境,例如道路网、室内空间等,因为位置表示与此相关.此外,不确定性时空数据也是研究内容之一,包括数据模型和查询语言.时空数据类型可作为关系属性嵌套在关系模式下,从而对查询语言 SQL 扩展(运算操作、谓词),以得到时空数据查询语言,支持形式化查询描述.

- 索引结构

根据不同时空数据的特点设计访问结构,以支持快速查询处理,常见的空间和时态索引有 R-tree、K-d Tree、Interval-tree 等.不同的索引结构有相应的运算操作,包括创建、插入、删除、更新及查询.其中,R-tree 是最为广泛使用的结构,为提高查询效率,需对数据排序(例如 z-order),目的是将相似数据存储在邻近结构里,以减少搜索的 I/O 代价.同时,基于该结构的预测模型可以估计查询的 I/O 代价,为进一步优化提供分析的依据.根据时间因素,移动对象索引可分为两类:(i) 历史数据索引,管理移动对象从开始到结束的所有位置和时间;(ii) 当前数据及预测索引,管理移动对象当前位置和速度并进行预测,提供有效的位置更新策略及数据缓存方法.由于移动对象的位置、时空分布、查询等频繁发生变化,主存索引及并行技术往往比外存索引更具优势,自动调优技术对索引的参数动态调整以使性能最优.时空数据索引可以融入语义描述,从而拓展时空数据管理能力,以支持具有语义的时空查询.

- 查询处理及优化

选择查询和最近邻查询是空间和移动对象数据库最常见的两类查询:前者返回在空间/时空查询窗口内的对象,后者返回距离查询目标最近的对象.当查询目标是移动对象时,其最近邻对象也动态地发生变化,称为连续最近邻查询;当返回结果的最近邻是查询目标对象时,称为反向最近邻查询.相似性查询定义评价函数,用于计算对象之间的相似度,返回与查询目标最相近的对象;连接查询用于返回两个数据集中所有符合查询谓词条件(例如相交、重合)的实体对,例如找出所有长江和黄河途经的城市.与选择查询、最近邻查询相比,连接查询的复杂性更高,相关优化技术有数据划分、索引创建、排序等.时空数据查询还包括聚类查询、模式匹配、距离查询等.将关键字或语义描述与位置相结合,可进行具有语义的 ranking 或 top-k 查询,返回对象不仅符合时空约束,而且满足关键字条件,增加了用户对时空对象的全面理解.

- 时空数据管理系统

在定义了抽象模型的基础上,需要有系统实现模型包括数据结构和算法、逻辑设计及实现,同时需要将时空数据模型和关系模型有效融合,从而扩展数据库处理能力.由于时空数据管理系统涉及多种索引结构和查询算法(例如,基于 R-tree 的深度优先和宽度优先),因此需对数据库系统的功能、性能及可扩展性等进行全面测试和评估,发现系统性能瓶颈从而进行优化.这主要通过基准测试完成,一般包括数据集(真实和模拟)、查询集、索引结构及参数设置等,为模拟各种时空数据分布,需要相应的数据产生器及可视化工具.

除上述研究问题外,时空数据库管理还涉及时空数据仓库、时空图数据、时空数据流、基于位置服务(最优路径规划和交通预测)、轨迹数据压缩、时空数据挖掘和分析等方面.

4.3 国内外研究现状

4.3.1 空间数据库

依据不同的环境,空间数据库的研究包括自由空间和受限空间(例如道路网、有障碍空间),主要区别在于距离函数,受限空间的距离计算依赖于最短路径求解,比自由空间要复杂.相关查询有范围查询、最近邻(反向最近邻)、Skyline 查询、动态道路网下最短路径查询和路径规划等,索引技术和搜索策略在查询中起到了关键性作用[94].查询过程一般包括过滤和提炼两个阶段:过滤阶段借助于索引和估计值找到一组备选对象,提炼阶段对每个备选对象进行准确值求解.空间数据库查询还包括最大范围求和、容量受限分配等.在基于位置服务的应用中,隐私保护是一个重要的研究内容,已有的工作包括基于位置隐私的攻击及保护方法,如模糊表示、匿名等.

近 10 年来,空间关键字查询(spatial keyword search)得到了广泛和深入的研究[95],通过将空间位置与文本相结合,用户可以查询同时符合空间和语义条件约束的对象,常见查询有 Top-k、k-NN 等.由于传统的空间数据索引不支持文本数据管理,一般将空间索引与文本索引或位图技术相结合构成混合索引结构,支持同时对空间和文本数据的查询,以缩小搜索范围.

4.3.2 时态数据库

在过去的 20 年里,时态数据管理一直是数据库的活跃领域之一,研究内容包括数据模型、查询语言、索引结构及高效查询算法,各种查询语言也被提出以支持时态数据查询的形式化描述.Enderle 等人基于常见的时态数据索引之一——Interval-tree 设计了相应的外存结构以及在关系数据库系统中的实现方法,可以有效支持相交查询和连接查询[96];Top-k 查询用于返回与查询点(区间)相交且权重最大的 k 个对象.Dignös 等人将时态数据运算操作、转换原则及查询优化方法集成到关系数据库系统内核中(PostgreSQL),以扩展其处理能力[97],商用数据库 Oracle 提供了数据类型 PERIOD 及相关谓词和函数.

近几年,时态数据库的研究主要集中在高效处理各种连接查询(例如 overlap join,merge join)、聚类查询(aggregation)以及数据划分和排列方法(partition/splitter,align).同时,硬件技术(例如多核 CPU)的发展也有助于提高查询效率.不确定性时态数据将时态数据和不确定性相结合,也有不少相关研究工作,包括数据表示及建模、不确定性时态数据查询等;时态数据集成是根据用户指定优先规则对多源时态数据加以融合.

4.3.3 移动对象数据库

早期的移动对象数据库研究主要集中在数据模型、索引和查询处理等[98],代表性索引结构有 TB-Tree、SETI、TPR-tree、STRIPES 等[99],这些结构的差异主要体现在时空数据的管理方法(例如插入原则、时空优先权)上,常见的移动对象查询有范围查询、(连续)最近邻、相似性轨迹、连接查询等.针对大规模移动对象位置的实时更新,有学者提出了有效的更新策略及监控方法,也有学者对不确定性移动对象进行了研究[100].近年来,面向特定应用的移动对象查询得到了广泛的关注,例如轨迹模式匹配、异常现象分析、基于轨迹的用户行为推荐、轨迹压缩等.由于大规模移动对象数据获取已相对容易,对历史数据分析其结果可为应用提供支撑,例如最优路径推荐、最优出行方式及路线规划、交通流量预测等.除了支持时空查询,系统也需要对用户的位置信息进行有效保护,针对这一问题,有学者开展了基于位置隐私保护的研究.

人的运动除了在自由空间下,更多的时候是在受限空间下,例如道路网[101]、有障碍空间[102]和室内环境.不同环境的主要区别在于移动对象位置表示和距离函数:自由空间的位置通过坐标表示,距离函数基于欧式距离;而受限空间下的位置依赖于底层空间环境,距离函数与最短路径相关,求解过程相对复杂.例如,道路网环境下采用 Map-matching 技术,将 GPS 位置(经纬度)映射到道路网从而得到道路网移动对象;在室内环境,移动对象位置获取一般依靠 RFID、WiFi 等技术,位置表示则采用基于符号的表示方法.上述工作均是针对单个空间环境下的移动对象,也有学者将多个环境的不同位置表示方法相融合,形成统一的位置表示方法,支持人的完整运动轨迹表示以及不同运动方式的移动对象数据管理,例如步行→公交车→步行→室内.

在大数据背景下,新应用要求数据包含更多的信息以全面理解用户行为,移动对象数据也从传统的时空数据拓展到具有语义信息和行为描述[103,104].语义轨迹是将 GPS 数据和时空场景相结合,例如兴趣点或用户行为,给移动对象赋予相关描述(可通过数据挖掘算法得出并以标签形式存储),丰富移动对象表示.基于语义轨迹的常见查询有模式挖掘和匹配[103]、时空语义关键字查询、top-k 查询以及移动用户行为分析(规律性地访问某些位置、规避和会合等).基于硬件的技术也被用于大规模轨迹数据查询和分析,例如基于主存的轨迹存储和查询方法、分布式/并行轨迹数据处理平台(基于 Spark 和 Hadoop)、基于 GPU 的交互式时空数据查询等,轨迹数据可视化技术也有相关研究.

4.3.4 时空数据管理系统

时空数据管理系统的设计主要有两种思路:一种是对传统关系数据库管理系统的内核修改或扩展以支持时空数据管理,包括数据类型、访问方法、查询语言等;另一种则通过在应用层和传统数据库管理系统层之间构建一层结构,用于时空数据和传统数据的相互转换,即,在应用层以时空数据处理而在系统存储层还是以传统

数据形式来加以处理.第 1 种方法能够保证效率最优,第 2 种方法则能够在较短时间内达到实际可行的效果.

并行处理技术在时空数据库领域也得到了快速发展,主要用于大规模数据查询处理[105].在空间数据库方面,SpatialHadoop 和 HadoopGIS 均是基于 Hadoop 的空间数据处理系统,Simba 是基于 Spark 技术的空间数据分析系统[106],其对 SparkSQL 进行了扩展,能够有效地支持并发查询.在时态数据库方面,有基于 PostgreSQL 的时态数据查询原型系统和在线实时时态数据分析系统 OceanRT.在移动对象数据库方面,有针对轨迹数据处理的引擎 Hermes、支持多种轨迹数据挖掘操作及可视化系统 MoveMine、基于内存的分布式系统 SharkDB、DITA[107]、轨迹数据在线分析系统 T-Warehouse、大规模轨迹数据管理和分析平台 UlTraMan[108].SECONDO 是一个开源可扩充性数据库管理系统,可对空间、时态和移动对象数据有效管理且支持并行处理[109].

4.4 总结与展望

时空大数据具有多维度、多类型、变化快等特点,给数据库管理系统提出了新的挑战:一方面,需要提供数据类型和运算操作以支持时空数据查询;另一方面,高效查询处理对数据库性能有较高的要求.迄今,时空数据库的发展趋势包含以下几点.

- 具有语义描述的时空数据管理,可分为时空数据和流数据两类:前者针对包含关键字的时空数据进行查询,后者针对高频率的流数据进行连续查询.为增加用户满意度,交互式和探索式查询也是进一步研究的方向之一;
- 并行/分布式环境下的大规模时空数据管理系统.现有的时空数据库原型系统需要在支持的查询种类和通用性数据表示上进一步提升.同时,随着越来越多的时空数据管理系统被研发,需要在统一标准下对系统的功能及性能进行全面测试和评估(benchmark).新型存储设备(例如,SSD 具有快速随机写等特点)的发展,将给位置频繁更新的移动对象研究带来新的契机;
- 具有智能性的时空数据库系统.在人工智能技术快速发展的背景下,如何融入机器学习方法以增加系统的智能性,是新一代时空数据库管理系统研究的内容,即,系统根据当前处理数据及查询的特点自动进行索引结构和相关算法的调整以使性能最优,例如参数配置、数据划分、缓存设置等.

5 众包数据库

5.1 引 言

Web 2.0 时代涌现出了海量的在线互联网应用.这些应用在悄然改变人们生活的同时,也为传统的人本计算(human computation)提供了一种通过群体智慧求解问题的新模式——众包[110],即“一种把过去由专职员工执行的任务,通过公开的 Web 平台,以自愿的形式外包给非特定的解决方案提供者群体来完成的分布式问题求解模式”[111].在过去的 10 余年里,基于 Web 的众包技术已与人们的日常生活息息相关.维基百科(Wikipedia)、雅虎问答(Yahoo! Answers)以及百度知道等各类“问答系统”平台均是这一技术的典型代表.近年来,移动互联网的爆发更是催生出众包的新形态——时空众包[112,113].这是一种新型众包计算模式,以时空数据管理平台为基础,将具有时空特性的众包任务分配给非特定的众包参与者群体为核心操作,并要求众包参与者以主动或被动的方式来完成众包任务,并满足任务所指定的时空约束条件.时空众包因具有信息世界与物理世界相联系的特点,使其成为共享经济的新型计算范式.实时专车类应用滴滴出行和物流派送类应用百度外卖都是共享经济时代时空众包应用的典型代表,并取得了巨大成功.众包在通过互联网汇聚群体智慧求解各类问题的过程中,动态生成海量多源异构数据,对这些数据进行有效的管理,是发挥众包应用价值的关键.

5.2 主要研究问题

众包数据管理的主要研究问题来自众包工作流程中的 3 个不同阶段.

- 第 1 阶段:众包任务的发布者将任务提交至众包数据管理系统,系统将任务分配给适当的众包参与者执行;
- 第 2 阶段:众包参与者将任务执行结果提交给众包数据管理系统,系统对这些结果进行集成和处理后,

将最终结果反馈给众包任务发布者;

- 第 3 阶段:众包任务发布者收到系统反馈的任务执行结果后,根据任务完成质量等因素向众包参与者提供适当的报酬.

上述工作流程中的 3 个阶段反映了众包数据管理中的 3 个研究问题.

- 任务分配:该问题涉及众包工作流程的第 1 阶段,其核心目标是将众包任务分配给合适的参与者,以实现各类不同的优化目标.例如,在基于 Web 的众包中,众包任务会根据其类型被分配给擅长执行该类任务的参与者,以优化任务完成的质量;在时空众包中,众包任务通常会被分配给任务位置附近的参与者,以优化任务发布者的等待时间.对该问题的研究主要面临以下难点:(1) 任务分配具有动态性,即众包任务和参与者动态出现在众包平台上;(2) 任务分配具有约束复杂性,即将任务分配给参与者通常需满足各项约束条件;(3) 任务分配还要求兼顾有效性和效率.上述难点要求研究人员在对众包数据进行有效存储和索引的基础上,设计高效的求解算法;
- 质量控制:该问题涉及众包工作流程的第 2 阶段.众包通过汇聚人类的群体智慧求解各类问题.因为人难免会犯错,所以众包数据管理系统通常无法将众包参与者所反馈的任务执行结果直接反馈给任务发布者,而需要对结果进行质量处理.在基于 Web 的众包中,系统通常根据对众包参与者可靠程度、擅长领域和对问题的难度等因素建模,通过投票等方式对任务的结果正确性进行推断,并将结果的可靠程度一并返回给任务发布者;在时空众包中,系统则主要会考虑到任务位置与参与者的距离以及参与者的上线等时空因素,对任务完成的质量进行控制;
- 激励机制:该问题涉及众包工作流程的第 3 阶段.众包参与者执行任务需要花费各类稀缺资源,例如:在基于 Web 的众包中,参与者执行各类图片标注任务需花费注意力;而在时空众包中,网约车平台的私家车车主则需付出车辆的使用成本等.众包数据管理系统需要对参与者进行有效激励,以使得他们愿意继续留在众包系统中以提供服务.虽然金钱激励是直接而有效的激励方式,但是制定健全而合理的激励额度并不是一个简单的问题.在基于 Web 的众包中,参与者完成任务的质量以及任务的难度是激励机制设计的重要参考指标,而在时空众包中,不同时空区域内任务和参与者之间的供需状况也会对激励机制产生影响.

除上述研究问题外,众包数据管理还研究众包任务的设计、对众包参与者隐私的保护等方面.

5.3 国内外研究现状

近年来,国内外研究人员已展开了众包数据处理的相关研究,并取得了不少技术突破[114−116].众包数据处理可分为两类:众包数据管理机制与基于众包策略的数据处理技术.此外,基于上述研究,人们还设计和开发了各类众包数据管理系统.

5.3.1 众包数据管理机制

下文将对任务分配、质量控制与激励机制这 3 类问题进行简单分析.

1) 任务分配

任务分配一直都是算法研究领域的重要问题之一,文献[117]首先针对单一用户所提供的同一类型众包任务定义了在线任务分配问题,旨在一段时间内,以在线方式最大化众包任务分配数量.文献[117]设计了一种采用原始对偶模式(primal-dual schema)的近似算法用于在线任务分配.后续工作扩展了在线任务分配问题的定义,提出了在线异构任务分配问题,也针对此问题扩展了原始对偶模式近似算法[118].文献[119]针对时空众包中任务和参与者均动态出现的应用场景,提出了具有竞争比保证的任务分配算法,以最大化总收益.文献[120]研究了最小化众包参与者总移动距离的双边在线任务分配问题.文献[121]考虑到众包任务执行场所的影响,研究了面向 3 类对象的在线任务分配问题.文献[122]提出了基于预测[123]的众包参与者调度和众包任务分配算法.文献[124]针对拼车问题提出了通用的优化方法,其解决方案适用于众包参与者具有容量约束的众包任务分配问题.

2) 质量控制

众包技术已被应用于众多领域,质量控制机制是众包研究的核心问题之一,针对各类具体应用的众包质量

控制研究被广泛提出.具体而言,众包数据处理的质量控制机制研究可分为如下两类:(1) 众包参与者误差估计;(2) 众包结果的集成机制.前者侧重于对不同众包参与者单一个体的误差估计;而后者需根据误差估计所得误差概率,再进一步对不同众包参与者的反馈进行符合质量控制要求的结果集成,并汇聚成最终答案.下文将对这两方面研究分别加以简要回顾.

众包参与者误差估计通常是指采用少数服从多数原则、EM 算法或其他学习算法,根据众包参与者的历史数据推断出众包参与者的误差率.文献[125]针对标注查询设计了一种概率图模型,用来推断任务标注、每位众包参与者的误差率与众包任务难度.文献[126]提出了一种经验贝叶斯算法——SpEM,通过 EM 框架中的迭代过程来清除水军用户,从而采用真实标注对最终标注结果进行估计.另一类有代表性的工作是针对不同的众包任务选择不同的众包参与者,从而保证执行每个众包任务的众包参与者的误差率尽可能地小.例如,文献[127]提出了一种选择性重复标注的方法,通过重复标注来提高标注结果的准确性.以上 3 项研究均提供了启发式的众包参与者误差估计算法,而如下 3 项工作又进一步对其所估计的误差率进行了定界分析:当给定一个众包参与者的集合时,文献[128]证明了全部众包参与者集体的误差估计值上界,但无法用于单一众包参与者误差的估计;对于单一的众包任务,文献[129]证明了每项任务所最终得到正确答案的概率上界与下界,但其所分析的目标为众包任务而非众包参与者的误差率;最后,文献[130]对单一众包参与者误差进行了区间估计分析,并对众包参与者的误差估计提供了置信区间估计算法.

众包结果的集成机制:根据众包参与者误差的估计值,对一项众包任务的不同众包参与者反馈进行集成并产生最终任务结果,也是众包质量控制机制的一个主要研究方向.文献[131]提出了一个质量敏感应答模型 CDAS,包含预测模型与验证模型两个子模型:前者用于估计一项指定众包任务所需众包参与者的数量,后者则根据用户反馈选择最优答案.另一类有代表性的工作是根据众包参与者的误差率,为一项众包任务发现一个最优的众包参与者群体,又被称为陪审团(jury).最优有两类定义:第 1 类是指在给定众包预算成本的条件下,发现一个陪审团,使其对此任务的陪审团错误率(jury error rate,简称 JER)尽可能地小[132];另一类定义是指在给定陪审团错误率的条件下,发现一个陪审团,使其支付成本尽可能地小[133].文献[132]证明了第 1 类定义是 NP-Hard 的,并给出相应的近似求解算法;随后也对第 2 类定义给出了相应的近似算法[133].

(3) 激励机制

激励机制也一直是众包机制研究的核心.对于一项指定的众包任务,在众包平台中应该如何定价才能保证有足够的众包参与者协同完成此任务.文献[134]针对通用在线众包平台上的众包市场提出了两种定价策略:第 1 种是在给定众包任务预算约束的条件下,通过优化定价策略来最大化被分配的任务数量;第 2 种是在给定需完成任务数量的约束条件下,最小化支付成本.针对每种定价策略,文献[135]分别给出了常数竞争比的在线近似算法.文献[135]提出了一个基于遗憾最小化方法(regret minimization approach)的在线定价机制,该机制提出一种基于贪心策略的采购拍卖算法,从而获得近似最优的求解保障.文献[136]着重研究了完成众包任务的时间延迟与众包定价策略之间的关系.针对给定任务完成截止时间与给定任务预算成本两类不同约束条件,分别设计了一系列基于随机过程的最优定价算法,并证明了近似算法存在近似最优解.文献[137]研究了针对复杂众包任务的激励机制,通过将复杂任务适当地分解成微任务,在满足任务完成质量的约束下最小化支付成本.文献[138]针对时空众包市场中不同空间区域和时间段供需不平衡的特点,提出了基于匹配的动态定价策略.

综上所述,3 类众包数据管理机制的研究都针对于现存的在线众包平台,如 AMT、CrowdFlower 和 oDesk 等.下面两节将分别进一步阐述 3 类机制在众包数据处理和众包数据管理系统中的应用和扩展工作.

5.3.2 基于众包策略的数据处理技术

众包数据处理技术作为当前数据库与数据挖掘领域一项新兴的研究热点,主要侧重于如何将众包策略融入到传统数据库管理系统的经典查询处理与挖掘操作之中,从而提高数据处理的质量或拓宽查询处理与挖掘操作的应用范围.本节主要对现存的基于众包策略的几类经典查询与挖掘技术进行简要的回顾.

(1) 筛选查询

在基于众包策略的查询处理中,最基础的一项查询处理操作是筛选查询(flitering query),文献[139]首次在

众包数据处理的背景下提出此类查询和其求解框架 CrowdScreen.CrowdScreen 的主要贡献在于,其并不认为简单的少数服从多数原则是合理的,并分别以众包期望成本与结果期望误差率为优化目标,将众包的筛选查询细化为 5 类情况,并对每种情况给出了确定性与概率性求解算法.其后续扩展工作进一步放松了文献[139]的众多假设条件,从而进一步提高了通用性[140].通过增加众包参与者们的先验信息与后验信息,以及对不同用户能力的细化分析,大幅度提高了算法执行效率,同时也降低了众包支付成本.

(2) 排序查询与 Top-*k* 查询

排序查询(ranking query)与 Top-*k* 查询始终是数据处理技术中的核心操作之一.文献[141]首先定义了众包环境下的最大者查询问题,即 Top-1 查询,该问题旨在返回满足投票矩阵的条件下具有最大可能性的数值最大者.文献[141]也证明了该问题是 NP-Hard,设计了一系列的启发式求解算法.文献[142]扩展了上述研究结果,提出了基于众包的 Top-*k* 查询,即:在满足误差阈值的情况下,最小化众包支付成本(即众包参与者为成对数据项进行比较的次数).文献[142]证明了此众包期望支付成本的理论下界.文献[143]提出了一个基于打分和排序的众包 Top-*k* 查询计算框架.

(3) 连接查询、实体同一与模式匹配

连接查询是数据库领域的经典查询操作之一,其在众包数据处理中通常是指基于众包的实体同一性查询.实体同一性查询(entity resolution)与模式匹配(schema matching)是关系数据库的数据集成过程中两类重要的操作.

实体同一(基于众包的连接查询):文献[144]提出一种人机混合的实体同一性查询通用求解框架——CrowdER.该框架假设众包参与者的反馈一定正确,将基于众包的实体同一性查询归约到基于聚类的 HIT (human intelligence task)产生问题,并证明了此问题是 NP-Hard,同时给出两种高效的启发式算法求解该问题.文献[145]扩展了文献[144]的求解框架,并融入了实体间传递性的概念,大幅度提升了基于众包的实体同一性查询效率.文献[146]考虑了不同实体之间同一性的先验概率分布,以最小化众包任务期望询问数量为目标,证明该优化问题是 NP-Hard,并给出了相应的随机求解算法.

模式匹配:文献[147]首先提出了基于众包的模式匹配问题,并通过可能世界语义对该问题进行了形式化描述,采用信息熵来度量模式匹配中任意两模式相似度的稳定性.根据信息熵设计了基于贪心策略的模式匹配问询机制,从而最小化众包支付成本.文献[148]研究了基于众包策略的 Web 表格匹配问题,提出一种新型的效用函数,既包含匹配难度,又可测量不同 Web 表格属性间的相似性.基于此效用函数,文献[148]证明 Web 表格匹配问题是 NP-Hard,并给出了近似比为(1–1/e)的近似算法.

(4) 计数问题与枚举查询

计数问题是集成查询(aggregation query)的基础,也是传统关系数据库理论中 Group By 子句执行的关键.为了扩展传统集成查询的适用范围,文献[149]首先提出了基于众包的计数问题,即:给定一个数据项集合与一个筛选规则,令每名众包参与者反馈符合筛选规则的数据项个数(可以不很精确),汇聚全部众包参与者的反馈并估计最终符合筛选规则的数据项个数.不同于传统的基于二选一投票类型的众包操作,文献[149]的众包基础操作就是一个粗糙的计数问题.同时,也针对网络水军(spammers)的检测与清除设计了处理机制,最终给出了一种基于置信区间的计数估计算法.

枚举查询也是传统数据处理的核心问题之一,其结果集的唯一性源自于传统数据库中的封闭世界假设(closed world assumption).为了扩展众包数据处理的范围,文献[150]打破了封闭世界假设,并提出了基于众包策略的枚举查询.在没有封闭数据库支持的情况下,新型查询将处理如“枚举北京航空航天大学喜爱看电影的同学”的查询问题.核心思想在于通过不同众包参与者的反馈分析其所枚举的交集比重,采用置信区间以估计值快速计算枚举结果的可信度,同时,在满足置信度约束的情况下最小化众包支付成本.

(5) 关联规则挖掘与聚类分析

本部分主要介绍两类基于众包策略的数据挖掘研究.其中,关联规则挖掘与聚类分析都是传统数据挖掘中的核心技术,最近的相关研究已将众包策略有效地融入到两类技术之中,扩展了它们的适用范围.文献[151]打破

了传统数据库中的“封闭世界假设”,进而提出基于众包的关联规则挖掘问题.旨在通过收集人脑中的关联规则经验,通过询问众包参与者形如“*A* 与 *B* 是否经常一起出现?”的问题来估计不同项集的支持度,从而估计出关联规则的置信度.文献[152]提出了一个贝叶斯模型来学习不同众包参与者的聚类特点与数据项自身的结构性特点,从而获取基于众包的高质量聚类结果.

5.3.3 众包数据管理系统

近年来,随着对众包数据管理机制和基于众包策略的数据处理技术的研究日益深入,依据上述成果的众包数据管理系统也被设计和开发出来.代表性的众包数据管理系统有众包数据库 CrowdDB[153]、Deco[154]、Qurk[155]、CDB[156]和众包数据管理平台 DOCS[157]、gMission[158]等,下面分别对这些系统进行简要介绍.

CrowdDB 通过引入众包机制,使得数据库系统能够完成一些本来无法完成的查询操作,如针对未知或不完整信息的查询、涉及主观比较的查询等.CrowdDB 使用扩展自 SQL 的查询语言 CrowdSQL,支持用户使用“CROWD”关键字定义数据表和属性,指示数据表的内容和属性列中的值需要通过众包补充完整.CrowdDB 设计和实现了相应的查询编译和运行时系统,能够通过自动生成的用户接口利用众包获取数据.

Deco 与 CrowdDB 类似,同样提供了基于 SQL 的查询语言,允许用户通过该语言在系统中存储的关系数据和通过众包获取的数据上进行各类查询.此外,Deco 设计了具备可扩展性和通用性的数据模型,支持数据清洗等功能,并定义了精确的查询语义.Deco 的查询处理器使用了一种新型的推拉混合式执行模型,在执行查询的同时,兼顾众包本身具有的时延、不确定性等特点所带来的挑战.

Qurk 使用基于 SQL 的查询语言,并支持用户定义函数(user-defined functions,简称 UDFs).为了方便用户使用 UDFs,Qurk 提供了预定义的众包任务模板,可以生成支持过滤和排序等众包任务的众包界面.此外,Qurk 设计了任务缓存和任务模型进行查询优化:任务缓存将先前众包任务的结果缓存起来,任务模型基于已经收集到的众包数据训练模型预测众包任务的结果.无法通过任务缓存和任务模型获取的数据,将由系统自动地通过众包方式获取.

CDB 定义了查询语言 CQL 并使用基于图的模型进行查询优化,可提供细粒度的元组级别优化.在该模型的基础上,CDB 采取了一种统一框架,可对时延、成本、质量等多种目标进行优化.

表 6 对上述众包数据库进行了比较[159].

Table 6 Comparison of crowdsourcing DB systems

表 6 众包数据库系统的比较

		CrowdDB	Deco	Qurk	CDB
支持操作	收集(collect)	√	√		√
	填充(fill)	√	√		√
	选择(select)	√	√	√	√
	连接(join)	√	√	√	√
	排序(rank)	√		√	√
优化目标	成本(cost)	√	√	√	√
	质量(quality)	√	√	√	√
	时延(lantency)				√

除了上述众包数据库之外,学术界也提出了一些众包数据管理平台.DOCS 是一个具备领域感知能力的众包数据管理平台,它通过知识库对众包参与者和众包任务涉及的知识领域进行分析,并基于分析结果对参与者完成不同领域任务的质量进行细粒度建模.DOCS 同时具备真值推断和动态任务分配功能,能够利用对参与者擅长领域的细粒度建模准确推断出任务的真实结果,并把众包任务分配给擅长任务相关领域的众包参与者.gMission 是一个开源的通用时空众包数据管理平台,具有任务分配和结果集成功能.

5.4 总结与展望

现存的研究已将众包策略成功地集成到传统数据处理技术之中.众包策略可以提高传统数据处理技术的准确性,同时可打破传统数据处理的“封闭世界假设”,从而扩展传统数据处理技术的适用范围.此外,近年来,随

着移动互联网技术的广泛应用,时空众包数据处理技术正在成为众包数据处理研究中的新兴热点.为了进一步完善众包数据管理系统,以下问题还有待解决.

- 众包数据管理系统的查询优化问题.不同于传统数据库的查询优化问题,其不同查询方案的执行成本可得到较为准确的估计,且优化目标较为单一;在众包数据库中,由于所涉及的众包群体存在较大不确定性,且优化目标涉及时延、质量、花费等诸多复杂因素,众包数据管理系统的查询优化问题还未得到有效解决;
- 时空众包数据的存储与索引问题.因为时空众包数据包含动态的时空数据、高维属性数据与时空冲突数据,所以,传统的离线静态场景中的时空数据查询索引技术并不适用.如何对空间众包数据进行有效的存储与索引,进而支持各类时空众包数据查询处理,是未来研究的关键.一个极具潜力的研究问题是设计一种针对时空众包数据特性的存储策略与通用的索引结构;
- 隐私与数据保护问题.众包机制已经广泛应用于数据标注和收集等任务,在这些任务,特别是时空众包任务的执行过程中,存在任务发布者数据泄露以及任务执行者隐私泄露的问题.通用的隐私和数据保护方案可在一定程度上缓解该类问题,但均会对众包的效率和质量造成影响.

6 其他研究热点

除了上述研究方向外,数据库领域还涌现出很多其他研究热点.例如:新硬件技术(包括内存技术)改变了数据库的底层框架设计和查询优化的代价模型;近似查询技术能够以更小的代价支持更大规模数据上的查询;数据的可视化技术为用户提供更加友好的数据展示方式.下面分别简述这些研究热点.

6.1 新硬件

新硬件技术的发展为数据管理技术带来新的挑战,也带来明显的机遇.作为系统软件,数据库底层需要针对新硬件的发展做出适应性调整,充分利用新硬件带来的便利,同时避免新硬件自身约束导致的新瓶颈.目前研究较多的新硬件包括高性能和专用处理器、高速网络、大内存(见下一小节)和非易失性内存等.

- 基于高性能和专用处理器的数据管理方法:目前,高性能处理器技术进入多核时代.相对应地,数据库底层核心算法需要充分考虑多核并行的能力,重新设计连接、排序等基本操作[160].图形处理器 GPU[161]、现场可编程门阵列 FPGA[162]等专用处理器具备更大规模的数据并行操作能力,从而提升数据的向量处理效率,支持数据库内核范围内的机器学习等任务.同时,不同特性异构硬件的协同操作也成为研究问题.例如,GPU 的显存相比于内存容量要小,数据加载到 GPU 显存的操作代价较高,面向数据管理的 CPU 和 GPU 的协同架构就是希望充分发挥不同硬件的优势[163],避免其中可能的瓶颈操作;
- 基于高速网络连接的数据管理方法:在传统的分布式数据库或者并行数据库环境中,网络的传输速率远低于内存访问速率,在分布式查询和事务管理等部件中都将网络传输作为主要代价之一.随着 RDMA 等高速网络技术的发展,网络传输代价大幅度降低.现有的研究工作基于 RDMA 高速网络的特性,设计了新的分布式连接方法[164]和分布式并发控制策略[165]等;
- 基于非易失存储的数据管理方法:非易失型存储器支持内存式的按字节的高速寻址,同时支持外存式的持久化能力,得到数据管理领域的高度关注.非易失型存储器存在读写操作不对称和写耗损等约束.目前,非易失型存储器的价格较高、容量较小.现有的研究讨论了非易失性存储器和内存、闪存等不同特征的存储介质在体系结构层面的结合方式[166]、非易失存储环境中的数据库恢复机制[167]等问题.

6.2 内存数据库

内存数据库就是以内存为主要数据存储介质、在内存中直接对数据进行操作的数据库.相对于以磁盘为主要存储介质的传统数据库,内存数据库带来数量级的性能提升.内存数据库的发展受益于内存价格的降低以及内存容量的迅猛增加.同时,目前内存常见的 64 位寻址,使得将全部数据加载到内存空间成为可能.

传统数据库查询执行的主要瓶颈在于 I/O 操作,而在内存数据库中,内外存数据交换不再成为代价的主要

来源.内存数据库需要考虑现有 CPU 特性对内存操作的影响,如 CPU 中的缓存、指令和数据的预取、共享数据结构上并发访问的控制机制等.上述变化导致内存数据库在数据组织、数据索引、事务机制、查询优化等方面与传统数据库不同[168].

- 数据组织.从 CPU 的角度,内存数据库中数据可以按照其处理器核进行划分,同一个划分中数据操作串行,减少并发控制带来的各种代价;也可以采用所有处理器核都可以访问全部数据的方式.从数据版本的角度,内存数据库通常采用多版本机制,提升查询处理效率.从行列存储的角度,内存数据库可以选择行存储或者列存储,同时,其列存储在交易型应用中表现良好[169];
- 数据索引.内存数据库索引设计主要考虑两个主要因素:首先,内存索引节点的大小一般与 CPU 缓存大小相关,其索引数据的存储满足一定的连续性,从而在索引操作过程中提升 CPU 缓存的命中率,典型工作如 CSB+树[170]等;其次,内存索引结构的设计需要考虑多核环境中的并发查询和更新,尽量减少内存数据结构中并发锁的使用,降低索引维护代价,典型工作如 BW 树[171]等;
- 事务机制.内存数据库的并发控制机制主要采用传统数据库的多版本并发控制协议[172],通过保存不同版本,从而支持无阻塞高效率的读取操作,部分协议采用乐观并发机制,引入验证阶段判定事务是否可以提交.此外,考虑 CPU 的多核特性,内存数据库还可以在数据按照处理器核来划分的情况下,设计划分串行执行协议[173],即:同一划分内数据串行操作,划分之间数据并行操作.目标是减少多核对同一数据并发控制代价或者冲突导致的回滚代价.

6.3 近似查询

相对于传统的数据库精确查询,近似查询能够以较小的代价获得近似的查询结果.近似查询技术在大规模数据分析、趋势发现、快速可视化等领域都有一定的应用前景.近似查询技术可以通过不同维度来刻画,包括所支持查询的表达能力、错误模型和精度保证、运行时刻代价节省以及预计算结构的维护代价等[174].通常认为,几乎不可能构建这样的近似查询系统:其能够支持丰富 SQL 特征查询,运行时刻提供高质量的查询质量保证,并且能够明显地节省运行时刻的代价.近似查询技术可以粗略划分为两大类——在线查询执行和通过预计算结果支持查询[175].

- 在线查询执行.其基本思想是:查询能够时刻反馈当前运行结果,同时提供结果精度区间.随着时间的推移,参与计算的数据量增多,查询结果精度不断提升.用户可以决定什么启动查询或者停止查询.这一模式在粗略观察数据的场景中得到应用.此外,这一模式没有预计算结果的维护代价.执行过程中一般采用采样策略,查询基于动态采样的数据执行.现有工作提出了不同采样策略及其查询处理策略,例如,偏有效查询结果集合采样的 Wander 连接策略[176]等.某些在线查询的方法已经集成到开源或者商业数据库系统中;
- 通过预计算结果支持查询.其基本思想是:引入可控误差,将大数据预计算为某种形式的、与原有数据某个特征类似的“小数据”.后续的查询直接基于预计算结果执行.相对于在线采样,基于预算结果的查询执行方式更加高效.针对特定的预计算结果和查询,这种模式能够给出查询的精度保证.但是随着底层数据的变化,这些预计算的数据也需要维护.根据预计算结果的形态不同,预计算模式可以进一步划分为基于直方图的近似查询(等宽直方图、等高直方图、压缩直方图、自适应直方图等)、基于采样的近似查询(随机采样、有偏采样、重要性采样等)、基于小波的近似查询等.这一模式也有相关的原型系统,如 BlinkDB[177]等.

6.4 数据可视化

数据可视化利用计算机图形学、数据分析、用户交互界面等技术,通过数据建模等手段,为用户提供有效的数据呈现方式.数据可视化能够帮助用户迅速理解数据、定位问题.近期发展的可视化技术可以从不同维度来刻画,如可视化后台的数据类型、不同类型的可视化交互技术等[178].我们主要从数据类型的角度,分析数据可视化技术的进展.

- 图数据可视化:图数据在很多领域中自然存在,如社交网络等.图数据的海量规模(包括海量的节点和边数据)以及有限的可视空间限制,成为图数据可视化的主要挑战.现有工作侧重于图简化的思路[179],通过边聚集或者点聚集,构建不同层次的图,同时引入交互策略,支持用户对其感兴趣的部分作进一步的动态分析;
- 时空数据可视化:时空数据是包含时间维度和空间维度的数据,其空间维度通常与地理系统加以结合.为了展示对象随着时空维度的变化情况,采用属性可视化技术,如将事件流和地理流相结合的Flowmap方式[180]、时间-空间-事件等信息的三维立方体方式等;
- 多维数据可视化:多维数据是包括多个维度属性的数据,如数据仓库中销售数据等.多维数据的可视化技术目标是更加友好地呈现数据,从而方便用户对整体分布和不同维度之间关系的理解.具体的展示方式包括散点图、平行坐标等[181].

7 总 结

数据相关技术的发展给整个社会带来了巨大的变革,也给相关的技术领域带来了巨大的挑战.不同领域的学者均尝试从自身的角度出发来解决大数据的种种问题,基于这些成果构建了若干实际可行的新型系统.但是随着数据规模以及应用需求的进一步发展,未来的数据管理技术仍然面临着新的问题和转变.

- 新型数据管理系统需要更自然、更高效地支持不同类型、不同来源的数据.针对应用中出现的不同类型数据管理需求,现有的系统大多是通过构建专用系统来解决,例如图数据管理系统、时空数据管理系统等.而应用中,这些数据混杂在一起,按照数据类型划分到不同数据系统中,这种管理方式不高效,也不自然.新型数据管理系统需要提供通用的底层数据模型,统一支持不同类型数据的存储、查询、分析、优化等操作;
- 新型数据管理系统需要在体系结构方面实现扩展.为了减少系统复杂性,提高系统稳定性,在现阶段通过松耦合包容不同类型数据的管理系统,为用户提供统一的数据管理和分析服务,是支持不同数据模型的可行技术路线.此外,数据管理系统需要考虑异构的计算资源.异构计算环境广泛存在于真实应用场景中,包括资源共享与竞争、网络和计算能力差异以及新型硬件带来的异构性.异构计算环境会对新型数据管理系统的效率带来极大的影响,同时,新型硬件的发展也为新型数据管理系统提供了新的机遇;
- 新型数据管理系统需要在计算模型方面实现扩展,满足不同数据模型的管理需求,支持系统松耦合管理体系.机器学习是目前的技术热点,在自然语言处理、计算机视觉等方面取得突破.新型数据管理系统需要和机器学习实现融合,包括在数据库内核层面实现机器学习方法,深度分析数据,提供更加强大、友好的用户接口.此外,机器学习技术为现有数据操作实现带来新的思路,如通过学习构建索引、自然语言查询等,需要在数据管理内核方面融入更多的机器学习技术,通过紧耦合提升现有数据管理系统的效率和可用性.

References:

[1] Gilbert S, Lynch NA. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant Web services. SIGACT News, 2002,33(2):51−59.

[2] https://aws.amazon.com/cn/rds/

[3] Krishnan SPT, Gonzalez JLU. Building Your Next Big Thing with Google Cloud Platform. Apress, 2015. 159−183.

[4] Talaat S. Pro PowerShell for Microsoft Azure. Apress, 2015. 95−115.

[5] https://cn.aliyun.com/product/rds

[6] https://cloud.tencent.com/product/cdb

[7] https://www.163yun.com/

[8] Ongaro D, Ousterhout JK. In search of an understandable consensus algorithm. In: Proc. of the USENIX Annual Technical Conf. 2014. 305−319.

[9] George L. Hbase—The Definitive Guide: Random Access to Your Planet-Size Data. O'Reilly, 2011. 1−522.

[10] Lakshman A, Malik P. Cassandra: A decentralized structured storage system. Operating Systems Review, 2010,44(2):35−40.

[11] https://redis.io/

[12] http://memcachedb.org/

[13] Sivasubramanian S. Amazon DynamoDB: A seamlessly scalable non-relational database service. In: Proc. of the SIGMOD Conf. 2012. 729−730.

[14] Sciore E. SimpleDB: A simple Java-based multiuser syst for teaching database internals. In: Proc. of the SIGCSE. 2007. 561−565.

[15] Chodorow K, Dirolf M. MongoDB—The Definitive Guide: Powerful and Scalable Data Storage. O'Reilly, 2010. 1−193.

[16] Anderson JC, Lehnardt J, Slater N. CouchDB—The Definitive Guide: Time to Relax. O'Reilly, 2010. 1−245.

[17] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber R. Bigtable: A distributed storage system for structured data. In: Proc. of the OSDI. 2006. 205−218.

[18] Ghemawat S, Gobioff H, Leung ST. The Google file system. In: Proc. of the SOSP. 2003. 29−43.

[19] Borthakur D. HDFS architecture guide. In: Proc. of the Hadoop Apache Project. 2008.

[20] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. In: Proc. of the OSDI. 2004. 137−150.

[21] Mohan C, Haderle DJ, Lindsay BG, Pirahesh H, Schwarz PM. ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Trans. on Database Systems, 1992,17(1):94−162.

[22] Olson MA, Bostic K, Seltzer MI. Berkeley DB. In: Proc. of the USENIX Annual Technical Conf. on FREENIX Track. 1999. 183−191.

[23] Palankar MR, Iamnitchi A, Ripeanu M, Garfinkel S. Amazon S3 for science grids: A viable solution? In: Proc. of the 2008 Int'l Workshop on Data-Aware Distributed Computing (DADC 2008). 2008. 55−64.

[24] Bernstein PA, Goodman N. Multiversion concurrency control—Theory and algorithms. ACM Trans. on Database Systems, 1983, 8(4):465−483.

[25] http://fallabs.com/tokyocabinet/

[26] Feinberg A. Project Voldemort: Reliable distributed storage. In: Proc. of the 27th Int'l Conf. on Data Engineering. 2011.

[27] Klophaus R, Core R. Building distributed applications without shared state. In: Proc. of the ACM SIGPLAN Commercial Users of Functional Programming. 2010.

[28] Corbett JC, Dean J, Epstein M, Fikes A, Frost CC, Furman JJ, Ghemawat S, Gubarev A, Heiser C, Hochschild P, Hsieh WC, Kanthak S, Kogan E, Li HY, Lloyd A, Melnik S, Mwaura D, Nagle D, Quinlan S, Rao R, Rolig L, Saito Y, Szymaniak M, Taylor C, Wang R, Woodford D. Spanner: Google's globally-distributed database. In: Proc. of the OSDI. 2012. 261−264.

[29] Shute J, Vingralek R, Samwel B, Handy B, Whipkey C, Rollins E, Oancea M, Littlefield K, Menestrina D, Ellner S, Cieslewicz J, Rae I, Stancescu T, Apte H. F1: A distributed SQL database that scales. PVLDB, 2013,6(11):1068−1079.

[30] https://oceanbase.alipay.com/

[31] https://cloud.tencent.com/product/dcdb

[32] https://www.oschina.net/p/tidb

[33] WeChat _Wikipedia. https://en.wikipedia.org/wiki/WeChat

[34] Linked data. http://linkeddata.org/

[35] Graph 500|large-scale benchmarks. http://graph500.org/

[36] Dorogovtsev SN, Goltsev AV, Mendes JFF. *K*-Core organization of complex networks. Physical Review Letters, 2006,96(4): 040601.

[37] Trusses CJ. Cohesive subgraphs for social network analysis. National Security Agency Technical Report, 2008. 16.

[38] Palla G, Derényi I, Farkas I, *et al*. Uncovering the overlapping community structure of complex networks in nature and society. Nature, 2005,435(7043):814.

[39] Page L, Brin S, Motwani R, *et al*. The PageRank Citation Ranking: Bringing Order to the Web. Stanford InfoLab, 1999. http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf

[40] Jeh G, Widom J. SimRank: A measure of structural-context similarity. In: Proc. of the 8th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2002. 538−543.

[41] Brin S, Page L. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 1998, 30(1-7):107−117.

[42] Deshpande M, Kuramochi M, Wale N, *et al*. Frequent substructure-based approaches for classifying chemical compounds. IEEE Trans. on Knowledge and Data Engineering, 2005,17(8):1036−1050.

[43] Wasserman S, Faust K. Social Network Analysis: Methods and Applications. Cambridge University Press, 1994.

[44] Zou L, Mo J, Chen L, Özsu MT, Zhao D. gStore: Answering SPARQL queries via subgraph matching. PVLDB, 2011,4(8): 482−493.

[45] Zou L, Özsu MT, Chen L, Shen X, Huang R, Zhao D. gStore: A graph-based SPARQL query engine. The VLDB Journal, 2014, 23(4):565−590.

[46] Malewicz G, Austern MH, Bik AJC, Dehnert JC, Horn I, Leiser N, Czajkowski G. Pregel: A system for large-scale graph processing. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2010. 505−516.

[47] Han M, Daudjee K. Giraph unchained: Barrierless asynchronous parallel execution in pregel-like graph processing systems. Proc. of the VLDB Endowment, 2015,8(9):950−961.

[48] Gonzalez JE, Low Y, Gu H, Bickson D, Guestrin C. PowerGraph: Distributed graph-parallel computation on natural graphs. In: Proc. of the 10th USENIX Symp. on Operating Systems Design and Implementation. 2012. 17−30.

[49] Low Y, Gonzalez J, Kyrola A, Bickson D, Guestrin C, Hellerstein JM. GraphLab: A new framework for parallel machine learning. In: Proc. of the 26th Conf. on Uncertainty in Artificial Intelligence. 2010. 340−349.

[50] Zhang Q, Yan D, Cheng J. Quegel: A general-purpose system for querying big graphs. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2016. 2189−2192.

[51] Shao Y, Cui B, Ma L. PAGE: A partition aware engine for parallel graph computation. IEEE Trans. on Knowledge and Data Engineering, 2015,27(2):518−530.

[52] Shao B, Wang H, Li Y. Trinity: A distributed graph engine on a memory cloud. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2013. 505−516.

[53] Gonzalez JE, Xin RS, Dave A, Crankshaw D, Franklin MJ, Stoica I. GraphX: Graph processing in a distributed dataflow framework. In: Proc. of the 11th USENIX Symp. on Operating Systems Design and Implementation. 2014. 599−613.

[54] Valiant LG. A bridging model for parallel computation. Communications of the ACM, 1990,33(8):103−111.

[55] Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I. Spark: Cluster computing with working sets. In: Proc. of the 2nd USENIX Workshop on Hot Topics in Cloud Computing. 2010.

[56] Webber J. A programmatic introduction to Neo4J. In: Proc. of the Conf. on Systems, Programming, and Applications: Software for Humanity. 2012. 217−218.

[57] Aberger CR, Lamb A, Tu S, Nötzli A, Olukotun K, Ré C. EmptyHeaded: A relational engine for graph processing. ACM Trans. on Database Systems, 2017,42(4):20:1−20:44.

[58] Aberger CR, Tu S, Olukotun K, Ré C. EmptyHeaded: A relational engine for graph processing. In: Proc. of the SIGMOD Conf. 2016. 431−446.

[59] Peng P, Zou L, Özsu MT, *et al*. Processing SPARQL queries over distributed RDF graphs. The VLDB Journal, 2016,25(2): 243−268.

[60] Cypher. https://neo4j.com/docs/developer-manual/current/cypher/

[61] Ngo HQ, Porat E, Ré C, Rudra A. Worst-Case optimal join algorithms. Journal of the ACM, 2018,65(3):16:1−16:40.

[62] RDF. https://www.w3.org/RDF/

[63] SPARQL. https://www.w3.org/TR/rdf-sparql-query/

[64] Zou L, Huang R, Wang H, *et al*. Natural language question answering over RDF: A graph data driven approach. In: Proc. of the 2014 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2014. 313−324.

[65] http://www.omnicoreagency.com/twitter-statistics

[66] Arasu A, Babu S, Widom J. The CQL continuous query language: Semantic foundations and query execution. The VLDB Journal, 2006,15(2):121−142.

[67] Team S. StreamSQL: A data stream language extending SQL. 2017.

[68] Abadi DJ, Carney D, Çetintemel U, *et al*. Aurora: A new model and architecture for data stream management. The VLDB Journal, 2003,12(2):120−139.

[69] STREAM Group. STREAM: The Stanford Stream Data Manager. Stanford InfoLab, 2003.

[70] Chandrasekaran S, Cooper O, Deshpande A, *et al*. TelegraphCQ: Continuous dataflow processing. In: Proc. of the 2003 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2003. 668.

[71] Chen J, De Witt DJ, Tian F, *et al*. NiagaraCQ: A scalable continuous query system for Internet databases. ACM SIGMOD Record, 2000,29(2):379−390.

[72] Cranor C, Johnson T, Spataschek O, *et al*. Gigascope: A stream database for network applications. In: Proc. of the 2003 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2003. 647−651.

[73] http://storm.apache.org

[74] Toshniwal A, Taneja S, Shukla A, *et al*. Storm@Twitter. In: Proc. of the 2014 ACM SIGMOD Int'l Conf. on Management of Data. 2014. 147−156.

[75] http://spark.apache.org/streaming

[76] http://samza.apache.org

[77] Noghabi SA, Paramasivam K, Pan Y, *et al*. Samza: Stateful scalable stream processing at LinkedIn. Proc. of the VLDB Endowment, 2017,10(12):1634−1645.

[78] http://flink.apache.org

[79] Carbone P, Katsifodimos A, Ewen S, *et al*. Apache flink: Stream and batch processing in a single engine. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2015,36(4).

[80] http://kafka.apache.org

[81] Urban TL. Establishing delivery guarantee policies. European Journal of Operational Research, 2009,196(3):959−967.

[82] Weise T, Ramanath MV, Yan D, Knowles K. Learning Apache Apex: Real-Time Streaming Applications with Apex. Packt Publishing, 2017.

[83] http://spark.apache.org

[84] Zaharia M, Chowdhury M, Franklin MJ, *et al*. Spark: Cluster computing with working sets. HotCloud, 2010,10(10-10):95.

[85] http://lambda-architecture.net

[86] Hasani Z, Kon-Popovska M, Velinov G. Lambda architecture for real time big data analytic. In: Proc. of the ICT Innovations. 2014.

[87] http://rocksdb.org

[88] Kreps J, Narkhede N, Rao J. Kafka: A distributed messaging system for log processing. In: Proc. of the NetDB. 2011. 1−7.

[89] http://yahooeng.tumblr.com/post/135321837876/benchmarking-streaming-computation-engines-at

[90] McGregor A. Graph stream algorithms: A survey. ACM SIGMOD Record, 2014,43(1):9−20.

[91] Song C, Ge T, Chen C, *et al*. Event pattern matching over graph streams. Proc. of the VLDB Endowment, 2014,8(4):413−424.

[92] Qiu X, Cen W, Qian Z, *et al*. Real-Time constrained cycle detection in large dynamic graphs. Proc. of the VLDB Endowment, 2018, 11(12).

[93] Sutanay C, Lawrence BH, George CJ, Khushbu A, John F. A selectivity based approach to continuous pattern detection in streaming graphs. In: Proc. of the 18th Int'l Conf. on Extending Database Technology. 2015. 157−168.

[94] Samet H. Foundations of Multidimensional and Metric Data Structure. Morgan Kaufmann Publishers, 2006.

[95] Cong G, Jensen CS. Querying geo-textual data: Spatial keyword queries and beyond. In: Proc. of the SIGMOD. 2016. 2207−2212.

[96] Enderle J, Schneider N, Seidl T. Efficiently processing queries on interval-and-value tuples in relational databases. In: Proc. of the VLDB. 2005. 385−396.

[97] Dignös A, Böhlen MH, Gamper J, Jensen CS. Extending the kernel of a relational DBMS with comprehensive support for sequenced temporal queries. ACM Trans. on Database Systems, 2016,41(4):26:1−26:46.

[98] Güting RH, Schneider M. Moving Objects Databases. Morgan Kaufmann Publishers, 2005.

[99] Dinh L, Aref WG, Mokbel MF. Spatio-Temporal access methods: Part 2 (2003-2010). IEEE Data Engineering Bulletin, 2010, 33(2):46−55.

[100] Trajcevski G, Wolfson O, Hinrichs KH, Chamberlain S. Managing uncertainty in moving objects databases. ACM Trans. on Database Systems, 2004,29(3):463−507.

[101] Güting RH, de Almeida VT, Ding Z. Modeling and querying moving objects in networks. The VLDB Journal, 2006,15(2):165−190.

[102] Gao Y, Zheng B, Chen G, Chen C, Li Q. Continuous nearest-neighbor search in the presence of obstacles. ACM Trans. on Database Systems, 2011,36(2):9.

[103] Zhang C, Han J, Shou L, Lu J, La Porta TF. Splitter: Mining fine grained sequential patterns in semantic trajectories. PVLDB, 2014, 7(9):769−780.

[104] Zheng K, Su H. Go beyond raw trajectory data: Quality and semantics. IEEE Data Engineering Bulletin, 2015,38(2):27−34.

[105] Singh H, Bawa S. A survey of traditional and MapReduce based spatial query processing approaches. SIGMOD Record, 2017,46(2): 18−29.

[106] Xie D, Li F, Yao B, Li GL, Zhou L, Guo M. Simba: Efficient in memory spatial analytics. In: Proc. of the SIGMOD. 2016. 1071−1085.

[107] Shang Z, Li GL, Bao Z. DITA: Distributed in-memory trajectory analytics. In: Proc. of the SIGMOD. 2018. 725−740.

[108] Ding X, Chen L, Gao Y, Jensen CS, Bao H. UlTraMan: A unified platform for big trajectory data management and analytics. PVLDB, 2018,11(7):787−799.

[109] Güting RH, Behr T, Düntgen C. SECONDO: A platform for moving objects database research and for publishing and integrating research implementations. IEEE Data Engineering Bulletin, 2010,33(2):56−63.

[110] Law E, Ahn L. Human computation. In: Proc. of the Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

[111] Howe J. Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business. Crown Business, 2009.

[112] Tong YX, Chen L, Shahabi C. Spatial crowdsourcing: Challenges, techniques, and applications. Proc. of the VLDB Endowment, 2017,10(12):1988−1991.

[113] Tong YX, Yuan Y, Cheng YR, Chen L, Wang GR. Survey on spatiotemporal crowdsourced data management techniques. Ruan Jian Xue Bao/Journal of Software, 2017,28(1):35−58 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5140.htm [doi: 10.13328/j.cnki.jos.005140]

[114] Chittilappilly A, Chen L, Amer-Yahia S. A survey of general-purpose crowdsourcing techniques. IEEE Trans. on Knowledge and Data Engineering, 2016,28(9):2246−2266.

[115] Li GL, Wang J, Zheng Y, Franklin MJ. Crowdsourced data management: A survey. IEEE Trans. on Knowledge and Data Engineering, 2016,28(9):2296−2319.

[116] Feng JH, Li GL, Feng JH. A survey on crowdsourcing. Chinese Journal of Computers, 2015,38(9):1713−1726 (in Chinese with English abstract).

[117] Ho C, Vaughan J. Online task assignment in crowdsourcing markets. In: Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI 2012). Toronto, 2012. 45−51.

[118] Ho C, Jabbari S, Vaughan J. Adaptive task assignment for crowdsourced classification. In: Proc. of the 30th Int'l Conf. on Machine Learning (ICML 2013). Atlanta, 2013. 534−542.

[119] Tong YX, She JY, Ding B, Wang L, Chen L. Online mobile micro-task allocation in spatial crowdsourcing. In: Proc. of the 32nd Int'l Conf. on Data Engineering. 2016. 49−60.

[120] Tong YX, She JY, Ding B, *et al*. Online minimum matching in real-time spatial data: Experiments and analysis. Proc. of the VLDB Endowment, 2016,9(12):1053−1064.

[121] Song TS, Tong YX, Wang L, *et al*. Trichromatic online matching in real-time spatial crowdsourcing. In: Proc. of the 33rd Int'l Conf. on Data Engineering. 2017. 1009−1020.

[122] Tong YX, Wang L, Zhou Z, *et al*. Flexible online task assignment in real-time spatial data. Proc. of the VLDB Endowment, 2017, 10(11):1334−1345.

[123] Tong YX, Chen YQ, Zhou ZM, *et al*. The simpler the better: A unified approach to predicting original taxi demands based on large-scale online platforms. In: Proc. of the 23rd Int'l Conf. on Knowledge Discovery and Data Mining. 2017. 1653−1662.

[124] Tong YX, Zeng Y, Zhou ZM, *et al*. A unified approach to route planning for shared mobility. Proc. of the VLDB Endowment, 2018, 11(11):1633−1646.

[125] Whitehill J, Ruvolo P, Wu T, Bergsma J, Movellan J. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In: Proc. of the 23rd Annual Conf. on Neural Information Processing Systems (NIPS). British Columbia, 2018. 2035−2043.

[126] Raykar V, Yu S. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. Journal of Machine Learning Research, 2012,13:491−518.

[127] Sheng V, Provost F, Ipeirotis P. Get another label? Improving data quality and data mining using multiple, noisy labelers. In: Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD 2008). Las Vegas, 2018. 614−622.

[128] Dalvi N, Dasgupta A, Kumar R, Rastogi V. Aggregating crowdsourced binary ratings. In: Proc. of the 22nd Int'l World Wide Web Conf. (WWW 2013). Rio de Janeiro, 2013. 285−294.

[129] Karger D, Oh S, Shah D. Efficient crowdsourcing for multi-class labeling. In: Proc. of the ACM SIGMETRICS Int'l Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS 2013). Pittsburgh, 2013. 81−92.

[130] Joglekar M, Garcia-Molina H, Parameswaran A. Evaluating the crowd with confidence. In: Proc. of the 19th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (SIGKDD 2013). Chicago, 2013. 686−694.

[131] Liu X, Lu M, Ooi B, Shen Y, Wu S, Zhang M. CDAS: A crowdsourcing data analytics system. Proc. of the VLDB Endowment (PVLDB), 2012,5(11):1495−1506.

[132] Cao CC, She JY, Tong YX, Chen L. Whom to ask? Jury selection for decision making tasks on micro-blog services. Proc. of the VLDB Endowment (PVLDB), 2012,5(11):1495−1506.

[133] Cao CC, Tong YX, Chen L, Jagadish H. WiseMarket: A new paradigm for managing wisdom of online social users. In: Proc. of the 19th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (SIGKDD 2013). Chicago, 2013. 455−463.

[134] Singer Y, Mittal M. Pricing mechanisms for crowdsourcing markets. In: Proc. of the 22nd Int'l World Wide Web Conf. (WWW 2013). Rio de Janeiro, 2013. 1157−1166.

[135] Singla A, Krause A. Truthful incentives in crowdsourcing tasks using regret minimization mechanisms. In: Proc. of the 22nd Int'l World Wide Web Conf. (WWW 2013). Rio de Janeiro, 2013. 1167−1178.

[136] Gao Y, Parameswaran A. Finish them! Pricing algorithms for human computation. Proc. of the VLDB Endowment (PVLDB), 2014, 7(14):1965−1976.

[137] Tong YX, Chen L, Zhou ZM, *et al*. SLADE: A smart large-scale task decomposer in crowdsourcing. IEEE Trans. on Knowledge and Data Engineering, 2018. [doi: 10.1109/TKDE.2018.2797962]

[138] Tong YX, Wang L, Zhou ZM, *et al*. Dynamic pricing in spatial crowdsourcing: A matching-based approach. In: Proc. of the Int'l Conf. on Management of Data. 2018. 773−788.

[139] Parameswaran A, Garcia-Molina H, Park H, Polyzotis N, Ramesh A, Widom J. Crowdscreen: Algorithms for filtering data with humans. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2012). Scottsdale, 2012. 361−372.

[140] Parameswaran A, Boyd S, Garcia-Molina H, Gupta A, Polyzotis N, Widom J. Optimal crowd-powered rating and filtering algorithms. Proc. of the VLDB Endowment (PVLDB), 2014,7(9):685−696.

[141] Guo S, Parameswaran A, Garcia-Molina H. So who won? Dynamic max discovery with the crowd. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2012). Scottsdale, 2012. 385−396.

[142] Davidson S, Khanna S, Milo T, Roy S. Using the crowd for top-*k* and group-by queries. In: Proc. of the 16th Int'l Conf. on Database Theory (ICDT 2013). Genoa, 2013. 225−236.

[143] Li KY, Zhang XH, Li GL. A rating-ranking method for crowdsourced top-*k* computation. In: Proc. of the Int'l Conf. on Management of Data. 2018. 975−990.

[144] Wang J, Kraska T, Franklin MJ, Feng J. CrowdER: Crowdsourcing entity resolution. Proc. of the VLDB Endowment (PVLDB), 2012,5(11):1483−1494.

[145] Wang J, Li G, Kraska T, Franklin MJ, Feng J. Leveraging transitive relations for crowdsourced joins. In: Proc. of the 2013 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2013). New York, 2013. 229−240.

[146] Vesdapunt N, Bellare K, Dalvi N. Crowdsourcing algorithms for entity resolution. Proc. of the VLDB Endowment (PVLDB), 2014, 7(12):1071−1082.

[147] Zhang C, Chen L, Jagadish H, Cao C. Reducing uncertainty of schema matching via crowdsourcing. Proc. of the VLDB Endowment (PVLDB), 2013,6(9):757−768.

[148] Fan J, Lu M, Ooi B, Tan W, Zhang M. A hybrid machine-crowdsourcing system for matching Web tables. In: Proc. of the 30th Int'l Conf. on Data Engineering (ICDE 2014). Chicago, 2014. 976−987.

[149] Marcus A, Karger D, Madden S, Miller R, Oh S. Counting with the crowd. Proc. of the VLDB Endowment (PVLDB), 2012,6(2): 109−120.

[150] Trushkowsky B, Kraska T, Franklin MJ, Sarkar P. Crowdsourced enumeration queries. In: Proc. of the 29th Int'l Conf. on Data Engineering (ICDE 2013). Brisbane, 2013. 673−684.

[151] Amsterdamer Y, Grossman Y, Milo T, Senellart P. Crowd mining. In: Proc. of the 2013 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2013). New York, 2013. 241−252.

[152] Gomes R, Welinder P, Krause A, Perona P. Crowdclustering. In: Proc. of the 25th Annual Conf. on Neural Information Processing Systems (NIPS 2011). Granada, 2011. 558−566.

[153] Franklin MJ, Kossmann D, Kraska T, Ramesh S, Xin R. CrowdDB: Answering queries with crowdsourcing. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. 2011. 61−72.

[154] Park H, Pang R, Parameswaran AG, Garcia-Molina H, Polyzotis N, Widom J. Deco: A system for declarative crowdsourcing. Proc. of the VLDB Endowment, 2012,5(12):1990−1993.

[155] Marcus A, Wu E, Karger DR, Madden S, Miller RC. Demonstration of qurk: A query processor for humanoperators. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. 2011. 1315−1318.

[156] Li GL, Yuan HT, Chai C, *et al*. CDB: Optimizing queries with crowd-based selections and joins. In: Proc. of the ACM Int'l Conf. on Management of Data. 2017. 1463−1478.

[157] Zheng Y, Li G, Cheng CK. DOCS: Domain-Aware crowdsourcing system. Proc. of the VLDB Endowment, 2016,10(4):361−372.

[158] Chen Z, Fu R, Zhao Z, *et al*. gMission: A general spatial crowdsourcing platform. Proc. of the VLDB Endowment, 2014,7(13): 1629−1632.

[159] Li G, Zheng Y, Fan J, Wang J, Cheng R. Crowdsourced data management: Overview and challenges. In: Proc. of the 2017 ACM Int'l Conf. on Management of Data. ACM Press, 2017. 1711−1716.

[160] Balkesen C, Alonso G, Teubner J, Özsu MT. Multi-Core, main-memory joins: Sort vs. Hash revisited. PVLDB, 2013,7(1):85−96.

[161] Stehle E, Jacobsen HA. A memory bandwidth-efficient hybrid radix sort on GPUs. In: Proc. of the SIGMOD Conf. 2017. 417−432.

[162] Kara K, Giceva J, Alonso G. FPGA-Based data partitioning. In: Proc. of the SIGMOD Conf. 2017. 433−445.

[163] Shahvarani A, Jacobsen HA. A hybrid B+-tree as solution for in-memory indexing on CPU-GPU heterogeneous computing platforms. In: Proc. of the SIGMOD Conf. 2016. 1523−1538.

[164] Barthels C, Loesing S, Alonso G, Kossmann D. Rack-Scale in-memory join processing using RDMA. In: Proc. of the SIGMOD Conf. 2015. 1463−1475.

[165] Yoon DY, Chowdhury M, Mozafari B. Distributed lock management with RDMA: Decentralization without starvation. In: Proc. of the SIGMOD Conf. 2018. 1571−1586.

[166] van Renen A, Leis V, Kemper A, Neumann T, Hashida T, Oe K, Doi Y, Harada L, Sato M. Managing non-volatile memory in database systems. In: Proc. of the SIGMOD Conf. 2018. 1541−1555.

[167] Wang TZ, Johnson R. Scalable logging through emerging non-volatile memory. PVLDB, 2014,7(10):865−876.

[168] Larson PÅ, Levandoski JJ. Modern main-memory database systems. PVLDB, 2016,9(13):1609−1610.

[169] Diaconu C, Freedman C, Ismert E, Larson PÅ, Mittal P, Stonecipher R, Verma N, Zwilling M. Hekaton: SQL server's memory-optimized OLTP engine. In: Proc. of the SIGMOD Conf. 2013. 1243−1254.

[170] Rao J, Ross KA. Making B+-trees cache conscious in main memory. In: Proc. of the SIGMOD Conf. 2000. 475−486.

[171] Levandoski JJ, Lomet DB, Sengupta S. The Bw-tree: A B-tree for new hardware platforms. In: Proc. of the ICDE. 2013. 302−313.

[172] Larson PÅ, Blanas S, Diaconu C, Freedman C, Patel JM, Zwilling M. High-Performance concurrency control mechanisms for main-memory databases. PVLDB, 2011,5(4):298−309.

[173] Ren K, Thomson A, Abadi DJ. Lightweight locking for main memory database systems. PVLDB, 2012,6(2):145−156.

[174] Chaudhuri S, Ding BL, Kandula S. Approximate query processing: No silver bullet. In: Proc. of the SIGMOD Conf. 2017. 511−519.

[175] Garofalakis MN, Gibbons PB. Approximate query processing: Taming the TeraBytes. In: Proc. of the VLDB. 2001.

[176] Li FF, Wu B, Yi K, Zhao ZY. Wander join: Online aggregation via random walks. In: Proc. of the SIGMOD Conf. 2016. 615−629.

[177] Agarwal S, Panda A, Mozafari B, Iyer AP, Madden S, Stoica I. Blink and it's done: Interactive queries on very large data. PVLDB, 2012,5(12):1902−1905.

[178] Ren L, Du Y, Ma S, Zhang XL, Dai GZ. Visual analytics towards big data. Ruan Jian Xue Bao/Journal of Software, 2014,25(9): 1909−1936 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4645.htm [doi: 10.13328/j.cnki.jos.004645]

[179] Herman I, Melancon G, Marshall MS. Graph visualization and navigation in information visualization: A survey. IEEE Trans. on Visualization and Computer Graphics, 2000,6(1):24

[180] Tobler W. Experiments in migration mapping by computer. The American Cartographer, 1987,14(2):155−163.

[181] Keim DA, Kriegel HP. Visualization techniques for mining large databases: A comparison. IEEE Trans. on Knowledge and Data Engineering, 1996,8(6):923−938.

附中文参考文献:

[113] 童咏昕,袁野,成雨蓉,陈雷,王国仁.时空众包数据管理技术研究综述.软件学报,2017,28(1):35−58. http://www.jos.org.cn/1000-9825/5140.htm [doi: 10.13328/j.cnki.jos.005140]

[116] 冯剑红,李国良,冯建华.众包技术研究综述.计算机学报,2015,38(9):1713−1726.

[178] 任磊,杜一,马帅,张小龙,戴国忠.大数据可视分析综述.软件学报,2014,25(9):1909−1936. http://www.jos.org.cn/1000-9825/4645.htm [doi: 10.13328/j.cnki.jos.004645]

崔斌(1975－),男,浙江宁波人,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为数据库系统,大数据管理和分析.

高军(1975－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为分布式数据管理,图数据管理和深度分析.

童咏昕(1982－),男,博士,副教授,CCF 专业会员,主要研究领域为众包数据管理,群体智能,时空数据管理与挖掘,不确定数据管理与挖掘.

许建秋(1982－),男,博士,副教授,CCF 专业会员,主要研究领域为空间,移动对象数据管理.

张东祥(1985－),男,博士,教授,CCF 专业会员,主要研究领域为时空大数据分析,智能交通优化.

邹磊(1981－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为图数据库系统,知识图谱分析.