云数据管理系统中查询技术研究综述

史英杰 孟小峰
(中国人民大学信息学院 北京 100872)

摘 要 作为一种全新的互联网应用模式,云计算在工业界和学术界备受关注.人们可以通过终端设备便捷地获取云端服务,并以按需使用的方式获得存储资源、计算资源以及软硬件资源.云计算的发展带来了一系列挑战性问题,而云数据的管理问题首当其冲.文中结合云数据的特点提出了一个云数据管理系统的框架,并在此基础上从索引管理、查询处理、查询优化以及在线聚集等几个方面对云数据管理系统中查询技术的研究工作进行了总结分析,指明了该领域面临的挑战和未来的研究工作.

关键词 云计算;云数据管理;查询处理;查询优化;索引管理;在线聚集
中图法分类号 TP392 **DOI 号** 10.3724/SP.J.1016.2013.00209

A Survey of Query Techniques in Cloud Data Management Systems

SHI Ying-Jie MENG Xiao-Feng
(*School of Information, Renmin University of China, Beijing* 100872)

Abstract As a revolutionary application mode in the internet, cloud computing has attracted more and more attentions from both industry and academia. Users can obtain cloud service conveniently through terminals, and access resources of storage, computing and hardware in the Pay-As-You-Go model. The development of cloud computing brings about a series of challenging problems, data management in the cloud is of great importance. In this paper, we propose a framework of cloud data management system. Based on this framework, the key research works of query techniques in cloud data management system are classified and surveyed from several aspects: index management, query processing, query optimization and online aggregation. At last, the suggestions for future research are put forward.

Keywords cloud computing; cloud data management; query processing; query optimization; index management; online aggregation

1 引 言

云计算是当今信息产业备受关注的一种全新领先的计算模式.在云计算模式下,企业和个人可以根据自己的需要购买存储设备和计算能力,而不用花费大量资金购买大规模高性能计算机,这使得用户在软硬件维护以及升级上的成本投入大大减少.作为一项有望大幅降低成本的新兴技术,云计算受到了众多信息业巨头的关注:Amazon、Google、IBM、微软等公司都对云计算的研发进行了大规模的投入.与此同时,云计算的发展也产生了一系列新的挑战性问题,云数据管理是亟待解决的问题之一.

随着信息产业的发展,企业和各种组织产生的

收稿日期:2012-02-28;最终修改稿收到日期:2012-08-21.本课题得到国家自然科学基金(61070055,91024032,91124001)、中国人民大学科学研究基金(11XNL010)、国家"八六三"高技术研究发展计划项目基金(2012AA010701)资助.**史英杰**,女,1983 年生,博士研究生,主要研究方向为云数据管理、大数据上的在线聚集.E-mail:shiyingjie1983@yahoo.com.cn.**孟小峰**,男,1964 年生,博士,教授,主要研究领域为 Web 数据管理、移动数据管理、XML 数据管理、云数据管理等.

数据量快速增长.据 IDC(互联网数据中心)统计,2011 年全球产生的数据量达到 1.8 ZB,比 2010 年增长了 1ZB①.如何对这些海量数据进行有效管理和分析以获取数据背后潜在的巨大价值,是目前互联网、通信和生物医学等诸多领域面临的问题.传统数据管理系统中的很多技术对于如此大规模的数据管理往往不再有效,而且相关软硬件以及维护的昂贵成本也是让大部分企业望洋兴叹.云环境是由大量性能普通、价格便宜的计算节点组成的一种无共享大规模并行处理环境[1],所以从成本和性能两方面考虑,越来越多的组织更愿意把数据中心从昂贵的高性能计算集群转移到公有云或私有云环境中.

另一方面,随着 Web2.0 和普适计算应用的流行,其"瘦客户端+服务"的运行模式对服务器端的计算能力和数据处理能力要求越来越高.在这种模式下,用户通过浏览器就可获得各种各样的服务,所有的计算都交由服务器端执行.为支持这些应用,服务系统需要存储、索引和备份海量的异构万维网页面、用户访问日志以及用户信息,并且还要保证对这些数据进行快速准确的访问[2].同时,服务系统所要处理的数据不仅包括产品和客户信息等结构化数据,还包含大量半结构和非结构化数据.在上述应用需求的推动下,云数据管理系统应运而生.

目前,随着 Google、Yahoo!、Facebook 等企业的推动,出现了不少基于云计算平台的数据管理系统,而且大部分系统已经投入生产环境使用[3-4].与传统数据库系统相比,目前云数据管理系统提供的接口有很多限制,只提供简单的数据存取接口或者极小化的查询语言,这增加了用户使用的难度,也增加了开发人员的负担.同时,相比于传统的分布式关系数据库,云数据管理系统的查询性能也有很大的提升空间[5-6].如何在现有云计算平台的基础上,完善云数据管理系统的查询功能并提高其数据处理的性能,是目前备受关注的挑战性问题.

本文第 2 节对云数据查询技术进行概述;第 3 节提出云数据管理系统的基本框架并依据该框架对云数据查询的关键性技术进行总结分析;第 4 和第 5 节对未来工作进行展望并对全文工作进行总结.

2 云数据查询处理概述

与传统关系数据库中的数据查询相比,云数据管理系统中的查询处理特点鲜明.本节阐述云数据管理系统的应用场景,结合已有的云数据管理系统和相关研究工作,对云环境中数据的特性进行了分析,指出云数据查询处理技术的目标,并总结云数据管理系统中查询技术的特征与面临的挑战.

2.1 云数据管理系统的应用场景

与传统的关系数据库相比,云数据管理系统具有良好的扩展性和容错性,利用云计算平台中大规模计算资源和存储资源管理海量异构数据,为用户提供高性价比的数据管理方式.目前云数据管理系统在实际生产环境中得到了广泛的应用,主要集中在两个方面:海量数据分析和大规模 Web 数据管理.

数据分析主要用于生成报表、数据挖掘和决策支持等.与事务型数据处理不同,在分析型的数据处理中,数据是一次写多次读的,更新操作较少.数据分析可以在并行数据库上完成,但是随着数据规模的扩大以及对性能要求的提高,并行数据库系统的维护需耗费大量的资金及人力.云数据管理系统在扩展性和性价比上均占有天然的优势,其中类 BigTable 系统[7](BigTable、HBase②、Hypertable③)、HadoopDB[8] 和 Hive[9] 等支持 MapReduce 框架的系统是面向数据分析型应用的.

随着 Web2.0 技术的发展,超大规模和高并发的社交网站逐渐兴起,参与人数迅速攀升.以微博网站 Twitter 为例,2010 年 2 月用户每日发送的微博数量是 5 千万,而到了 2011 年 3 月用户每日发送的微博数量达到 1 亿 4 千万④,用户和网站交互产生大量动态信息.这种海量 Web 数据管理应用要求数据库能够满足高并发的数据读写和高效实时的数据访问,同时要求数据库具备可扩展性以应付数据的不断快速增长.关系数据库在这些需求面前显得力不从心,云数据管理系统则以灵活的扩展性和高性能的数据读写受到 Web2.0 网站的青睐,其中 Cassandra⑤、CouchDB⑥ 和 PNUTS[4] 等系统广泛应用在 Facebook、Twitter 和 Yahoo!等大型网站中.

2.2 云数据的特点

云计算将大量用网络连接的计算资源进行统一管理和调度,以服务的方式为用户提供计算资源、存储资源和软硬件资源,其最鲜明的特点是可扩展性、高可用性和按需服务性.云计算环境中存储和管理的数据具备如下特点[1,8,10-11]:

(1)海量性.随着移动设备的普及、传感器技术的发展以及社交网络的扩大,云计算平台存储和管

① http://www.emc.com/collateral/demos/microsites/emc-digital-universe-2011/index.htm
② http://hbase.apache.org/
③ http://hypertable.org/
④ http://blog.twitter.com/2011/03/numbers
⑤ http://cassandra.apache.org
⑥ http://couchdb.apache.org/

理的数据量十分庞大，TB 级别和 PB 级别的数据规模十分常见.

(2)种类多样性. 随着 Web2.0 的兴起，互联网应用不断推陈出新. 一些新兴应用领域(微博、社交网络等)所处理的数据除了传统数据库里的结构化数据，还包括半结构化数据和非结构化数据，使得云计算平台中的数据种类纷繁多样.

(3)异地备份. 数据的高可用性是云计算的重要特征之一，而这种面临软硬件错误的高水平容错性是通过对用户透明的数据异地备份实现的.

云数据的特征导致了传统的关系数据库无法满足其多样化的应用需求. 云数据管理系统必须提供灵活的数据模型以有效管理多样化的数据，并针对数据分布和冗余的特性设计相应的存储方式和查询优化策略，从而向用户提供“按需所取”、可靠的、高性能的数据存取与查询服务.

2.3　云数据查询处理的目标

为了提供高效可靠的云数据管理服务，云数据的查询处理技术需要达到以下目标[1,11-14]：

(1)可扩展性. 云平台的规模大小不一，小的私有云平台规模为十几个节点，大的公有云平台规模可达到几千个节点①[15]. 此外，云计算提供的是一种“按需计费”的服务方式，随着应用需求的变化，云平台的规模也会发生变化. 这就要求云数据管理系统中的查询处理及优化算法具备良好的扩展性，不仅能够扩展到庞大规模的云平台上，而且能够实现资源的可动态增长及其带来的性能提升.

(2)可用性. 云平台由大量廉价计算机构成，与高性能服务器构成的分布式系统相比，云平台的硬件出错率较高. 云数据管理系统需要将软硬件错误看成系统运行的常态，错误发生时既要保证数据不丢失，又要保证数据的读写操作能够正常进行.

(3)在异构环境运行的能力. 随着应用的发展以及数据量的不断增长，云平台势必要通过增加新的节点来提高计算和存储能力. 因此，保证一个云平台中所有节点的硬件配置同构是非常困难的. 即使在一个硬件配置相同的环境中，不同节点的软硬件性能也会出现波动[16]. 云数据的查询技术要有在异构环境运行的能力，从而避免性能较差的节点影响整个系统的运行效率这种“木桶效应”的出现.

(4)丰富灵活的用户接口. 一方面，云数据管理系统要提供 SQL 接口，这样习惯于关系数据库查询语言的用户不必重新学习新的接口或者编程方法，而原来基于关系数据库的各种应用也可以平滑的转移到云上；另一方面，云数据管理系统还要提供 UDF(User Defined Function)接口，用户可以根据业务需求自己定义数据查询操作.

(5)高效的数据存取性能. 云数据管理系统的软硬件成本远远低于高性能分布式数据库，其处理海量数据的效率也是云计算用户关注的重要问题. 云数据管理系统应当针对云数据的特点设计数据分布策略和查询优化相关算法，从而提高其管理海量数据的能力.

云数据管理系统可以通过云计算平台的资源虚拟以及 MapReduce[15] 框架的使用而得到良好的扩展性和可用性，也可以在并行任务调度过程中采取投机任务(speculative task)[16] 等措施保证其在异构环境中运行的能力. 从支持的查询接口看，目前大部分云数据管理系统只提供了简单的数据存取接口或者极小化的查询语言，这限制了其对复杂数据查询和分析的支持. 从查询性能来看，目前云数据管理系统的查询优化主要针对键值进行，而对非键值的查询主要是依靠批量的全表扫描. 因此，用户接口和查询性能是目前云数据管理系统亟待提高的两个方面.

2.4　云数据管理系统中查询处理的特征

传统关系数据库中的查询技术无法同时满足上节提到的目标，特别是可扩展性和可用性. 现有的云数据管理系统的查询技术和传统关系数据库系统的查询技术在处理的数据类型、容错性和支持接口等方面表现出明显差异，表 1 从多个方面对二者进行了对比.

表 1　云数据管理系统与关系数据库的查询技术比较

	云数据管理系统的特点	关系数据库的特点
数据类型	结构化数据，半结构化数据，非结构化数据	结构化数据
数据模型	Key-Value 模型，文档模型，简化的关系模型	关系模型
扩展性	易大规模扩展(几百到几千节点)	不易大规模扩展(最多几百节点)
容错性	数据容错，查询容错	数据容错
服务方式	Pay-As-You-Go	Pay-Before-You-Go
支持接口	简单的 API 和极小化的查询语言，亟待丰富	复杂的 SQL 语言
查询优化技术	基于键值和基于规则的优化技术，亟待相关研究成果	基于规则和基于代价的优化技术，技术比较成熟

① http://hadoop.apache.org/

传统关系数据库的查询主要面向结构化数据，其数据模型基于关系模型.云数据管理系统处理的数据对象除了结构化数据，还包括半结构化和非结构化数据，其数据模型包括 key-value 模型、文档模型和简化的关系模型[3-4,9].之所以称其为简化的数据模型是因为它虽然以表的形式管理数据，但不提供实体完整性和参照完整性.除此以外，关系数据库的数据模型是一种模式优先(schema-first)的逻辑结构，即在数据入库之前设计好数据模式.而云数据管理系统中的数据模型是从数据到模式(from-data-to-schema)数据模式可以是松散的、滞后的，可以在数据入库时根据数据内容定义数据模式.

查询容错是指一个查询运行过程中出现了硬件错误，该查询不必重新开始.传统的关系数据库系统一般不保证查询容错.云数据管理系统把硬件错误看成一种常态，它同时保证数据容错和查询容错.因为云平台上硬件错误率较高，如果每次出现错误都需要重启查询，那么一个耗时较长的查询很可能无法完成.从服务方式来看，传统关系数据库是一种 pay-before-you-go 的方式，即通过需求分析设计数据库模式并构建数据库软硬件，并在较长时间内保持相对稳定，因此查询优化的目标是在已有的软硬件环境下获得最好的查询性能.而云数据管理系统是一种 pay-as-you-go 的方式，用户根据使用的计算资源和存储资源向服务提供商付费，因而查询优化的目标是如何利用更少的计算资源获得用户期望的查询性能.从查询接口和查询优化技术来看，关系数据库支持复杂的 SQL 语言，而且查询优化技术也非常成熟.相比之下，现有的云数据管理系统支持的查询语言比较匮乏，而且已有的查询优化技术主要集中在基于规则的优化，因此在这两个方面亟待加强.

3 云数据管理系统中查询技术研究

作为一种新型数据管理技术，云数据管理系统的研究仍处于起步阶段.这种新兴的数据管理技术可以扩展到大量廉价节点上，为用户提供按需所取、高性价比的数据管理服务.本节首先提出云数据管理系统的整体框架，然后从数据存储与索引技术、查询处理及优化、在线聚集几个方面对云数据查询相关工作和研究成果进行分析总结.

3.1 云数据管理系统基本框架

为了有效管理海量、种类多样的云数据，并提供“按需所取”的云服务，云数据管理系统必须具有可扩展性、可裁剪性、可用性以及在异构环境中运行的能力.这使得云数据管理系统在面临查询处理、查询优化和索引管理等问题时采用不同于传统数据库的全新解决方法.同时，一些在传统数据库中提出但是没有得到广泛应用的研究问题在云环境下显现出重要的意义，例如查询进程估计和在线聚集等.目前已有的数据管理系统大都面向某一类特定应用，因此系统架构和实现方式各有不同.我们结合云计算中数据管理应用的特点以及数据查询处理的目标，提出了云数据管理系统的整体架构，如图 1 所示，该架构被划分为 5 个部分.

(1)应用接口层.负责接收用户提交的请求并交给查询处理层相应的模块进行处理.提供查询语言接口、用户自定义接口 UDF(key/value 操作)、数据分析和在线聚集等应用.用户不仅可以通过查询接口和 UDF 接口进行数据操作，还可以通过可视化工具执行数据分析和在线聚集.

(2)查询处理层.对上层提交的查询语句进行解析和逻辑优化后转化成操作符树，进而生成 MapReduce 执行计划；如果上层提交的是用户自定义操作，则直接生成 MapReduce 执行计划.如何根据查询类型和数据分布等信息生成合适的查询计划，以及如何利用云数据的特点对查询计划进行逻辑优化是查询处理层的主要任务，也是云数据管理领域备受关注的研究问题.

(3)数据控制层.该层主要负责 3 个方面的工作：利用全局索引和元数据信息进行数据定位；备份数据的一致性处理和数据迁移；在线聚集过程中进行数据采样和进程估计.数据层涉及到查询执行和在线聚集的核心部分，目前的研究工作主要围绕查询处理优化、索引构建、数据采样和查询结果估计.

(4)数据存储层.负责数据的实际存储以及在各节点范围内数据的索引设计、缓冲区管理和日志管理.存储层的节点可通过多种方式组织，例如主-从结构或者点对点结构等，主要通过不同的通信协议体现.无论采用哪种结构，数据都被分区到多个节点存储.如何在保证数据分布均衡的情况下提高每个节点上数据存取的效率是存储层必须解决的问题.

(5)服务管理模块.负责元数据的管理、操作管理和系统监控.元数据管理部分为查询处理层提供访问接口，同时保证元数据与数据模式之间的一致性.操作管理主要面向数据控制层，包括数据读写锁机制、容错机制以及负载均衡.系统监控模块从数据

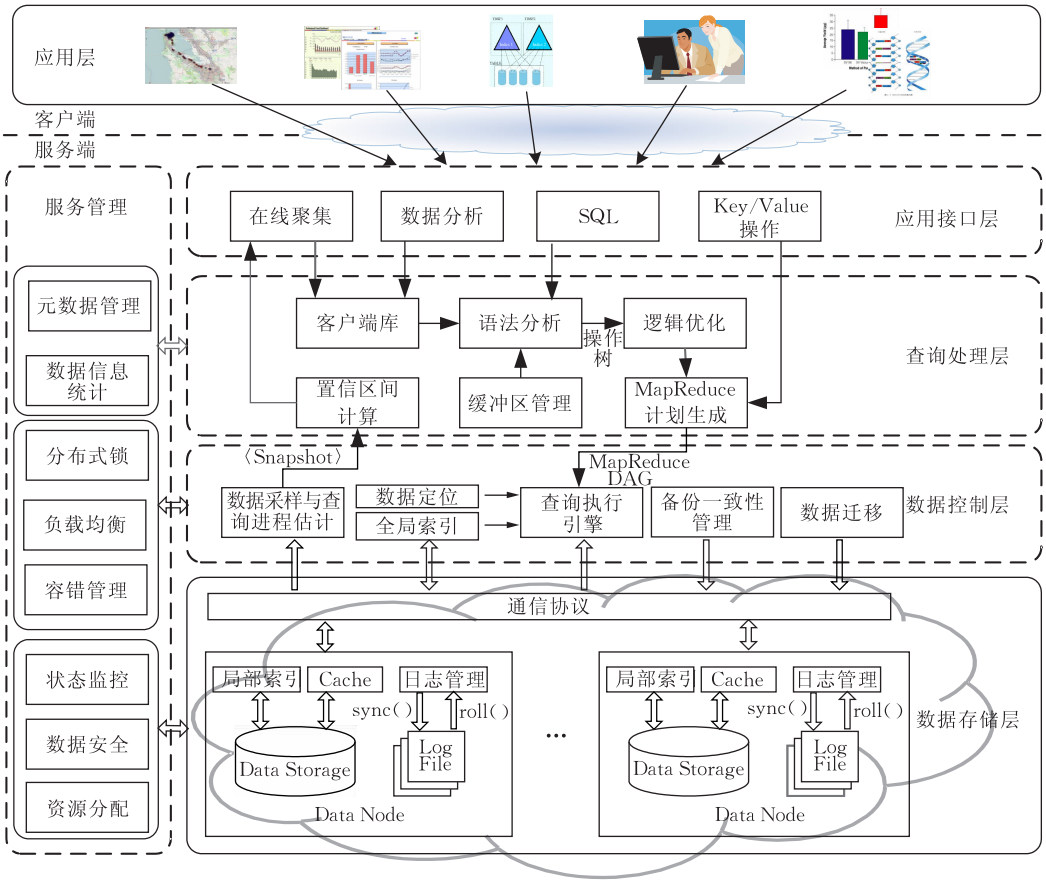

图 1　云数据管理系统框架

存储层收集监控信息，并通过图形界面将其展示给用户. 资源分配模块负责管理系统中的负载，节点能够被动态地添加或删除以适应工作负载的变化.

3.2　云数据管理系统关键技术研究

依据云数据管理系统的整体框架，可以看出云数据的查询领域存在许多研究问题：数据存储与索引设计、基于 MapReduce 的查询处理、查询优化、在线聚集过程中的数据采样与置信区间计算等. 目前索引管理、查询处理、查询优化以及在线聚集等问题已经得到了初步的研究，本节对目前已有的相关工作进行分析总结.

3.2.1　索引技术

现有的云数据管理系统大都以 key-value 方式存储数据，能够提供基于键值的快速查询，但是对于非键值的查询只能通过全表扫描来完成. 尽管可以通过 MapReduce 实现并发扫描，但是面对海量数据，对于选择度比较高的查询来说，全表扫描的效率仍然比较低. 目前很多学者对云数据管理系统中的索引技术进行了研究. 根据索引的实现方式，本文把已有的索引分成 3 类：双层索引[17-21]、二级索引①②[22]和基于线性化技术的全局索引[23].

(1) 双层索引

云数据管理系统中的双层索引框架由 Wu 等人[17]在 2009 年提出，后续双层索引方案的研究工作大都基于该框架，其结构如图 2 所示. 索引由局部索引和全局索引两部分构成. 为每个节点的数据建立局部索引，该索引只负责本地节点上的数据. 除局部索引外，每个计算节点还要共享一部分存储空间来存储全局索引. 全局索引依据局部索引构建，由于

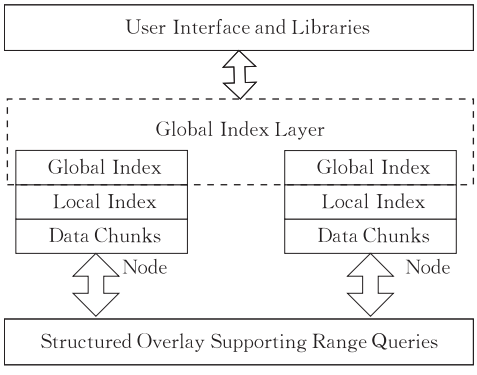

图 2　云数据管理系统中的双层索引框架

① https://github.com/hbase-trx/
② http://github.com/gkulbak/ibase

存储空间的限制和查询效率的要求，并不是所有的局部索引都发布到全局索引中，而是按照一定的规则对索引节点进行选择.

根据全局索引的组织方式，双层索引可以分成两类：P2P 结构的双层索引和集中式结构的双层索引. 如表 2 所示，前 3 种方案的全局索引均采用 P2P 结构的覆盖网络[17-19]，这种方式易于实现可扩展性，使系统能够同时支持大规模的查询. 但是也存在一些不足：首先，维护 P2P 网络需要一定的代价，查询时往往需要较高的网络传输代价；其次，对于主从结构(master-slave)的云数据管理系统，实现这种索引要重新构建一个 P2P 网络，会增加原有系统的负担. 基于上述原因，文献[20-21]在全局索引中采用了集中式的索引方式. EMINC[20] 在每个节点建立 KD 树作为局部索引，其中每个索引节点被看成一个多维度的立方体，全局索引利用 R 树对这些立方体进行索引. 当索引维度比较高，或者索引数据量比较大时，R 树各个节点之间的重叠部分较多，查询时会产生大量的误判(false positive)结果. 为解决这一问题，文献[21]的全局索引采用带 bloom filter 的 R 树. 进行查询时，首先通过 bloom filter 来验证，如果查询点不在其中，则不再进行 R 树查询. 这样减少了误判的几率，从而提高查询效率. 上述各种索引技术方案具有较好的扩展性，但总的来说实现过程比较复杂，索引更新维护的代价比较高. 特别是对于数据更新比较频繁的应用，对系统性能的影响较大.

表 2 云数据管理索引技术对比

索引方案		代表技术	索引构成	特点
双层索引	P2P	Efficient B-tree[17]	局部索引：B^+-tree 全局索引：BATON	优点：具有较好的扩展性 缺点：索引实现较复杂；更新维护代价较高
		RT-CAN[18]	局部索引：R-tree 全局索引：CAN	
		QT-Chord[19]	局部索引：IMX-CIF quad-tree 全局索引：Chord	
	集中式	EMINC[20]	局部索引：KD-tree 全局索引：R-tree	
		A-Tree[21]	局部索引：R-tree＋Bloom filter 全局索引：Array Based	
二级索引		ITHBase①	索引项内容：索引列信息＋rowkey	优点：实现简单；维护代价较低 缺点：多维查询效率较低；空间冗余大
		IHBase②	索引项内容：索引列信息＋rowkey	
		CCIndex[32]	索引项内容：索引列信息＋rowkey＋数据信息	
基于线性化技术的全局索引		MD-HBase[33]	索引项内容：最长公共前缀＋Z-value 值	优点：写入吞吐量较高；维护代价较低 缺点：数据一致性维护复杂

(2) 二级索引

二级索引(secondary index)方案主要应用于 key-value 存储的云数据库管理系统中，如 Bigtbale、HBase 等. 在这类系统中，针对非键值列的二级索引通过为索引列构建索引表实现. 索引表中的键值由原数据表中键值和索引列的组合构成，实现索引列与原有键值的映射. 查询过程中，首先根据查询条件在索引表找到相应键值的列表，然后根据这些键值到原数据表中定位所需数据. 目前基于二级索引的实现方案主要有 ITHBase①、IHBase② 和 CCIndex[22]. 其中 ITHBase 和 IHBase 均是开源的实现方案，二者实现方式相似，都从 HBase 源码级别进行扩展，重新定义和实现了客户端和服务端的处理逻辑，具有强侵入性. 与 IHBase 相比，ITHbase 更关注数据一致性，其重要特性之一是事务性.

ITHBase 和 IHBase 两种方案中的索引表仅存放索引列与原表的键值信息. 在查询过程中，先通过查询索引表得到键值，再根据键值到原表查找数据. 由于得到的键值大都是随机的，所以需要进行大量的随机查找才能得到最终的查询结果，效率较低. 为了减少随机查询带来的开销，Zou 等人提出了另外一种二级索引方案：互补聚簇式索引(Complemental Clustering Index)，简称 CCIndex[22]. CCIndex 把数据的详细信息也存放在索引表中，查询时可以直接在索引表中通过顺序扫描找到相应的数据，从而大大减少查询时间. 然而把详细信息存储在索引表中会造成存储空间的增加. 为了尽可能地减少存储空间的开销，作者把 HDFS 文件块备份数设为 1 来保证存储空间不会增加太多，但同时数据的容错性又成了新的问题. 为了解决这一问题，作者创建了聚簇检验表(clustering check table)，和索引表一起来实现错误发生后的快速恢复. 同时，CCIndex 还给出了一种查询优化机制以支持多维查询. 该优化机制主要利用 HBase 中的一些元数据信息(region-to-server information)来估算子查询结果的大小，根据估算结果生成合适的查询计划，从而减少查询时间.

二级索引方案易于实现，维护代价较低，但也存在一些不足：当索引列较多时，存储开销比较大；索引更新代价比较高，会影响系统的吞吐量；索引对多

① https://github.com/hbase-trx/
② http://github.com/gkulbak/ibase

维查询的支持效率较低.

(3) 基于线性化技术的全局索引

上述两类索引方案均需维护特定的索引结构，当数据更新十分频繁时，索引更新维护的代价很高. 在保证系统性能的前提下，为降低索引更新维护的代价，文献[23]提出了一种基于空间目标排序的索引方案. 其基本思想是：按照一定的规则将覆盖整个研究区域的范围划分为大小相等的格子，并给每一个格子分配相应的编号，用这些编号为空间目标生成一组具有代表意义的数字. 其思想是将 k 维空间的实体映射到一维空间，从而可以利用比较成熟的一维索引技术. 常见的用一维数值对多维空间目标进行排序的方法有 Z 排序、Hilber 曲线、位置键等. 这些技术的思路基本相同，利用一个线性序列来填充空间，构造一种空间填充曲线. 文献[23]以 HBase 作为数据存储方案，用 Z 排序技术对数据进行排序，以 Z-value 作为每条记录的键值. 单纯的 Z 排序方法在搜索过程中会带来一些不必要的搜索空间(false positive search)，作者在此基础上利用 KD 树或四叉树对多维数据空间进行划分，根据最长公共前缀计算每个子空间的名称，并以此作为索引项对各个子空间的数据进行索引，从而提高搜索效率. 但是，该方法在进行空间划分的过程中会产生数据一致性的问题. 虽然目前有相应的解决方案[24]，但是实现起来仍比较复杂，并且带来额外的负担. 而且当数据分布不均匀时，KD 树和四叉树的深度会很大，影响查询效率.

3.2.2　查询处理

从支持的查询接口和查询语言来看，早期的云数据管理系统，例如 BigTable、HBase 和 Cassandra 仅支持一些基本的数据插入和获取接口[3,10]. 随后很多公司和研究机构在丰富查询语句上开展了工作并提出一些“类-SQL 语言”，例如 Yahoo! 的 PigLatin[25]，Facebook 的 HQL[9]，微软的 SCOPE[26] 和 Dryad-LINQ[27] 以及 IBM 的 JAQL[28] 等等. 从查询处理算法来看，目前针对云数据的查询处理和优化主要集中在基于 MapReduce 框架的查询处理. MapReduce 天然地支持分组聚集操作和选择操作，而连接操作的实现则比较复杂. 在分布式环境下数据传输和数据倾斜等问题的出现使得在 MapReduce 上实现连接成为一个非常具有挑战性的问题，下面主要对云数据的连接查询工作进行深入的总结分析. 已有的相关工作主要分为两类，一类是直接在 MapReduce 上实现连接[9,25,28-33]，一类是修改 MapReduce 框架使之更利于连接的实现[34-35]. 下面我们分别介绍这两类工作.

(1) 基于原始 MapReduce 的连接算法

这类算法通过设计 Map 函数、Reduce 函数和数据流来完成连接，涉及到的连接方式包括两表等值连接[9,25,28-29]、两表 θ 连接[30]、多表等值连接[31-32] 和两表集合相似性连接[30] 3 种类型. 如表 3 所示，我们首先对两表等值连接的算法进行分析比较. 设参加连接的两个表分别为 R 和 S，并且 R 为其中数据量较小的表.

表 3　基于原始 MapReduce 的连接算法对比

算法名称	支持的连接类型	数据传输代价	作业个数	主要特性
标准重分区算法[9,25,28-29]	两表等值连接	$\|R\|+\|S\|$	1	优点：算法简单，易于实现 缺点：若部分连接键值对应数据量较大，会造成内存溢出和计算资源不均
改进的标准重分区算法[29]	两表等值连接	$\|R\|+\|S\|$	1	优点：连接阶段对内存要求较小 缺点：当出现数据倾斜时，会造成计算资源不均
广播算法[9,25,28-29]	两表等值连接	$s\times\|R\|$(s 为 S 表数据所在节点个数)	1	优点：适用于连接的表中有一个较小情况，可减少数据排序和传输代价 缺点：适用范围比较狭窄
半连接算法[29]	两表等值连接	$l\times\|R_s\|$(R_s 为利用 S 表过滤后的数据)	3	优点：减少了数据广播过程中的传输代价，可用于广播的表比较大的情况 缺点：增加了对两表进行全表扫描的代价
分片半连接算法[29]	两表等值连接	$\sum\|Rs_i\|$	3	优点：可进一步减少数据广播过程中的传输代价 缺点：要对广播的表进行多次扫描
冗余重分区算法[30]	两表 θ 连接	最大值为 $4r\sqrt{\|R\|\|S\|/r}$(r 为 reducer 个数)	1	优点：可实现两表 θ 连接，解决重分区过程中带来的数据倾斜问题 缺点：混洗过程中数据传输代价较大
数据冗余传输算法[31]	多表等值连接(星形连接，链式连接)	$\sum(\|R_i\|\times\prod\omega_j)$($R_i$ 为参加连接的表，ω_j 为属性共享值)	1	优点：一个 MapReduce job 即可实现多表的连接 缺点：链式连接会引起较大数据传输代价
基于二部图连接算法[32]	多表等值连接(链式连接)	$\sum\|R_{i,uk}\|$($R_{i,uk}$ 为最终参加连接的数据)	$4n-3$(n 是表个数)	优点：数据传输代价较小 缺点：MapReduce job 个数较多，不支持星形连接
基于前缀过滤的算法[33]	两表集合相似度连接	$\|R_{pre}\|+\|S_{pre}\|$($R_{pre}S_{pre}$ 为前缀过滤后的数据)	3	优点：通过前缀过滤减少数据传输代价，利用索引提高连接性能 缺点：只适用于字符串类型的相似连接，分词处理要重复执行多次

标准重分区算法[9,25,28-29]. 该算法类似于 DBMS 中的排序-合并算法,由一个 MapReduce 作业构成. Mapper 读入两个表的数据文件,并根据查询条件对数据进行过滤. 输出的键值对中,键值是连接的列值,数值部分包括记录值和标签两部分,标签用于标识该记录来自哪个表. 在 reduce 的混洗(shuffle)过程中,具有相同连接值的记录被分区到同一个 reducer 上. 针对每一个连接值,reducer 根据标签把记录分成两个集合,然后计算两个集合元素的向量积从而完成连接. 标准重分区算法在现有云数据管理系统比较常见,Pig、Hive 和 Jaql 均实现了这种算法[8,25,28]. 该算法的一个潜在问题是针对某个连接键值计算向量积时,两个表的相关数据都要放入内存进行缓存. 当连接键值基数比较少或者出现数据倾斜时,会导致某个连接键值对应的数据量较大,一方面可能会造成内存溢出,另一方面造成计算资源分布不均匀.

改进的重分区算法[29]. 为了解决标准重分区算法的内存缓存问题,该算法从两个方面进行了改进:首先,map 阶段输出的键值由连接的列值和表名的标签值混合构成,标签值放到键值中可以保证在 reduce 阶段进行排序时,来自其中一个表的数据总是排在另一个表的前面. 其次,在计算卡氏积时,内存中只缓存较小表的数据,而另一个大表的数据以数据流的方式读入内存. 这样,算法对内存大小的要求大大降低.

广播算法[9,25,28-29]. 该算法将两个表中较小的一个以广播的形式传输到另一个表数据所在的节点上,然后在每个节点上直接进行连接. 算法由一个只有 map 函数而 reduce 函数为空的 MapReduce 作业完成. 作业初始化阶段对小表 R 进行数据广播,然后在 Map 阶段直接对数据进行 Hash 连接. 由于广播算法没有 reduce 操作,因此避免了混洗过程中的数据传输和排序. 当进行连接的两个表数据量相差很大时,广播小表的数据传输代价将会大大小于混洗过程中的数据传输代价,从而提高连接效率.

半连接算法[29]. 半连接算法基于广播算法进行改进,旨在减少广播过程中的数据传输量. 广播的表 R 中并不是所有的数据都会参与连接,因此在传输数据之前通过半连接操作去除部分数据. 该算法由 3 个 MapReduce 作业构成. 第 1 个作业主要扫描 S 表并生成其连接键值文件 $S.uk$. 第 2 个作业根据文件 $S.uk$ 中的键值过滤 R 中每个子表的数据,生成一系列的数据文件 Ri. 第 3 个作业依据过滤后的 R 表数据执行广播算法. 尽管半连接算法减少了广播过程中的数据传输量,但增加了对表 S 和 R 的扫描. 因此具体选择哪种算法要根据连接表的大小以及连接键值的分布情况决定.

分片半连接算法[29]. 该算法将半连接的粒度缩小到 S 的每个分片子表 Si,它同样由 3 个 MapReduce 作业构成. 第 1 个作业生成 S 的链接键值文件,与前一个算法不同的是这个作业只有 map 操作,针对每个子表 Si 生成连接键值文件 $Si.uk$. 第 2 个作业执行半连接,针对每个 Si,根据 R 的匹配记录文件和标记生成与子表 Si 相对应的广播数据文件 R_{Si}. 第 3 个作业只有 map 操作,每个 mapper 读入对应的数据文件 R_{Si} 并直接进行连接操作. 与普通的半连接算法相比,分片半连接算法在广播过程中数据传输量较少,但是需要为每个子表 Si 过滤一次 R.

冗余重分区算法[30]. 该算法使用一个二维矩阵表示两表的笛卡尔积,通过将满足连接条件的元素设定为"真"表示各种不同连接类型的结果. 所有的连接均由一个 MapReduce 作业完成,通过均衡每个 reducer 任务输入和输出的数据量来达到减少查询执行时间的目的. 算法根据 reducer 任务的个数 r 将二维矩阵分成 r 个大小均衡的区域,每个 reducer 负责产生相应区域的连接结果,其输入的数据量则等于区域矩形的周长之半. 与以往的连接算法不同,在冗余重分区算法中,一个记录可能被重定向到多个 reducer 任务的区域中. 算法正是通过这种冗余重定向实现了非等值连接,并减轻了数据倾斜的影响,但增加了混洗过程中的数据传输量.

除了两表等值连接,多表等值连接在数据分析和决策支持中的应用也非常广泛,星形连接和链式连接是主要的两种连接形式. 文献[31]提出了一种基于"数据冗余传输"的算法. 该算法只包含一个 MapReduce 作业,数据的冗余传输在 map 之后的混洗过程中进行,冗余传输的次数和方式则由"map key"决定. Map 键值是多个连接属性的集合,其中每个连接属性对应着一个共享值(share),表示该属性 Hash 后的桶数. Mapper 输出的每个键值对可能传输到多个 reducer,其个数由 Map 键值中没有被该表覆盖的连接属性共享值的乘积决定. 在 reduce 阶段,直接对传输到本地的数据进行连接. 这种算法比较适合星形连接或者表数不多的链式连接,随着链式连接的表数不断增多,传输代价也成倍增加. 文献[32]提出了一种利用二部图进行连接的算法,该算法主要应用在链式连接上. 设参加链式连接表的

个数为 n，首先使用 n 个 MapReduce 作业为每个表生成一个二部图，然后执行 $2(n-1)$ 个作业根据二部图按照和链式连接相反的顺序减少每个表参与连接的记录数，最后利用浓密树提高连接的并行度，这样最少再执行 $(n-1)$ 个作业执行连接. 该算法从最大程度上减少了连接过程中的数据传输量，但是需要的 MapReduce 作业个数较多.

与等值连接不同，集合相似性连接要求计算两个表(或者集合)中所有元素的相似度，因此减少数据传输的方法比等值连接复杂. 文献[33]提出了一种使用 MapReduce 实现集合相似性连接的算法，利用“前缀过滤”[36]原则减少参加连接的候选数据对. 该算法包括 3 个步骤：第 1 步计算用于前缀过滤的全局词项排序，包括两个 MapReduce 作业，分别用于统计和排序，第 2 步利用词项排序执行前缀过滤并生成连接结果的行键值对(row-ID pair)，第 3 步根据行键值对取得实际的连接结果，这两步各使用一个 MapReduce 作业. 该算法通过前缀过滤减少了连接过程中的数据传输代价，但其应用范围比较固定，适用于字符串类型的相似连接.

(2) 基于调整后 MapReduce 的连接算法

原始的 MapReduce 框架是一个“过滤-聚集”的过程，这对处理同构的数据源比较有效[37]，然而在处理多表连接时会遇到两方面的问题. 一方面，参加连接的数据源往往是异构的，因此在连接处理过程中需要对不同数据源的数据进行同构化处理，例如增加数据源标记等. 同构化处理过程不但需要额外的存储开销，而且增加了数据传输量. 另一方面，原始的 MapReduce 框架在处理多表连接时会产生大量中间结果和检查点，这也增加了数据传输量.

文献[34]针对异构数据源问题对 MapReduce 框架进行了扩展，在 reduce 步骤结束后增加了一个 merge 的步骤，形成 Map-Reduce-Merge 框架. Merge 的输入数据可以来自不同 reducer 的输出，这样在一个 MapReduce 作业里可以处理多个数据源. 实现连接的过程类似于传统 MapReduce 上的重分区连接，不过在 map 阶段不需要为不同表的数据登记标签，merge 阶段可以将两个表对应 reduer 输出的排序数据进行合并连接. 新加坡国立大学的研究人员提出了 Map-Join-Reduce 框架[35]，并对原始 MapReduce 的处理过程进行了两方面的扩展. 针对第 1 个问题，文献[35]提出了“过滤-连接-聚集”的编程框架，连接函数可从多个数据源读入数据进行处理，连接函数内容和连接顺序由用户定义. 针对第 2 个问题，Map-Join-Reduce 对 Map 完成后的混洗过程进行了扩展，将原来的“一对一”模式扩展成“一对多”模式，Map 函数输出的中间结果一次可以传给多个连接函数. 这样通过相应的分区策略可以用一个 MapReduce 作业完成多表连接，从而减少多个作业处理过程带来的大量中间结果存储和传输问题.

与基于原始 MapReduce 的连接算法相比，基于调整 MapReduc 的连接算法可以通过较少的作业完成原始 MapReduc 框架需要多个作业才能完成的复杂连接，因此可以减少中间结果的数据传输和检查点数量. 对 MapReduce 框架的调整主要通过增加处理函数或者扩展部分数据流程实现，这使得原来简单易用的 MapReduce 框架变得复杂，也增加了编程接口的使用难度.

3.2.3　查询优化

在数据管理系统中，对于一个给定的查询，通常有多种处理策略，查询优化技术负责从多种策略中找出最有效的查询处理计划. 云数据管理系统中的查询优化可以从两个方面进行：一方面在解析查询语句并生成 MapReduce 计划时进行，根据数据的元信息选择执行更为高效的 MapReduce 计划；另一方面在执行 MapReduce 任务时进行，根据数据的统计和资源分配等信息构造详细的任务执行策略. 已有的查询优化工作主要集中在第 2 个方面，下面从任务的调度、任务的处理优化两个方面对已有工作进行总结.

(1) 调度优化

云计算是一个多用户的环境，服务提供商依据签订的相关协议向用户提供不同级别的服务，因此对不同用户提交的查询进行调度以保证服务质量是非常必要的. 另一方面，云计算环境通常是分布式异构的，查询往往被分解成多个任务并行执行，根据资源的占用情况和节点的运行情况对任务进行有效的调度对查询优化有着至关重要的作用. 目前针对调度的优化已经有不少工作，根据调度对象的粒度，可以把已有工作分成 3 个类型：查询调度[38]、MapReduce 作业调度[39]和 MapReduce 任务调度[16,40].

文献[38]提出了一种在云环境下对用户提交的查询进行调度的算法 iCBS. 服务提供商和用户之间通过签订服务等级协议 SLA(Service Level Agreement)来保障云服务的质量和可靠性，SLA 定义了为用户提供的服务标准以及服务商不能满足服务需求的惩罚代价. SLA 涉及云服务中可用性、安全性

等多个方面，iCBS 主要关注查询响应时间. 该算法根据查询的提交时间和该查询的 SLA 相关定义以增量的方式计算其优先系数，依据优先系数对查询进行调度，以尽量减少查询的响应时间，并减少服务提供商因不能满足 SLA 需求而产生的代价，iCBS 的时间复杂度为 $O(\log N)$，其中 N 为查询的数量.

表 4 调度优化算法

算法名称	调度粒度	优化目标	算法复杂度
iCBS[38]	查询	最小化 SLA 代价	$O(\log N)$
FAIR[39]	MapReduce 作业	保证资源平均分配	$O(1)$
CSP 模型算法[40]	MapReduce 任务	最小化实时作业的延迟响应时间	NP 完全问题
LATE[16]	MapReduce 任务	最小化异构环境下作业执行时间	$O(\log M)$

文献[39]提出了一种对 MapReduce 作业进行调度的算法 FAIR 来优化作业的执行效率. 传统的 MapReduce 作业调度方法是先进先出(FIFO)算法，这种算法实现起来比较简单，但是在多用户的环境下会影响作业的执行效率. FAIR 提供了一种让用户公平获取计算资源的调度算法，它使用资源池组织作业，并把资源公平的分到资源池中. 每个用户使用一个资源池，这样每个用户可以获得等同的资源分配. 除此之外，FAIR 允许赋给资源池保证最小共享资源(guaranteed shared resourece)，这样可以保证特定用户、群组或生产应用程序总能获取到足够的资源. Phan 等人[40]关注异构环境下 MapReduce 作业的任务调度优化，把每个任务的执行时间、心跳检测时间间隔、数据输入时间等 5 个变量组合成约束集合，以最小化作业的延迟相应时间为目标函数，将 MapReduce 作业调度问题转化成约束满足问题(Constraint Satisfaction Problem，CSP)进行解决. 文献[16]的调度粒度也是 MapReduce 任务，主要关注掉队任务(straggler task)的调度优化. 在传统的 MapReduce 调度中，为了防止作业执行过程中“木桶效应”的出现，会将掉队任务进行备份执行. 然而原有的掉队任务调度方法假设集群环境的同构性和任务执行的等速性，这在实际的云计算环境中往往是无法保证的. 基于上述问题，文献[16]提出了 LATE 算法，根据所在节点的性能预测每个任务的剩余完成时间，并选择剩余时间最长的任务作为掉队任务进行调度. 在调度过程中，如果有空闲的任务槽位(task slot)出现并且正在运行的任务总数小于特定阈值，则创建该任务的执行副本. 该算法需要对所有正在运行的任务进行剩余时间的预测和排序，算法复杂度为 $O(M)$，M 为正在运行的任务个数.

(2)任务处理优化

基于 MapReduce 实现云数据的查询可以获得良好的扩展性、容错性以及较高的性价比，然而粗犷的批处理模式导致基于原始 MapReduce 框架的查询性能有很大的提升空间. 查询任务处理的优化问题引起了学术界的广泛关注，已有的优化措施包括以下几种：

① 任务共享. 云环境中的数据查询通常是以批处理的方式处理大规模数据，在该模式下通过查询之间的任务共享来减少冗余计算将有效减少查询执行时间和耗费的计算资源. Hive[9]提供了一种用户自定义模式的数据扫描共享(scan share)，如果两个作业的输入数据文件相同，则会创建一个新的 MapReduce 作业负责数据的读入和解析，并为两个作业产生相应的临时输入文件. 这种任务共享方法增加了一个 MapReduce 作业，而且还需要用户自己定义共享函数. 另一类任务共享方法是把满足共享任务条件的作业分到一个组中，使用一个 MapReduce 作业来完成原来多个作业需要完成的工作，不需要用户自定义，也不需要产生临时文件[41-44]. 文献[42-43]主要支持数据扫描共享，而文献[43-44]则支持扫描共享、Map 输入 Map 输出以及 Map 函数的共享.

② 增量计算. 目前在大多数云数据管理应用中，查询的数据规模往往随着新数据的产生而不断增加. 如何使查询流程增量化，并利用已有的查询结果处理新的查询也是目前学术界关注的一个问题. 根据增量计算的触发方法，已有的工作可以分为两类：对用户不透明的方法[45-46]和透明的方法[42,47-48]. Google 的 Percolator 建立在 GFS-BigTable 之上，它通过快照隔离实现了跨行和跨表数据的一致性，使得用户可以跟踪计算过程中的状态，并实现增量计算[45]. Yahoo!的 CBP 提出了一个新的并行编程模型，用来存储和使用运行状态，并实现查询的增量处理[46]. 这两种方法的基本缺陷是要求用户自己编写动态程序来对数据进行有效的增量处理. Nova[42]在 Pig/Hadoop 基础上创建了一个数据流管理器，用来管理不同查询的数据集和查询结果，并支持有状态的数据追加操作. 当查询提交后，管理器判断该查询任务是否可以利用已有的结果进行增量计算. 与 Nova 不同，HaLoop[47]和 Incoop[48]从 MapReduce 任务的层次进行增量计算的处理. Incoop 在分布式文件层使用基于内容的数据块划分方法来增加 map 任务的重用度，并通过在 combine 阶段将混洗的数据粒度减小来最大化 reduce 任务的重用度.

③ 数据组织优化. 云数据管理系统中的数据被分布到多个节点进行管理，在进行查询特别是多表查询时，需要在各个节点间进行数据传输. 如果较多的相关数据存储在一个节点上，那么网络传输代价就会减少，查询时间也会随之减少，因此数据的组织方式会对查询性能产生很大的影响[49]. HadoopDB将数据从分布式文件系统导入到每个节点上的关系数据库系统中，这样可以在本地的关系数据库上分别执行连接[8]. Hadoop＋＋[50]将数据组织优化模块植入 Hadoop 系统之上，主要关注两表连接时的查询优化. Hadoop＋＋在数据导入时对输入数据建立“特洛伊”索引，并将具有相同连接键值的数据放入同一个数据分片中，这样在实现连接时不需要进行数据的网络传输. 该方法没有修改 Hadoop，而是在导入数据时进行数据的重新组织. CoHadoop[51]则是修改了 Hadoop 的数据组织方法，为每个文件增加了“Locator”属性来标识其位置，而所有具有相同“Locator”属性的文件的数据块将被组织到同一个数据节点集合中.

除了上述查询优化方法，目前还有部分工作对 MapReduce 的参数设置进行优化[52-53]，其中文献[52]通过分组的数目对 reducer 个数进行优化，而文献[53]则是通过估计 MapReduce 作业的执行时间提供对多个参数的基于代价的优化. 总的来说，目前已有的优化工作主要集中在数据控制层和数据存储层，而且大部分是基于规则的优化，基于代价的优化工作还比较少，亟待相关研究成果.

3.2.4　在线聚集

在线聚集(Online Aggregation，OLA)在查询处理过程中根据采样数据估计查询结果，并返回真实结果所在的置信区间[54]. 在线聚集的最大优势是可在较短时间内计算出接近实际的查询结果，当置信度和置信区间达到用户要求时，查询即可提前停止. 对于原本执行时间特别长而且对结果精确性要求不高的复杂查询，在线聚集可以大大缩短查询时间. 在线聚集最初提出是在单表上进行聚集的相关操作[55-56]，后来该工作被扩展到多表连接基础上的聚集操作[57-59]以及并行环境中的连接聚集[60-61]. 在线聚集基于关系数据库提出，并在研究领域取得了丰富的成果，但是相关成果在关系数据库领域带来的市场价值却很有限，原因有两点：首先，OLA 要求查询处理的数据以随机顺序出现，这与排序、索引等查询优化算法的原则相违背，因此在已有的关系数据库系统上实现 OLA 需要对其内核进行大规模改动；其次，OLA 的最主要目标是缩短查询运行时间和节省软硬件资源，然而在一个非弹性的数据中心，这个目标的吸引力并不大. 在云计算环境下，OLA 技术又重新引起了人们的关注. 一方面，云计算提供了一种 pay-as-you-go 的服务模式，节省计算资源直接意味着节省开销；另一方面，不同于传统的关系数据库，云数据管理系统内核轻量易于修改. 目前在云计算上的 OLA 已经有一些初步的工作，主要是在 MapReduce 框架上实现大规模数据的查询估计. 其相关技术包括 MapReduce 在线化、数据采样、查询结果估计和收敛程度计算，下面我们分别分析这些技术的已有工作.

(1) MapReduce 在线化. 传统的 MapReduce 数据流是一个批处理的过程，无论是 map 任务还是 reduce 任务，必须处理完所有数据后才产生输出结果，而且 reduce 任务也必须在所有的 map 任务完成后才开始执行. OLA 要求数据流是一个在线处理的过程，处理完部分样本数据后就输出估计的查询结果，MapReduce 的在线化处理[62-63]为云环境下的 OLA 提供了实现平台. 文献[62]的 MapReduce 在线化主要面向“自增迭代”的算法，通过 map 定期传送数据给 reduce 实现作业内部的在线化，并通过集群“共享内存”实现作业之间的在线化. 这种在线化方法结构简单，易于实现，但是其扩展性及容错性不及传统的 MapReduce. Condie 等人[63]基于操作器(operator)之间数据流水线实现了在线化的 MapReduce 系统 HOP. HOP 结合网络负载状况以及 combine 操作的压缩比等因素设计数据流控制机制，从而动态控制 mapper 与 reducer 之间的数据传输粒度. 当一个查询由多个 MapReduce 作业构成时，生产作业根据任务执行进度定期调用 reduce 并生成快照文件(snapshot)，消费作业通过读取快照文件从而实现数据在作业之间的流水化. HOP 保留了传统 MapReduce 的扩展性和容错性，比较适合作为在线聚集的实现平台.

(2) 数据采样. 为了保证估计结果和置信区间的准确性和收敛速度，在线聚集要求采样数据具有随机性和无偏性[55-56]. 从关系数据表进行采样的方法主要有三类[56,64]：顺序扫描、索引扫描和索引采样. Wu 等人[61]提出了从分布式数据表采样的方法，首先根据表在各节点上的分布情况计算每个节点应采样的数据量大小，然后在每个节点上进行索引采样. 在云环境下，很多数据以块(block)为单位直接存储在分布式文件系统上，MapReduce 处理数据也

通常以块为单位,因此上述基于关系数据库的采样方法无法直接使用.目前很多 MapReduce 的在线聚集工作假设数据以随机顺序存储或者假设一个随机数据输入队列的存在[54,63],通过顺序扫描数据队列即可获得随机无偏的数据.然而当数据以聚集相关列的顺序存储时,简单的顺序扫描便无法获取随机数据,因此在云环境下如何从直接存储在分布式文件中的数据中进行随机采样仍然是亟待解决的问题.

(3)查询结果估计.查询估计方法应当具有无偏性和持续性[57].无偏性是指如果不断重复采样和估计的过程,估计值的数学期望应该等于实际查询结果.持续性是指随着采样和估计步数的不断增加,估计值应该逐渐接近实际查询结果.目前已有的查询结果估计算法可以分为两类,一类是通过样本和总体数据量的大小对样本的聚集结果进行扩展[61,64].假设查询语句为 SELECT $op(expression(t))$ FROM T.设随机变量为 $|T|\times expression_p(t)$,当元组 t 满足查询选择条件时,$expression_p(t)$ 的取值为 $expression(t)$,否则取值为 0,则总体均值 μ 即为聚集查询结果,总体方差为 σ^2.根据中心极限定理,当采样数据随机且无偏时,样本数据的均值 $\bar{\mu}$ 趋近一个均值为 μ,方差为 σ^2/n 的正态分布.设 T_n 是总体表 T 的采样数据集合,那么总体查询结果可通过用 T 和 T_n 的大小比例对全表数据的 $expression_p(t)$ 之和进行扩展得到.这种方法实现简单,而且支持增量计算.但是需要预先得到总体表的数据量,而且查询结果的估计受数据分布和采样质量的影响较大.

为了解决上述问题,Pansare 等人[54]提出了利用未知样本概率分布进行估计的方法,假设每个数据块在 MapReduce 中的调度时间和处理时间均与聚集结果相关,并针对每个数据块 $block_i$ 构造随机变量 $Z_i=(x_i,t_i^{\text{sch}},t_i^{\text{proc}})$.该方法利用贝叶斯公式,根据已处理完数据块的聚集值计算未处理样本数据聚集值的概率分布:$P(\Theta|X)=\dfrac{P(X|\Theta)P(\Theta)}{P(X)}$,其中,$\theta$ 表示未处理样本的聚集值;X 表示已经处理完样本的聚集值.总体的查询结果通过对 $P(\Theta|X)$ 积分进行估计.这种方法通过贝叶斯理论从一定程度上消除了采样数据不均衡所带来的问题,但是算法的假设较强,而且只能支持一个 MapReduce 作业的查询处理,不支持由多个 MapReduce 作业构成的多表聚集的结果估计.

表 5 查询结果的估计方法

估计方法	理论依据	聚集类型	主要特性
利用样本和总体数据量[61,64]	中心极限定理	单表聚集 多表聚集	优点:算法简单,易于实现,可支持增量计算 缺点:对采样数据要求较高,数据不均衡导致估计结果不准确;要预先知道总体数据量
利用未知样本概率分布[54]	贝叶斯理论	单表聚集	优点:允许采样不均衡,不需预先知道总体大小 缺点:不支持增量计算,无法扩展到多表聚集

(4)结果收敛程度计算.结果收敛程度主要用来衡量当前估计值和实际结果的差距,帮助用户判断估计结果是否达到满意的程度.目前结果收敛程度的计算方法有两类,一类采用绘制"收敛曲线"的方法体现随着查询不断进行,估计结果的变化情况[62].变化的度量标准采用以下公式计算:$METRIC_f=diff(sig(R_i),sig(R_f))$,其中,$R_f$ 是到目前查询进程为止的最新结果;R_i 是与 R_f 相邻的估计结果;$sig(R_i)$ 代表结果 R_i 的一个标识,它可以是完整的结果 R_i,也可以是能够代表 R_i 的一个压缩表征;$diff()$ 用于计算两个结果标识的欧氏距离.用户可以根据收敛曲线的斜率来推测查询结果后续的变化情况.这种方法计算量不大,实现起来也比较容易,收敛曲线可以让用户直观地观察出估计结果的变化.但其缺点是无法给出估计值的精确度,而且仅仅根据相邻结果的距离来体现收敛程度还不够准确.

另一类收敛程度衡量方法是给定置信度 α,在每次采样并得到查询估计值后计算实际查询结果 v_r 的置信区间 $[v-\varepsilon,v+\varepsilon]$[54,56-57,61],这意味着 v_r 落入置信区间的概率为 α.随着查询的不断进行,置信区间的宽度逐渐变窄,用户可根据区间的宽度判断查询是否提前终止.当样本数据随机且无偏时,根据中心极限定理,置信区间可表示为 $[\bar{\mu}-\varepsilon,\bar{\mu}+\varepsilon]$,$\varepsilon=z_\alpha\bar{\sigma}_n/\sqrt{n}$[56-57,61],其中,$Z_\alpha$ 是和置信度相关的分位点;$\bar{\sigma}_n$ 是样本数据的方差,n 是样本数据量.文献[54]通过贝叶斯公式计算未知样本的分布函数,并在此基础上使用 Gibbs 采样算法[65]计算置信区间.通过置信区间可以比较精确的反应估计值的收敛程度,目前在云环境下的相关工作还局限于单个 MapReduce 作业,如何计算多表或者多个 MapReduce 作业构成的聚集查询的置信区间仍是待解决的问题.

4 未来工作展望

作为一项高性价比管理海量数据的技术,云数

据管理系统引起了工业界和学术界的广泛关注.本文依据云数据管理系统框架对云数据查询技术的相关工作进行了总结和分析.总体来说,目前该领域的研究工作处于起步阶段,还存在着大量有价值的研究问题:

(1)数据分布策略.数据的组织情况会直接影响数据插入以及查询的效率,均匀的数据分布将大大提高数据存取的性能.在云数据管理系统中,数据被划分到多个节点进行存储管理,其存储的节点位置往往由每条记录的主键值决定.现有的工作一般选择单个字段或者多个字段的简单组合作为主键值,而没有考虑到对数据分布的影响①[66].对于这个问题,可以从以下两个方面来考虑:首先根据查询类型和数据分布情况选定生成主键值的字段,这些字段的组合应当能够唯一地确定主键值,并有利于数据的分散和查询时数据记录的定位;其次,设计从多维字段的定义域到线性主键值的映射函数,该函数要保证数据分布的负载均衡,并在查询处理过程中尽可能地缩小目标数据集大小.此外,这种数据分布策略应当具有自适应性,可以根据插入数据的不断变化而进行相应的调整.

(2)索引管理技术.目前在云数据管理系统中针对海量数据的索引已经有一些研究工作,并取得了相应的成果,在以下两个方面还有待深入研究.一方面,目前云数据管理系统中的索引方案大都是以关系数据库中的索引为基础,对其进行适当修改而成.这些索引都是基于磁盘的索引,比较适合于相对稳定的数据.但是对于数据频繁更新的情况,索引更新维护的代价比较高.因此,如何在云计算环境下,设计能支持频繁更新和多维查询的索引方案是一个富有挑战性的工作.另一方面,现有的索引大都能够支持点查询、范围查询等简单查询,但对于一些复杂查询无法提供很好的支持.特别是在一些特定的应用领域,如海量空间数据管理、海量时间序列数据管理等领域,往往需要支持一些相对比较复杂的查询.因此,针对某些特定的应用领域,设计相应的索引方案,能够支持一些特定的复杂查询,具有重要的意义.

(3)查询优化算法.查询处理方法和优化策略对云数据管理系统来说是一个关键性的问题.目前的研究工作主要侧重在利用 MapReduce 处理框架实现一些关系数据库中传统的查询处理算法,或者改进 MapReduce 调度算法和处理流程以适应查询处理算法,但是对云计算环境下数据存储和查询处理的特点考虑得较少.云计算环境和传统的单机数据库环境相比,数据量大而且分布存储,但是数据的划分技术却不如分布式数据库中的划分技术成熟,因此数据的分布往往比较粗犷,很难利用数据划分带来的查询优势;另外为了达到较高的可用性和容错性,数据往往存在多个冗余备份,我们认为利用备份数据进行查询并行度和数据传输方面的优化是很有意义的.在进行查询优化的过程中,增加并行度可以充分利用系统的计算资源,提高查询性能,但是单纯的增加并行度可能导致传输数据代价过大,从而造成网络拥塞和计算节点的空闲等待;而单纯的最小化传输代价则可能导致数据倾斜问题加重,因此如何寻求查询并行度和数据传输代价的平衡也是一个不容忽视的问题.以上讨论的是根据优化规则生成查询计划和执行查询计划的问题,然而对不同的数据量和数据分布其最优的查询计划也不相同,如何为不同的查询选择查询计划也是一个亟待解决的问题.查询计划的选择往往通过估算其查询代价进行,代价可以通过查询总开销和总时间表示.结合云计算环境下数据和查询的特点建立不同查询算法的代价计算模型也是颇具挑战性的问题.

(4)查询进程估计.相比于传统的分布式数据库,云计算环境中查询进程和剩余时间的估计有着更重要的作用.一方面,云环境下往往面临生物、气象等领域大规模数据的查询和分析,运行时间较长,有的查询甚至需要十几天[67],提供查询剩余时间的反馈对用户来说有很强的实用价值.另外,对于云环境下的一些查询时间较短但是实时性要求比较高的应用,进程估计会给查询任务的调度提供重要的参考.对查询代价的估算、在线聚集的实现、云环境的性能调优和资源配置等问题来说,进程估计也是一个非常关键的步骤.在云环境下实现查询进程估计不仅任务并行带来的挑战,云环境中集群规模庞大、节点异构、高出错率、和数据倾斜等特点使得这个问题解决起来更加困难.目前的研究工作主要考虑了并行的因素[16,68],但是对于其他云环境的特点没有考虑,如何在一个大规模的云环境下提供准确的查询进程估计还有很多研究工作要做.

(5)基于多表的在线聚集算法.从聚集结果估计和置信区间的计算来看,已有的相关工作主要侧重在包含一个 MapReduce 作业聚集查询的 OLA 算法设计.实际应用中经常涉及到基于多表的复杂

① http://pkghosh.wordpress.com/2010/09/19/

查询,他们往往由多个 MapReduce 作业构成,实现这种查询的在线聚集是一个亟待解决的问题.在传统的 MapReduce 作业处理流程中,每个操作任务完成后将输出数据写入文件,后面的操作任务才能开始.OLA 要求数据以增量的方式进行处理,因此多 MapReduce 作业的 OLA 必须在处理过程流水线化的 MapReduce 上实现.在设计聚集查询处理和置信区间计算算法时还需要结合 MapReduce 以及云计算环境的特点提高在线聚集的处理速度,比如减少混洗过程中数据传输量和 reduce 阶段的工作,尽量避免增量计算过程中的重复工作等.从数据采样的实现过程来看,样本的随机性和无偏性会直接影响查询结果估计的准确性以及置信区间的收敛速度,已有的研究工作往往假设数据以随机顺序存储或者假设一个随机数据队列的存在,从队头读取数据即可达到随机的效果.然而在实际应用中,数据的存储顺序往往与某个属性相关,如何从这种非随机分布的数据上进行随机采样是在线聚集过程中的一个关键问题.数据的随机采样技术在单机数据库上有很多研究工作[56,64,69-70],提出的方法包括堆文件扫描[56]、索引扫描[64]、伯努利模型采样[69]等.云计算环境下数据分布在大量节点上,而且数据的读写以块为单位进行,这些特点增加了随机采样的难度,值得深入研究.文献[70]针对直方图估计提出了以数据块为单位的采样方法,并利用交叉验证的思想推导出估计值的准确性与样本大小和数据分布的关系公式,其思想可以借鉴到云数据在线聚集的采样算法中.不同之处是该文献提出的算法是一个一次性采样的过程,而在线聚集要求采样算法是在线并且增量的过程,即它能够保证样本大小平缓增长而且时刻保持随机的顺序.在线采样过程中不仅要保证数据随机性,还必须保证每步采样的数据与已采样本数据不重复,这也是算法设计中必须考虑的问题.

5 结 论

随着信息产业的不断发展,计算机要处理的数据规模呈指数级增长,各种应用对数据管理的需求也变得多样化,统一而复杂的关系数据库已经不能满足纷繁多样的应用.云数据管理系统为海量数据管理提供了一种高性价比的解决方案,日益成为学术界和工业界共同关注的热门问题.本文对近几年国内外在云数据查询领域的主要研究成果进行了总结,综述了云数据管理系统中查询技术若干主要问题的研究现状,包括云数据的索引管理、查询处理、查询优化以及在线聚集等,并对相关技术进行了深入的对比分析,最后指出仍然存在的问题和可能的解决办法.总的来说,云数据管理系统中查询技术的研究仍然处于刚刚起步的阶段,仍然有大量具有挑战性的关键问题需要深入研究,为国内的数据库研究者提供了广阔的研究空间.

参 考 文 献

[1] Abadi D J. Data management in the cloud: Limitations and opportunities. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2009, 32(1): 3-12

[2] Zhou Ao-Ying. Data intensive computing-challenges of data management techniques. Communications of CCF, 2009, 5(7): 50-54(in Chinese)
(周傲英. 数据密集型计算-数据管理技术面临的挑战. 中国计算机学会通讯, 2009, 5(7): 50-54)

[3] Chang F, Dean J, Ghemawat S, Hsieh W C, Wallach D A, Burrows M, Chandra T, Fikes A, Gruber R E. Bigtable: A distributed storage system for structured data//Proceedings of the 7th Conference on Symposium on Operating Systems Design and Implementation(OSDI2006). Seattle, 2006: 7-15

[4] Cooper B F, Ramakrishnan R, Srivastava U, Silberstein A, Bohannon P, Jacobsen H, Puz N, Weaver D, Yerneni R. PNUTS: Yahoo!'s hosted data serving platform//Proceedings of the 34th Conference on Very Large Databases (VLDB2008). Auckland, 2008: 1277-1288

[5] Pavlo A, Paulson E, Rasin A, Abadi D J, DeWitt D J, Madden S, Stonebraker M. A comparison of approaches to large-scale data analysis//Proceedings of the 2010 International Conference on Management of Data (SIGMOD2009). Rhode Island, 2009: 165-178

[6] Stonebraker M J, Abadi D, DeWitt D J, Madden S, Paulson E, Pavlo A, Rasin A. MapReduce and parallel DBMSs: friends or foes? Communications of the ACM, 2010, 53(1): 64-71

[7] Shi Y, Meng X, Zhao J, Hu X, Liu B, Wang H. Benchmarking cloud-based data management systems//Proceedings of the 2nd Workshop on Cloud Data Management (CloudDB2010). Toronto, 2010: 47-54

[8] Abouzeid A, Pawlikowski K B, Abadi D, Silberschatz A, Rasin A. HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads//Proceedings of the 35th Conference on Very Large Databases (VLDB2009). Lyon, 2009: 922-933

[9] Thusoo A, Sarma J, Jain N, Shao Z, Chakka P, Anthony S, Liu H, Wyckoff P, Murthy R. Hive: A warehousing solution over a map-reduce framework//Proceedings of the 35th Conference on Very Large Databases (VLDB2009). Lyon, 2009: 1626-1629

[10] Robert L G, Yunhong G. On the varieties of clouds for data intensive computing. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2009, 32(1): 44-50

[11] Kossmann D, Kraska T, Loesing S, Merkli S, Mittal R, Pfaffhauser F. Cloudy: A modular cloud storage system. Proceedings of the VLDB Endowment, 2010, 3(2): 1533-1536

[12] Dean J. Evolution and future directions of large-scale storage and computation systems at Google// Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC2010). Indiana, 2010: 1-1

[13] Schad J, Dittrich J, Quian-Ruiz J. Runtime measurements in the cloud: Observing, analyzing, and reducing variance. Proceedings of the VLDB Endowment, 2010, (1): 460-471

[14] Agrawal D, Abbadi A, Antony S, Das S. Data management challenges in cloud computing infrastructures//Proceedings of the 6th International Workshop on Databases in Networked Information Systems (DNIS). Aizu-Wakamatsu, 2010: 1-10

[15] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters//Proceedings of the 6th Conference on Symposium on Opearting Systems Design and Implementation (OSDI2004). California, 2004: 1-10

[16] Zaharia M, Konwinski A, Joseph A D, Katz R, Stoica I. Improving MapReduce performance in heterogeneous environments//Proceedings of the 8th Conference on Symposium on Operating Systems Design and Implementation (OSDI2008). California, 2008: 29-42

[17] Wu S, Jiang D, Ooi B, Wu K. Efficient b-tree based indexing for cloud data processing. Proceedings of the VLDB Endowment, 2010, 3(1): 1207-1218

[18] Wang J, Wu S, Gao H Li J, Ooi B. Indexing multidimensional data in a cloud system//Proceedings of the 2010 International Conference on Management of Data (SIGMOD2010). Indianapolis, 2010: 591-602

[19] Ding L, Qiao B, Wang G, Chen C. An efficient quad-tree based index structure for cloud data management//Proceedings of the 12th International Conference on Web-Age Information Management (WAIM2011). Wuhan, 2011: 238-250

[20] Zhang X, Ai J, Wang Z, Lu J, Meng X. An efficient multi-dimensional index for cloud data management//Proceedings of the 1st Workshop on Cloud Data Management (CloudDB2009). Hong Kong, China, 2009: 17-24

[21] Papadopoulos A, Katsaros D. A-Tree: Distributed indexing of multi-dimensional data for cloud computing environments//Proceedings of the 3rd International Conference on Cloud Computing Technology and Science (CloudCom2011). Athens, 2011: 407-414

[22] Zou Y, Liu J, Wang S. CCIndex: A complemental clustering index on distributed ordered tables for multi-dimensional range queries//Proceedings of the International Conference on Network and Parallel Computing (NPC2010). Zhengzhou, 2010: 247-261

[23] Nishimura S, Das S. MD-HBase: A scalable multi-dimensional data infrastructure for location aware services// Proceedings of the 11th International Conference on Mobile Data Management (MDM 2011). Luleå, 2011: 224-226

[24] Das S, Agrawal D, Abbadi A. G-Store: A scalable data store for transactional multi key access in the cloud//Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC 2010). Indianapolis, 2010: 163-174

[25] Olston C, Reed B, Srivastava U, Kumar R, Tomkins A. Pig latin: A not-so-foreign language for data processing//Proceedings of the 2008 International Conference on Management of Data (SIGMOD2008). Vancouver, 2008: 1099-1110

[26] Chaiken R, Jenkins B, Larson P, Ramsey B, Shakib D, Weaver S, Zhou J. SCOPE: Easy and efficient parallel processing of massive data sets. Proceedings of the VLDB Endowment, 2008, 1(2): 1265-1276

[27] Yu Y, Isard M, Fetterly D, Budiu M, Erlingsson U, Pradeep Kumar Gunda, Jon Currey. DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language//Proceedings of the 8th Conference on Symposium on Operating Systems Design and Implementation (OSDI2008). California, 2008: 1-14

[28] Beyer K, Ercegovac V, Gemulla R, Balmin A, Eltabakh M, Kanne C, Özcan F, Shekita E. JAQL: A scripting language for large scale semistructured data analysis. Proceedings of the VLDB Endowment, 2011, 4(12): 1272-1283

[29] Blanas S, Patel J, Ercegovac V, Rao J, Shekita E, Tian Y. A comparison of join algorithms for log processing in MaPreduce//Proceedings of the 2010 International Conference on Management of Data (SIGMOD2010). Indianapolis, 2010: 975-986

[30] Okcan A, Riedewald M. Processing theta-joins using MapReduce//Proceedings of the 2011 International Conference on Management of Data (SIGMOD2011). Athens, 2011: 949-960

[31] Afrati F, Ullman J. Optimizing joins in a map-reduce environment//Proceedings of the 13th International Conference on Extending Database Technology (EDBT2010). Lausanne, 2010: 99-110

[32] Zhou M Q, Zhang R, Zeng D D, Qian W N, Zhou A Y. Join optimization in the MapReduce environment for column-wise data store//Proceedings of the 6th International Conference on Semantics, Knowledge and Grids (SKG2010). Ningbo, 2010: 97-104

[33] Vernica R, Carey M, Li C. Efficient parallel set-similarity joins using MapReduce//Proceedings of the 2010 International Conference on Management of Data (SIGMOD2010). Indianapolis, 2010: 495-506

[34] Yang H, Dasdan A, Hsiao R, Parker D. Map-reduce-merge: Simplified relational data processing on large clusters//Proceedings of the 2007 International Conference on Management of Data (SIGMOD2007). Beijing, 2007: 1029-1040

[35] Jiang D, Tung A, Chen G. MAP-JOIN-REDUCE: Toward scalable and efficient data analysis on large clusters. IEEE Transactions on Knowledge and Data Engineering (TKDE), 2011, 23(9): 1299-1311

[36] Chaudhuri S, Ganti V, Kaushik R. A primitive operator for similarity joins in data cleaning//Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006). Atlanta, 2006: 5

[37] Pike R, Dorward S, Griesemer R, Quinlan S. Interpreting the data: Parallel analysis with Sawzall. Scientific Programming (SP), 2005, 13(4): 277-298

[38] Chi Y, Moon H, Hacigümüs H. iCBS: Incremental cost-based scheduling under piecewise linear SLAs. Proceedings of the VLDB Endowment, 2011, 4(9): 563-574

[39] Zaharia M, Borthakur D, Sarma J S, Elmeleegy K, Shenker S, Stoica I. Job scheduling for multi-user MapReduce clusters. University of California at Berkeley, California: Technical Report UCB/EECS-2009-55, 2009

[40] Phan L, Zhang Z, Loo B, Lee I. Real-time MapReduce scheduling. University of Pennsylvania, Pennsylvania: Technical Report MS-CIS-10-32, 2010

[41] Wang X, Sarma A, Olston C, Burns R. CoScan: Cooperative scan sharing in the cloud//Proceedings of the 2nd ACM Symposium on Cloud Computing. Cascais, 2011: 138-151

[42] Olston C, Chiou G, Chitnis L, Liu F, Han Y, Larsson M, Neumann A, Rao V, Sankarasubramanian V, Seth S, Tian C, ZiCornell T, Wang X. Nova: Continuous Pig/Hadoop workflows//Proceedings of the 2011 International Conference on Management of Data (SIGMOD2011). Athens, 2011: 1081-1090

[43] Nykiel T, Potamias M, Mishra C, Kollios G, Koudas N. MRShare: Sharing across multiple queries in MapReduce. Proceedings of the VLDB Endowment, 2010, 3(1): 494-505

[44] Gates A, Natkovich O, Chopra S, Kamath P, Narayanam S, Olston C, Reed B, Srinivasan S, Srivastava U. Building a highlevel dataflow system on top of MapReduce: The pig experience. Proceedings of the VLDB Endowment, 2009, 2(2): 1414-1425

[45] Peng D, Dabek F. Large-scale incremental processing using distributed transactions and notifications//Proceedings of the 10th Conference on Symposium on Operating Systems Design and Implementation (OSDI2010). Vancouver, 2010: 251-264

[46] Logothetis D, Olston C, Reed B, Webb K, Yocum K. Stateful bulk processing for incremental analytics//Proceedings of the 1st ACM Symposium on Cloud Computing. Indianapolis, 2010: 51-62

[47] Bu Y, Howe B, Balazinska M, Ernst M. HaLoop: Efficient iterative data processing on large clusters. Proceedings of the VLDB Endowment, 2010, 3(1): 285-296

[48] Bhatotia P, Wieder A, Rodrigues R, Acar U, Pasquini R. Incoop: MapReduce for incremental computations//Proceedings of the 2nd ACM Symposium on Cloud Computing. Cascais, 2011: 70-73

[49] Jiang D, Ooi B, Shi L, Wu S. The performance of MapReduce: An in-depth study. Proceedings of the VLDB Endowment, 2010, 3(1): 472-483

[50] Dittrich J, Quiané-Ruiz J, Jindal A, Kargin Y, Setty V, Schad J. Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). Proceedings of the VLDB Endowment, 2010, 3(1): 518-529

[51] Eltabakh M, Tian Y, Özcan F, Gemulla R, Krettek A, McPherson J. CoHadoop: Flexible data placement and its exploitation in Hadoop. Proceedings of the VLDB Endowment, 2011, 4(9): 575-585

[52] Chen S. Cheetah: A high performance, custom data warehouse on top of MapReduce programs. Proceedings of the VLDB Endowment, 2011, 3(2): 1459-1468

[53] Herodotou H, Babu S. Profiling, what-if analysis, and cost-based optimization of MapReduce programs. Proceedings of the VLDB Endowment, 2011, 4(11): 1111-1122

[54] Pansare N, Borkar V, Jermaine C, Condie T. Online aggregation for large MapReduce jobs. Proceedings of the VLDB Endowment, 2011, 4(11): 1135-1145

[55] Haas P. Large-Sample and deterministic confidence intervals for online aggregation//Proceedings of the 9th Conference on Scientific and Statistical Database Management (SSDBM 1997). Olympia, 1997: 51-63

[56] Hellerstein J, Haas P, Wang H. Online aggregation//Proceedings of the 1997 International Conference on Management of Data (SIGMOD1997). Tucson, 1997: 171-182

[57] Haas P, Hellerstein J. Ripple joins for online aggregation//Proceedings of the 1997 International Conference on Management of Data (SIGMOD1999). Philadephia, 1999: 287-298

[58] Jermaine C, Dobra A, Arumugam S, Joshi S, Pol A. A disk-based join with probabilistic guarantees//Proceedings of the 2005 International Conference on Management of Data (SIGMOD2005). Chicago, 2005: 563-574

[59] Hellerstein J, Avnur R, Chou A, Hidber C, Olston C, Raman V, Roth T, Haas P. Interactive data analysis: The control project. Computer, 2008, 32(8): 51-59

[60] Luo G, Ellmann C, Haas P, Naughton J. A scalable hash ripple join algorithm//Proceedings of the 2005 International Conference on Management of Data (SIGMOD2005). Baltimore, 2005: 252-262

[61] Wu S, Jiang S, Ooi B, Tan K. Distributed online aggregation. Proceedings of the VLDB Endowment, 2009, 2(1): 443-454

[62] Bose J H, Andrzejak A, Hogqvist M. Beyond online aggregation: Parallel and incremental data mining with online Map-Reduce//Proceedings of the Workshop on Massive Data Analytics on the Cloud. Raleigh, 2010: 3

[63] Condie T, Conway N, Alvaro P, Hellerstein J, Gerth J, Talbot J, Elmeleegy K, Sears R. Online aggregation and continuous query support in MapReduce//Proceedings of the 2010 International Conference on Management of Data (SIGMOD2010). Indianapolis, 2010: 1115-1118

[64] Olken F, Rotem D. Simple random sampling from relational databases//Proceedings of the 12th Conference on Very Large Databases (VLDB1986). Kyoto, 1986: 160-169

[65] Casella G, George E. Explaining the gibbs sampler. The American Statistician, 1992, 46(3): 167-174

[66] Gonzalez H, Halevy A, Jensen C, Langen A, Madhavan J, Shapley R, Shen W. Google fusion tables: Data management, integration and collaboration in the cloud//Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC 2010). Indianapolis, 2010: 175-180

[67] Schatz M C. Cloudburst: Highly sensitive read mapping with MapReduce. Bioinformatics, 2009, 25(11): 1363-1369

[68] Morton K, Friesen A, Balazinska M, Grossman D. Estimating the progress of MapReduce pipelines//Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE2010). California, 2010: 681-684

[69] Haas P, Koenig C. A bi-level Bernoulli scheme for database sampling//Proceedings of the 2004 International Conference on Management of Data (SIGMOD2004). Vancouver, Paris, 2004: 275-286

[70] Chaudhuri S, Das G, Srivastava U. Effective use of block-level sampling in statistics estimation//Proceedings of the 2004 International Conference on Management of Data (SIGMOD2004). Paris, 2004: 287-298

SHI Ying-Jie, born in 1983, Ph. D. candidate. Her current research interests include cloud data management, online aggregation techniques over big data.

MENG Xiao-Feng, born in 1964, professor and Ph. D. supervisor. His research interests include Web data management, mobile data management, native XML database and cloud data management.

Background

Cloud computing has emerged as a prevalent infrastructure and attracted a lot of attentions from companies and academic circles. Though there has not been a standard definition about cloud computing, we can summarize the substantial features of it: scalability, fault tolerance, high performance cost, pay-as-you-go, etc. Data management system is one of the applications that are deployed in the cloud, it provides a solution to manage big data with high cost performance. Many cloud-based data management systems have been proposed by companies and are serving online right now, however, there is still big room for function extension and performance improvement. In order to improve the performance and availability of cloud data management systems, more and more attentions have been given to this area. Recent years, researchers have done much work on query processing, query optimization and index technique in the cloud. However, many issues in this area are still in its infancy, or even not addressed yet. In this paper, the authors propose a system architecture for cloud data management. Based on this architecture, the content of this paper mainly provides a summary of previous work from several aspects: index management, query processing, query optimization and online aggregation. This paper also indicates the challenging problems and future research directions, which helps researchers pay attention to the interesting issues needed to address.

This research was partially supported by the National Natural Science Foundation of China (Nos. 61070055, 91024032, 91124001), the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (No. 11XNL010), the National High Technology Research and Development Program (863 Program) of China (No. 2012AA010701).