软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(2):175−197 [doi: 10.3724/SP.J.1001.2013.04345]

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

数据管理技术的新格局*

覃雄派[1,2,3], 王会举[1,2,3], 李芙蓉[1,2,3], 李翠平[1,2,3], 陈 红[1,2,3], 周 烜[1,2,3], 杜小勇[1,2,3], 王 珊[1,2,3]

[1](教育部数据工程与知识工程重点实验室(中国人民大学),北京 100872)

[2](萨师烜大数据管理与分析研究中心(中澳),北京 100872)

[3](中国人民大学 信息学院,北京 100872)

通讯作者: 覃雄派, E-mail: qxp1990@sina.com, http://info.ruc.edu.cn/t/cn/覃雄派

摘 要: 数据获取技术的革命性进步、存储器价格的显著下降以及人们希望从数据中获得知识的客观需要等,催生了大数据.数据管理技术迎来了大数据时代.关系数据库技术经历了20世纪70年代以来40年的发展,目前遇到了系统扩展性不足、支持数据类型单一等困难.近年来,noSQL技术异军突起,对多种类型的数据进行有效的管理、处理和分析;通过并行处理技术获得良好的系统性能;并以其高度的扩展性,满足不断增长的数据量的处理要求.试图沿着数据库技术进步的历史脉络,从应用维度(操作型与分析型应用)入手,为读者展开当今数据管理技术的新格局,讨论具有挑战性的重要问题,并介绍作者自己的研究工作.

关键词: 关系数据库;noSQL;大数据;操作型;分析型;新格局

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 覃雄派,王会举,李芙蓉,李翠平,陈红,周烜,杜小勇,王珊.数据管理技术的新格局.软件学报,2013,24(2):175−197. http://www.jos.org.cn/1000-9825/4345.htm

英文引用格式: Qin XP, Wang HJ, Li FR, Li CP, Chen H, Zhou X, Du XY, Wang S. New landscape of data management technologies. Ruanjian Xuebao/Journal of Software, 2013,24(2):175−197 (in Chinese). http://www.jos.org.cn/1000-9825/4345.htm

New Landscape of Data Management Technologies

QIN Xiong-Pai[1,2,3], WANG Hui-Ju[1,2,3], LI Fu-Rong[1,2,3], LI Cui-Ping[1,2,3], CHEN Hong[1,2,3], ZHOU Xuan[1,2,3], DU Xiao-Yong[1,2,3], WANG Shan[1,2,3]

[1](Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University of China), Ministry of Education, Beijing 100872, China)

[2](Sa Shi-Xuan Big Data Management and Analytics Research Center (Sino-Australia), Beijing 100872, China)

[3](Information School, Renmin University of China, Beijing 100872, China)

Corresponding author: QIN Xiong-Pai, E-mail: qxp1990@sina.com, http://info.ruc.edu.cn/t/cn/覃雄派

Abstract: The revolutionary progress of data collecting techniques, dramatic decrease of the price of storage devices, as well as the desirability of people to extract information from the data have given birth to the so-called big data and data management technologies usher in the age of big data. RDBMS (relational database management system) undergoes a development of 40 years since the 1970s and now encounters some difficulties such as limited system scalability and limited data variety support. In recent years, noSQL technologies has risen suddenly as a new force. The technologies can manage, process, and analyze various types of data, achieve rather high performance with the help of parallel computing, can handle even bigger volume of data with the nice property of highly scalability. The paper follows the path of database technology progress and unfolds the new landscape of data management technologies from the angle of

* 基金项目: 国家自然科学基金(61070054, 60873017, 61170013); “核高基”国家科技重大专项(2010ZX01042-001-002, 2010ZX01042-002-002-03); EMC 中国研究院“EMC 全球 CTO 办公室”资金

收稿时间: 2012-06-12; 定稿时间: 2012-10-16; jos 在线出版时间: 2012-11-23

CNKI 网络优先出版: 2012-11-23 12:08, http://www.cnki.net/kcms/detail/11.2560.TP.20121123.1208.003.html

applications (operational as well as analytic applications). The paper also identifies some chanllenging and important issues that deserve further investigation, with the authors' recent research work introduced at the end.

Key words: RDBMS (relational database management system); noSQL; big data; operational; analytic; new landscape

近年来,数据管理技术正在经历重大的变革.我们试图沿着技术进步的历史脉络,从应用维度入手(包括操作型应用和分析型应用),为读者展开当今数据管理技术的全景画卷,分享我们的研究体会.

1 关系数据库发展历史回顾

20 世纪 70 年代初,IBM 工程师 Codd 发表了著名的论文"A Relational Model of Data for Large Shared Data Banks",开启了数据管理技术的新纪元——关系数据库时代.关系数据库管理系统(relational database management system,简称 RDBMS)就是在这篇论文的基础上设计出来的.在此之前的数据库系统主要有基于层次模型的层次数据库(比如 IBM 公司的 IMS 系统)、基于网状模型的网状数据库(比如 IDS 数据库)等.这些数据库的主要缺点是,其数据模型对于普通用户来讲难以理解,同时,软件的编写和数据的模式(schema)联系过于紧密.层次和网状数据结构都是导航式(navigational)的数据结构,存取特定的数据单元必须在软件中按照一定的存取路径去提取,当数据模式发生改变时,已经编写好的软件需要做相应的修改.

Codd 提出的关系数据模型基于表格(关系)、行、列、属性等基本概念,把现实世界中的各类实体(entity)及其关系(relationship)映射到表格上.这些概念易于理解.Codd 还为关系模型建立了严格的关系代数运算.

关系模型在首次提出来的时候曾经受到严重的质疑.一些专家学者认为,关系数据库需要进行表格的连接操作,不可能获得和层次、网状数据库一样的高性能.但是历史的潮流不可阻挡,世界各地的研究人员致力于关系数据模型及相关技术(特别是查询处理技术)的研究和开发.研究人员对包括存储、索引、并发控制、查询优化、执行优化(包括各种连接算法)等关键技术进行了研究,并且针对数据库的 ACID(atomicity 原子性、consistency 一致性、isolation 隔离性、durability 持久性)保证提出了日志、检查点和恢复等技术.这些技术解决了数据的一致性、系统的可靠性等关键问题,为关系数据库技术的成熟以及在不同领域的大规模应用创造了必要的条件.研究人员还掀起了关系数据库系统开发热潮,特别值得提及的两个原型系统,一个是 IBM 开发的 System R,另一个是加州大学伯克利分校开发的 Ingres.这两个系统最后都演化为成熟的关系数据库产品.若干围绕关系技术的数据库公司和产品部门纷纷成立,大浪淘沙,其中获得商业上成功的公司(部门)主要有 IBM(DB2)、Oracle (oracle)、Informix、Sybase、微软(SQL server)、SAP 等.这些数据库技术公司创造了庞大的数据库产业,每年创造巨大的产值,仅 Oracle 一家(http://www.oracle.com/us/corporate/investor-relations/financials/q4fy11-421266.pdf),2011 年财政年度的销售收入就达到了 356 亿美元($35.6 Billion).

1974 年,交互式查询语言 SEQUEL(structured english query language)作为 System R 项目的一部分,被 IBM 的工程师开发出来.这是 SQL 语言的前身.现在,SQL 语言已经成为国际标准,成为各个厂家数据库产品的标准的数据定义语言、查询语言和控制语言.SQL 语言非常容易理解,普通用户经过简单培训就可以掌握和使用.使用 SQL 语言,用户只需要告诉系统查询目的是什么(需要查询什么数据),即"What",并不需要告诉系统怎么样去做,即"How",包括数据在磁盘上是怎么存储的、可以使用什么索引结构来加快数据访问以及使用什么算法对数据进行处理等,都无需用户关心.关系数据库系统的查询优化器根据用户的查询特点和数据的特点,自行选择合适的查询执行计划,通过过滤、连接、聚集等操作完成用户的查询,达到执行速度快、消耗资源少、尽快获得部分结果等目标.查询优化器经历了从简单到复杂、基于规则到基于代价模型的发展阶段,是关系数据库系统最重要的和最复杂的模块之一.

容易理解的模型、容易掌握的查询语言、高效的优化器、成熟的技术和产品,使得关系数据库成为数据管理的首要选择.虽然 1970 年~2000 年间并非所有的数据库系统都是基于关系模型的,针对特定应用出现了包括面向 XML 文档管理和查询(XQuery)的 XML 数据库、面向多媒体数据管理的多媒体数据库、面向高维时空数据处理的时空数据库、RDF 数据库、面向对象的数据库等(比如 gem stone,objectivity,object store 等),但是关系数据库技术和产品占据了绝对的统治地位(包括技术和市场).关系数据库管理系统厂商还通过扩展关系模型,

支持半结构化和非结构化数据的管理,包括 XML 数据、多媒体数据等,并且通过用户自定义类型(user defined type,简称 UDT)和用户自定义函数(user defined function,简称 UDF)提供面向对象的处理能力,进而巩固了关系数据库技术的王者地位.关系数据库还可以通过裁剪适应特定的应用场合,比如流数据的处理(stream processing,一般基于内存数据库实现),用户可以在源源不断涌来的序列数据上运行 SQL 查询,获得时间窗口上的结果,实现数据的及时(on the fly)分析和监控.

关系数据库系统最初主要用于事务处理领域.随着数据的不断积累,人们需要对数据进行分析,包括简单汇总、联机分析处理(online analytical processing,简称 OLAP,主要是多维分析)、统计分析、数据挖掘等.在关系模型上支持这些分析操作成为一个自然的选择,比如建立于 RDBMS 上的联机分析处理技术 ROLAP(MOLAP 模型是实现 OLAP 分析的另外一种模型,在此不赘述).分析型应用和操作型应用(主要是事务处理)具有不同的特点.操作型应用的数据处理任务主要包括对数据进行增加、删除、修改和查询以及简单的汇总操作,涉及的数据量一般比较少,事务执行时间一般比较短.而分析型应用(包括联机分析处理和数据挖掘等)则需要扫描大量的数据,进行分析、聚集操作,最后获得数据量相对小得多的聚集结果和分析结果.有些分析处理需要对数据进行多遍扫描(比如数据挖掘的 *K*-Means 算法),分析查询执行的时间以分钟计或者小时计.针对分析型应用的数据存取特点,一些学者提出了不同于行存储模型(面向事务处理的关系数据库一般采用行存储模型)的列存储模型(columnar storage model)[1].列存储模型有自己独特的优势,当查询只涉及关系的某些数据列时,不会造成无关数据列的提取,从而减少 I/O,提高查询效率.此外,由于同一列数据紧密存放在一起,这些数据来自同样的数据域,表现出很高的冗余度,很容易进行数据压缩,节省存储空间;数据压缩技术也能够提高处理效率,因为压缩数据的提取能够节省 I/O 带宽;一些压缩技术的研究,甚至可以支持在非解压的情况下进行查询处理[2−4].

围绕 RDBMS,形成了一个完整的生态系统(厂家、技术、产品、服务等),提供了包括数据采集、清洗和集成、数据管理、数据查询和分析、数据展现(可视化)等技术和产品[5],创造了巨大的数据库产业.

20 世纪 90 年代末以来,硬件技术发生了巨大的变化,主要有:

(1) CPU 的研发已经不再走单纯提升频率的路线(遇到功耗等困难),而是在一个芯片上集成更多的核心,出现了多核,甚至众核 CPU.面向图形处理的 GPU 则集成更多的处理核心,工作频率也比通用 CPU 高,提供了前所未有的并行处理潜能.

(2) 存储器的价格持续下降,使得服务器的内存容量不断增长,中小规模、甚至大规模的数据库系统可以把数据全部装载到内存中,实现快速访问和处理.

(3) 新的持久性存储器不断涌现,包括基于闪存技术的固态硬盘(solid state disk,简称 SSD)以及相变内存(phase change memory,简称 PCM)技术等,这些持久性的存储器具有与硬盘不同的读写特性,传统的 RDBMS 对其并未感知并加以利用.

针对这些新硬件,需要对数据库的存储、索引、并发控制方法、查询优化、恢复等技术进行必要的修改,使其适应新硬件的特点,获得更高的性能.这些研究包括:

(1) 面向全内存处理的数据结构、索引结构[6]:内存的访问模式与磁盘根本不同,面向磁盘的存储结构和索引结构不适合基于内存的处理,必须加以改变,或者提出新的存储结构和索引结构(比如 T 树[7]).

(2) 面向 Cache 优化的查询处理算法:一旦数据全部存放在内存中,则系统的查询性能的优化,主要考虑数据在内存和 CPU 之间的快速交换问题.研究人员提出了一系列 Cache 感知[8−10]和 Cache 不敏感[11−13]的算法,以加快数据处理操作.这些算法的共同目标是如何减少 Cache 缺失(cache miss).

(3) 随着 CPU 上集成的核心数量的增多,数据处理软件也需要做相应的改变,从面向多核的并行数据处理[14]到解决多个核心的总线争用[15]、多个核心的处理的协调等,都是需要解决的问题.

(4) GPU 集成的核心数量比通用 CPU 多得多,工作频率也更高,如何利用 GPU 实现快速的数据处理和分析的研究方兴未艾[16−20].

(5) 目前,基于闪存技术的存储器价格不断下降,开始走向大规模应用.闪存的读取一般比写入要快得多,写入操作不能在原有数据上进行,需要新的数据块进行写入,废弃的数据块要进行统一的收集和回

收.针对闪存的特点,研究人员调整了数据结构、日志处理方法[21−23]等,力图利用其读取速度快的优点,规避其写入速度慢的缺点,获得更高的数据处理性能.

(6) 保存在内存中的数据在掉电情况下将彻底丢失.为了保证内存数据库的可恢复性,必须针对内存处理研究新的恢复技术,实现高效的日志处理、检查点操作和恢复过程,否则,内存数据库的高速数据处理特性将无法得到有效发挥.研究人员研究了一系列面向内存数据库的恢复技术,其中,并行恢复技术[24,25]是加快日志处理、检查点操作和加快恢复过程的有效手段.

内存数据库系统是上述技术的综合体现,主要的内存数据库系统有面向事务处理的 Timesten,Altibase, Hana,CSQL 等以及面向分析型应用的 MonetDB(商业化版本为 VectorWise),C-Store(商业化版本为 Vertica)等.

2 大数据时代的来临与关系数据库的困难

存储器价格的下降和容量的巨大提升、互联网应用的发展(电子商务、社交网络)、科学研究的需要等,使得我们收集到前所未见的庞大数据集,大数据时代已经来临.大数据的主要来源包括(但不限于这些):

(1) 大型的电子商务系统.由于用户数量和交易数量巨大,其积累的数据量相当惊人,比如淘宝每天新增的数据量超过 20TB(http://www.csdn.net/article/2010-12-02/282918).为了有效处理如此大规模的数据,淘宝自行开发了海量数据库系统 Ocean Base.

(2) 基于互联网的社交网络是一个重要的大数据来源(参见第 3.1.4 节).

(3) 科学研究也积累了大量的数据,比如欧洲原子能中心的大型强子对撞机,每年积累的数据量是 15PB,需要有效的手段对其进行处理.

(4) 无线射频技术、传感器网络技术、物联网技术(Internet of things)的大规模部署,将以前所未有的速度生成数据.这些数据即使经过过滤,只保留有效的数据,其数据量也是惊人的.

(5) 在电信领域,为了对用户的行为进行有效的分析,以便提供更加优质的服务,需要对用户的历史通话记录和计费信息进行分析.在中国这样的大国,特别是在发达地区,由于用户数量庞大,电信公司每天新增的数据量是非常大的.

大数据的主要特点不仅仅是数据量巨大(volume),而且还包括其他重要的特点,比如数据类型多样(variety)/处理复杂(complex)、数据生成速度快/需要快速的处理能力(velocity)等.

- 首先,数据生成的速度在加快.在事务处理领域,基于互联网的电子商务系统可能迎来突发的事务请求,中国铁道部的网络售票系统即是一个典型例子(特别是在春运期间).这并不是一个特例,当电子商务系统通过互联网面向全球用户进行服务时,这种情况将变得很平常.科学研究中的天文观测和高能物理实验都需要对快速采集的大量数据进行处理.
- 其次,我们需要处理的数据类型丰富多样,除了结构化数据(主要指关系型数据)以外,还包括各种半结构化和非结构化的数据.比如,互联网积累和存储了大规模的非结构化数据,包括各种类型的文档、媒体文件;人们基于互联网的共享和协作关系,可以描述成一种网络关系,由于人数众多,形成了一个巨大的图,对社会网络进行分析是当前的研究热点,这也是非结构化的数据.
- 最后,在类型多样的数据上,其处理和分析操作也是多样和复杂的.我们不仅仅需要简单的增加、删除、修改、查询和汇总操作,更需要进行复杂的计算和分析,这些分析依赖于人工智能、机器学习、数据挖掘、情感计算、网络分析、搜索等技术.比如在“危险边缘”游戏中,打败人类冠军的“沃森(Watson)”计算机(IBM 公司研发的一个原型系统)集成了人工智能、自然语言处理等先进的技术.正如 Stonebraker[26]所说的,SQL 查询只是简单的汇总分析(little analytics),从数据中获得知识必须依赖于更复杂的计算,大数据之上必然要进行大分析(big analytics,即复杂分析),才能把数据变废为宝,否则,我们将迷失在数据的海洋中.

大数据时代迎面而来,关系数据库系统并未做好准备:

- 首先,关系模型不容易组织和管理所有类型多样的数据,比如在关系数据库里,管理大规模的高维时空

数据(倪明选等人在文献[27]描述了大规模高维时空数据处理的挑战和关系数据库技术的困难)、大规模的图数据(graph)等都显得力不从心.

- 其次,如何通过大量节点的并行操作实现大规模数据的高速处理,仍然是一个巨大的挑战.在关系数据库上进行大规模的事务处理,不仅要解决读操作(查询)的性能问题,更需要解决修改操作的性能问题,大量的新事务(操作)到达,需要有效的处理,才能保证数据的持久性和可靠性.
- 再次,在关系数据库上进行数据的复杂分析,可以使用统计分析和数据挖掘软件包;现有的统计分析、数据挖掘软件包(SPSS,SAS,R 等)能够处理的数据量受限于内存大小,并行化程度不够.从数据库中提取数据,注入到分析软件中进行分析,将导致大量的数据移动,在大数据时代已经不合适,分析应该向数据移动(move computation toward data),尽可能地靠近数据完成计算.通过数据的划分和并行计算,实现高性能的数据分析成为必然选择.

基于关系数据库系统,我们可以采用一些暂时的手段解决大数据处理问题.比如:

(1) 关系数据库的分割(sharding).首先对数据进行逻辑上的划分(比如按照地区对电信计费数据进行划分),把数据分布到多个服务器上;然后对应用程序进行修改,以便支持查询的路由选择,利用多个服务器分担工作负载来提高系统的性能.Sharding 技术带来一些严重问题:当某个数据分片突破一定的数据量后,对数据进行重新分割将是非常不容易的事情;数据重新分割将导致程序的修改,程序和模型的独立性(关系模型的优点)被丢弃;此外,为了对数据进行正确的操作,全局数据模式(schema)必须在每个服务器上保存一份,它们之间的同步也是一个大问题.

(2) 关系数据库的非规范化(de-normalization)处理.对关系数据进行非规范化处理虽然增加了数据的冗余,但是可以更容易地对数据进行分割,分布到多个节点上进行并行操作;操作过程中利用冗余信息,减少节点间的数据交换.可以说,非规范化技术利用数据冗余换取数据库系统的扩展性能.对数据进行规范化处理的目的是降低数据的冗余度,方便实现数据的一致性约束,非规范化处理则增加了数据冗余,使数据的一致性维护变得困难起来.

(3) 为关系数据库部署分布式缓存(distributed cache).在 RDBMS 的前端部署分布式缓存技术(比如 memcached),把最近存取的数据保存在若干服务器的内存中,方便后续的操作.缓存技术并非万能,它能够加速读操作,但是对数据的持久保存和写操作的作用却不大;当内存不足时,免不了要进行 Cache 和 RDBMS 的数据交换,这些数据交换对读操作和写操作都带来一定的延迟;Cache 层的引入,还增加了系统的复杂度和运维成本.因此,Cache 技术只能解决部分问题,而不是全部.

上述技术增强了 RDBMS 的能力,能够暂时应对大数据处理和分析的挑战.但是,这些技术不能完全应对现代数据管理的重要挑战(数据规模巨大、数据类型多样等),从根本上解决问题.一些半结构化数据强行使用关系模型进行建模,无法获得良好的性能和扩展能力,比如图数据(参见第 4.1.3 节 RDBMS 和图数据库(graph database)的遍历性能比较).

为了对大规模数据进行处理,无论是操作型应用还是分析型应用,并行处理都是唯一的选择.这个并行处理不仅是跨越多核的,更为重要的是,它是跨越节点的,依赖于大量节点的并行处理来提高性能.大量节点构成的分布式系统,即便成本不是问题而选用高端的、可靠的硬件设备,但由于集群规模很大,达到上千节点,节点的失败、网络的失效变得很平常,容错保证变得尤其重要.对系统进行大规模横向扩展,关系数据库系统并未为此做好准备,目前还没有一个关系数据库系统部署到超过 1 000 个节点规模的集群上,而 MapReduce 技术则已经达到 8 000 节点的部署规模.

根据 Brewer[28]提出的 CAP 理论(后来由 Gilbert 和 Lynch 证明[29]),在大型分布式系统中,一致性(consistency)、系统可用性(availability)和网络分区容忍性(network partition tolerance)这 3 个目标中,只可以获得其中两个特性,追求两个目标将损害另外一个目标,3 个目标不可兼得(如图 1 所示).换言之,如果追求高度的一致性和系统可用性,网络分区容忍性则不能满足.关系数据库一般通过 ACID 协议保证数据的一致性,并且通过分布式执行协议,比如两阶段提交协议等(2PC)保证事务的正确执行,追求系统的可用性,于是丧失了网络分区

的容忍性.在大量节点组成的集群系统中,由于节点失败稀松平常,有可能造成数据库查询不断重启,永远无法结束的情况.ACID 实施了强一致性(strong consistency)约束,使得关系数据库系统很难部署到大规模的集群系统中(几千个节点规模).

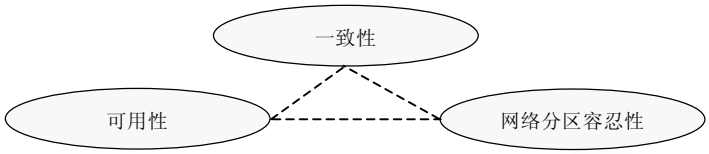


Fig.1 Theory of CAP

图 1 CAP 理论

3 noSQL 技术的崛起

与此同时,noSQL 技术(noSQL 技术不是一种技术,而是一类技术,其主要特点是采用与关系模型不同的数据模型)顺应时代发展的需要,异军突起,蓬勃发展.各类 noSQL 技术在设计的时候,考虑了一系列新的原则,首要的问题是如何对大数据进行有效处理.对大数据的操作不仅要求读取速度要快,对写入的性能要求也是极高的,这对于写入操作密集(write heavy)的应用来讲非常重要.这些新原则包括:

(1) 采用横向扩展的方式(scale out)应对大数据的挑战,通过大量节点的并行处理获得高性能,包括写入操作的高性能(参见第 3.1.2 节对 HBase 写入能力的介绍),需要对数据进行划分(partitioning)并进行并行处理.

(2) 放松对数据的 ACID 一致性约束,允许数据暂时出现不一致的情况,接受最终一致性(eventual consistency).

(3) 对各个分区数据进行备份(一般是 3 份),应对节点失败的状况等.一个广为接受的一致性约束框架 BASE(basically available,soft state,eventual consistency)是一种弱一致性(weak consistency)约束框架.

关系模型是一种严格的数据建模方法,而对于类型多样的数据,包括图、高维时空数据等,关系模型显得过于严格,人们希望通过列表、集合、哈希表等概念对数据进行建模,这也是 noSQL 兴起的重要原因之一.虽然使用关系模型能够对这些对象进行建模,但是某些操作的执行效率却不尽如人意,比如图的遍历等.

面向现实的挑战,基于上述的设计理念,各种 noSQL 技术和系统被研发出来,解决了类型多样的大数据的管理、处理和分析问题.下文从应用角度分别介绍各类操作型 noSQL 技术和分析型 noSQL 技术.

3.1 操作型(operational)noSQL技术及其特点

依据存储模型,操作型 noSQL 技术可划分成基于 Key Value 存储模型、基于 Column Family(列分组)存储模型、基于文档模型和基于图模型的 noSQL 数据库技术这 4 类.本文从数据划分和系统扩展性、持久性保证、数据一致性保证、事务语义等方面对典型的技术和系统进行统一介绍.限于篇幅,本文只介绍各类技术的主要技术特点和系统特色.

3.1.1 基于 Key Value 存储的 noSQL 技术

基于 Key Value(键值对)存储的主要系统包括 Tokyo Cabinet/Tyrant[30],Redis[31],Voldemort[32],Oracle Berkeley DB[33],Amazon Dynamo/SimpleDB[34]等.它们的共同特点是,利用哈希表维护 Key 值到具体数据(value)的映射,通过 Key 值可以很方便地对数据进行查找.对于单个 Key 的查找来说,Key Value 存储能够获得良好的性能.虽然 Key Value 存储的 Value 部分内部具有某种结构,即存储某种类型的数据,但是 noSQL 系统并不对其进行解释,而是返回给应用程序进行处理,应用程序必须根据事先约定的格式进行后续处理.用户无法根据 Value 的某个属性直接向 noSQL 系统提交查询.Redis 赋予了 Value 某些简单的类型,包括数值型、列表、无序集合、排序集合等.在 noSQL 数据库上,对于“查找一个部门的所有员工”这样的查询来说,需要应用程序首先获得某个部门的所有员工的 ID 列表(即 Key 列表),然后使用这个列表逐一地提取每个员工的数据(value),这种操

作依赖于应用程序来实现;在关系数据库系统中,这类查询可以通过 SQL 语句方便地实现,程序员的负担很轻.当应用程序只关心 Value 里某部分数据时,Key Value 存储的效率不高,因为必须提取整个 Value 进行解析.

由于 Key Value 存储模型和查询的简单性,有利于把数据进行横向分割,分布到大规模集群上进行存储和处理,从而获得很高的操作性能(特别是写入的性能).具体的数据分割方式和扩展性保证,参见第 3.1.5 节.

3.1.2 基于 Column Family 存储的 noSQL 技术

基于 Column Family(数据列分组)存储技术的主要 noSQL 系统包括 Cassandra[35],Big Table[36],HBase[37]等.Google 的 Big Table 系统的存储结构是典型的 Column Family 存储.在 Column Family 存储中,同样通过 Key Value 基础模型对数据进行建模,但是 Value 具有了更精巧的结构,即一个 Value 包含多个列,这些列还可以分组(column family),呈现出多层嵌套映射(map)的数据结构特点.由于每列数据是带有时间戳(timestamp)的,可以在 Column Family 里维护多个 Key Value 映射的版本.在需要对历史数据的变动情况进行分析的场合,这样的建模方法正好能够提供有力的支持.HBase 是受 Big Table 启发而开发的基于 Column Family 存储的开源 noSQL 技术.在接口方面,HBase 提供传统的 SQL 查询接口,可以方便地对数据进行增加、删除、修改、查询和简单汇总(聚集).HBase 凭借其强大的扩展能力,被应用于日志处理等领域(比如电信应用的日志分析).Facebook 对 HBase 进行了持续的改进,极大地提高了其吞吐能力(尤其是写入能力),达到每天完成 200 亿个写操作(折合每秒 23 万个写操作)[38]的性能(http://highscalability.com/blog/2011/3/22/Facebooks-new-realtime-analytics-system-hbase-to-process-20.html),对社交网络的用户信息发布和交互行为进行记录和分析.传统关系数据库由于 ACID 协议的强约束性,无法达到这样的扩展能力,虽然可以利用内存处理技术(内存数据库)达到极高的写入性能,却是在不保证持久性的情况下获得的,HBase 则没有这个瓶颈.Cassandra 也是受到 Big Table 的启发,但是做了大幅度的改变.Cassandra 在 Column Family 下引入超级列(super column)概念层次的映射关系,以便对数据进行更加精细的建模,并建立必要的索引,加快查询处理.Cassandra 还能把多个 Column Family 在磁盘上存储在一起,以便在同时访问时提供更高的性能.

3.1.3 基于 Document 存储的 noSQL 技术

基于 Document(文档)存储的技术由来已久,比如 IBM 的 Lotus Notes.这里介绍的基于 Document 存储的 noSQL 技术是基于传统文档存储技术的新发展.Document 存储技术仍然以 Key Value 存储模型作为基础模型.这个模型可以对文档的历史版本进行追踪,每个文档又是一个 Key Value 的列表,形成循环嵌套的结构,文档格式一般采用 JSON(Javascript object notation)或者类似于 JSON 的格式.对于特定的查询来说,Document 存储的效率更高.Document 存储给予数据库设计者极大的灵活性对数据进行建模,但是对数据进行操作的编程负担落在了程序员身上.由于数据的循环嵌套结构特点,应用程序有可能变得越来越复杂、难以理解和维护,需要掌握好灵活性和复杂性之间的平衡.主要的技术和产品包括 CouchDB[39],MongoDB[40]和 Riak[41]等.

3.1.4 基于 Graph 存储的 noSQL 技术

基于 Graph(图)存储的系统包括 Neo4J[42],InfoGrid[43],Infinite Graph[44],Hyper Graph DB[45]等.有些图数据库基于面向对象数据库创建,比如 Infinite Graph,在节点的遍历等图数据的操作中,表现出优异的性能.在新的图数据库的设计中,扩展性作为重要的目标被考虑,目的是对大规模的图数据进行有效的管理和分析.图数据库和上述 3 类 noSQL 技术在存储模型、物理设计、数据分布、数据遍历、查询处理、事务的语义等方面都具有明显的差异.随着社交网络、科学研究(药物、蛋白质研究)以及其他应用领域不断发展的需要,更多的数据以图作为基础模型进行表达更为自然,而且这些数据的数据量是极其庞大的,比如,Facebook 拥有超过 8 亿的用户,对这些用户的交互关系进行管理、分析是极大的挑战.这些分析不仅仅是根据一定的条件查找图的节点或者图的边,更为复杂的处理是在图的结构上进行分析,比如社区的发现等(如图 2 所示,http://www.nature.com/nphys/journal/v8/n1/full/nphys2162.html,译文见 http://www.ccf.org.cn/resources/1190201776262/2012/04/16/14.pdf).

针对图数据的处理,工业界和学术界联合制定了新的评测基准——GRAPH 500(如图 3 所示),用于对超级计算机以及大规模集群的图数据处理能力进行评估.人们可以基于这个 Benchmark 对硬件和软件进行持续的优化,以应对大规模图数据的处理挑战.

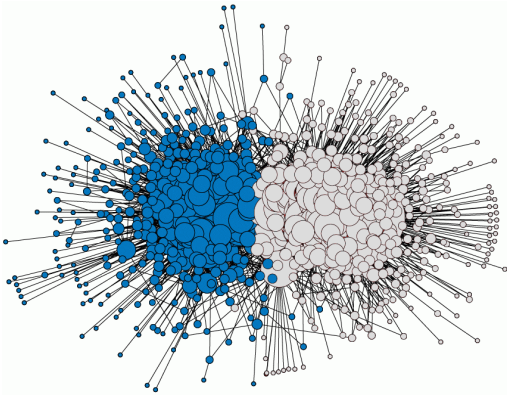


Fig.2 Analysis of a network of links between Web sites about US politics (liberal vs. conservative)

图 2 美国政治相关博客之间超链接的分析(自由派与保守派)

Fig.3 GRAPH 500 benchmark and the rank list

图 3 GRAPH 500 评测基准(benchmark)和排行榜

3.1.5 操作型 noSQL 技术小结

(1) noSQL 的数据模式

noSQL 技术的数据模式不是很严格,包括 Document 存储和 Column Family 存储,同一类实体的属性集可以是不同的.这种弱结构化(less structured)存储机制非常方便设计者根据应用的变化及时地更改数据的模式.而关系数据库的模式必须严格定义,对其进行更改是一个代价较大的操作.

(2) noSQL 的数据持久化与可靠性保证

对数据进行严格持久化(把数据写入稳定存储器)可以保证系统的可靠性,但是非常影响系统的性能;不同的 noSQL 技术采用不同的策略,在持久性保证和性能之间做出折中.在单个服务器上,可以通过把若干个写操作累积起来统一地写入磁盘(group commit),即调用 fsync(操作系统提供的文件更新接口函数)来提高写入操作的吞吐量,但是对于单个用户操作来说,则容易造成延迟的增加.Redis 给程序员提供了控制 fsync 操作的几个选项,包括每个更新操作调用一次 fsync,或者每 N 秒调用一次 fsync,以及针对某些应用(比如把 Redis 作为高速数据缓存)完全关闭 fsync 操作.对于磁盘来说,进行顺序写入比进行随机写入的效率要高.为了减少随机写入的次数,Cassandra,Hbase,Redis 和 Riak 把更新操作顺序写入一种日志结构文件.Cassandra 和 HBase 使用了一种从 Big Table 借鉴来的技术,即把日志和原有的数据组合起来的合并树结构(log structured merge tree).Riak 通过具有日志结构的 Hash 表(log structured Hash table),提供了类似的功能.CouchDB 则改变了传统的 B+树结构,使得所有的数据结构更新都以添加的方式保存到外存上.这些技术提高了写操作的吞吐量,但是需要经常对日志文件进行紧缩(compact)操作以截短日志文件,并且把所有更新集成到数据结构中.

为了应付节点失败、掉电等问题的影响,数据一般在不同节点上有备份.MongoDB 提供了复制集(replica set)的概念,指定每个文档由哪些服务器进行保存.MongoDB 给开发者一些选项,是保证所有的复制都收到更新、还是在不保证所有的数据都收到更新情况下,允许程序继续往下执行.几乎所有的 noSQL 技术都提供这样的多服务器数据复制机制.比如,HBase 是基于 HDFS 开发的,充分利用了 HDFS 的持久性保证,所有的写操作都复制到两个或者多个 HDFS 节点,保证数据的多个备份得到更新,以便互相保护.Riak,Cassandra 和 Voldemort 支持更多的复制选项,这几个系统虽然有些技术细节上的差别,但是它们都允许用户指定一个参数 N,即最终由 N 个节点保存数据的拷贝.另外,还可以指定参数 W(W<N),表示必须有 W 个节点确认数据已经写入稳定存储器,才允许程序把控制返回给用户端.为了应付整个数据中心垮掉的情况,跨越数据中心的数据复制是必须的,一般采用异步的方式实现,因为广域网的延迟太大,同步方式极大地影响了数据的写入性能.

(3) noSQL 的扩展性

几乎所有的 noSQL 技术和系统在设计之初都把高度的扩展性能作为首要的目标之一.这里讲的扩展性是

横向扩展(scale out),而不是纵向扩展(scale up).面对大数据处理的挑战,纵向扩展将很快遇到瓶颈,而且代价非常昂贵,横向扩展才是正确的技术选择.所谓横向扩展,首先需要把数据分割到多个节点上(机器上),然后对处理算法进行相应修改,对数据进行并行操作,通过并行计算提高系统的性能.当数据量增大或者负载加重时,可以增加处理节点,然后动态地把数据和负载重新分布到整个集群系统上,达到高效率的数据处理的目的.而纵向扩展则是通过更换或者升级一台机器的 CPU、内存、磁盘存储器、网络接口卡等来实现更高的性能.

在 noSQL 系统中,主要的数据存取都是通过基于 Key 的查找实现的,为了支持横向的扩展,自然的方式是对所有的数据基于 Key 值进行分割.主要的数据分割方式有哈希方法(Hash partitioning)以及 Range 方法(range partitioning).好的 Hash 方法可以把数据均匀地分布到各个节点上.在分布式环境下,一种成熟的 Hash 技术是分布式 Hash 表(distributed Hash table,简称 DHT).Amazon 的 Dynamo 使用了该项技术.这项技术还出现在 Cassandra,Voldemort,Riak 等系统中.在 DHT 中,需要使用一个 Hash 函数 H 把 Key 值均匀地映射到一系列整数中,比如从 0 到 L,可以把 0 和 L 首尾相连,形成一个环,$H(Key)$ mod L 运算就能把 Key 值映射到[0,L]上.图 4 展示了一个 DHT 实例.在图 4 中,服务器 A 负责所有 Hash 值落在[7,233]的 Key 值的管理,…,而服务器 E 则负责 Hash 值落在[875,6]的 Key 值的管理.为了对数据进行复制以支持更高的容错性,可以把数据保存到负责邻近 Key 范围的节点上.比如,当复制因子为 3(*replication factor*=3)时,所有映射到[7,233]的 Key 和对应的数据被保存到 A,B,C 这 3 个服务器上[46].

通过 Hash 方法进行数据分布,一般会导致数据不均匀.因此,很多采用 DHT 技术的 noSQL 系统(包括 Riak)通过在物理机器上创建虚拟节点来解决这个问题.比如,在一台物理服务器 A 上虚拟出 4 个节点 $A1$,$A2$,$A3$ 和 $A4$.不同的 Key 经过 Hash 映射后,分配到不同的虚拟节点,各个虚拟节点管理了 Key 空间的不同部分,虚拟节点再映射到物理节点,有利于各个物理机器的负载得到均衡.

Voldemort 通过手动配置一个比实际服务器数量大得多的分区数量,使得各个服务器分配到更多的数据分区(数据量较少),从而避免各个物理节点的数据量和负载出现不均衡状况.Cassandra 则根据历史负载情况,通过异步地调整服务器在环(ring)中的位置来调整服务器的负载.

Google 的 Big Table 则使用一种层次性 Range 分区方法把数据分割成 Tablet,分布到各个节点上.两个小的 Tablet 可以合并,而大的 Tablet 可以再分解,总是保持一个 Tablet 的大小在 100MB~200MB 之间.系统采用 Master/Slave 的主从架构,Master 节点负责保存元信息,并且实时监控每个 Tablet 的大小、负载情况、服务器的可用性等,对用户的查询进行路由选择,选择合适的 Tablet Server 进行服务.当元信息变得很大时,可以把元信息进一步划分到多个 Tablet 上,形成了 3 层导航寻访架构,如图 5 所示.HBase 项目是 Big Table 的开源实现,在分区技术上与 Big Table 是类似的.此外,MongoDB 也采用了类似的技术,其配置(configuration)节点存储了数据访问的路由信息,这些配置节点通过两阶段提交协议保证元信息的同步,扮演类似于 Big Table 系统的 Master 节点的功能.Twitter 的 Gizzard 系统也采用了层次性的路由选择和访问机制[46].

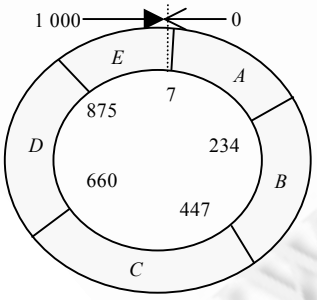


Fig.4 The ring of a distributed hash table[46]

图 4 分布式 Hash 表构成的环[46]

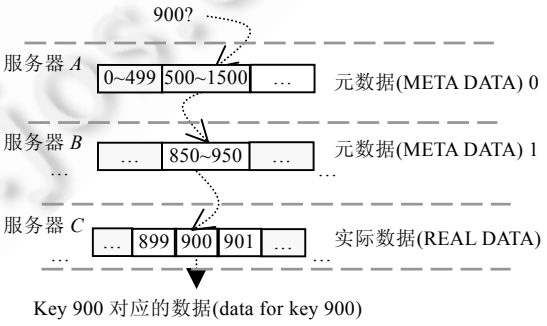


Fig.5 Range partition technique of Big Table[46]

图 5 Big Table 的 Range 分区技术[46]

(4) noSQL 的一致性保证

数据的一致性包括强一致性和最终一致性以及处于两者之间不同程度的一致性保证.回顾本节开始介绍的 CAP 理论,由于所有的 noSQL 系统都通过大量节点来实现系统的扩展性,从而提供足够高的性能.一个首要的特性必须得到保证,即系统的网络分区容忍性(partition tolerant).那么,在 noSQL 系统的设计目标上,必须在数据的一致性和系统可用性之间做出折中.这里的一致性描述的是,当客户请求某个数据时,存放该数据副本的服务器节点对数据该取什么样的值能够达成一致.比如,一个 Key 的数据被复制到 N 个机器上,有些机器(或其中一台)对用户的请求扮演协调者的角色,这个协调者保证 N 台机器的某几台对每个请求(特别是写请求)已经收到并且应答.当用户请求修改或者写入一个 Key 时,如果协调者保证至少 W 个副本(机器)已经收到数据的更新,当用户请求读取某个 Key 的数据时,如果协调者必须在 R 个副本已经给出同样值的情况下才能给出最后答案,那么当 $R+W>N$ 时,这样的 noSQL 系统提供了强一致性保证.为了支持高度的系统扩展能力,一般对一致性要求进行放松,有的应用场合,最终一致性约束已经足够.分布式系统的一致性必须通过一致性协议进行支持和保证,典型的一致性保证协议是 Paxos 协议,其他协议包括两阶段提交协议/三阶段提交协议(2PC/3PC)等,都是 Paxos 协议的变种,限于篇幅,本文不对 Paxos 协议进行深入的剖析.

HBase 建立在 HDFS 之上,由 HDFS 提供了强一致性的保证.在 HDFS 中,一个写操作只有到达 N(等于 2 或者 3)个备份之后才算成功,而读操作则由一个副本提供支持即可(R=1).为了避免写操作的堵塞作用,写入操作是异步的,即当所有副本接受到数据拷贝并应答以后,系统就把更新操作的结果开放给后续的操作.Dynamo, Voldemort,Cassandra 以及 Riak 等允许用户指定 N,R,W 等参数,根据用户的需要,甚至允许 $R+W\leqslant N$.这就意味着,用户可以获得强一致性或者最终一致性.为了把落后的数据副本尽快更新到最近状态,这些系统一般都提供必要的工具.这些工具基于版本技术和冲突解决技术实现.Dynamo 系统使用了一种向量时钟(vector clock)技术来标定不同节点对数据的修改,在一般情况下,利用向量时钟技术可以自动解决一些数据不一致问题;当系统不能自动解决某些版本不一致问题时,它把冲突交给应用程序来解决.比如,基于 Dynamo 的购物框,当两个版本不一致时,系统可以提示用户进行冲突解决.Voldemort 采用了类似的技术,当冲突发生时,如果系统不能自动解决,它把某个 Key 的多个数据版本一并返回给用户,让用户选择.Cassandra 给每个 Key 保存了一个时间戳(timestamp),当两个数据副本不一致时,以最近的时间戳为准.这种技术无法允许用户对数据进行合并.Riak 则提供了 Voldemort 和 Cassandra 技术供用户进行选择.CouchDB 在后台标定冲突情况,允许用户查询冲突数据进行人工修正,当修正完成后,系统就能给用户提供最终的数据版本.Big Table 使用 Chubby 分布式加锁系统来处理系统失败情况和保证数据的一致性(Hadoop 开源平台的 Zoo Keeper 子项目,是 Chubby 的开源实现). Chubby 的核心技术即是 Paxos,从工程实现的角度,Chubby 对 Paxos 的细节进行了补充,包括对避免拜占庭式(Byzantine)容错问题的考虑,是一个强壮的分布式系统的一致性保证协议.

(5) noSQL 的事务处理

在事务的语义方面,noSQL 技术为了支持强大的扩展能力而放弃了 ACID 约束,但是一般都能保证单个 Key 处理符合串行处理的约束,避免某个 Key 的数据因为并发的操作而受到损坏,出现不一致状况.对于大多数应用来说(比如电信日志处理、网站点击流处理等),这样的事务语义保证已经足够,一般不会带来不良的后果.而对于更加复杂的应用,比如网站购物车的更新、社交网络中好友关系的维护等,程序员需要精心设计程序,保证数据的一致性.Mega-Store[47]和 G-Store[48]两个原型系统则通过增强的协议,把多个 Key 组成一个 Key 群组(key group),通过一致性保证协议提供多 Key(multi-key)的事务处理语义.

3.2 面向分析型应用的noSQL技术(MapReduce)

面向分析型应用的 noSQL 技术主要包括 MapReduce 和 Dryad(由于 MapReduce 技术的大流行,微软已经彻底关闭 Dryad 项目,转而全面支持 MapReduce 技术(http://www.zdnet.com/blog/microsoft/microsoft-drops-dryad-puts-its-big-data-bets-on-hadoop/11226)).MapReduce 技术是由 Google 公司提出来的,旨在解决大规模非结构化数据快速批量处理的并行技术框架[49].MapReduce 在设计之初,致力于通过大规模廉价服务器集群实现大数据的并行处理.MapReduce 技术框架包含 3 个方面的内容:(1) 高度容错的分布式文件系统;(2) 并行编程模型;

(3) 并行执行引擎.MapReduce并行编程模型,其计算过程分解为两个主要阶段,即Map阶段和Reduce阶段.Map函数处理Key/Value对,产生一系列的中间Key/Value对;Reduce函数合并所有具有相同Key值的中间键值对,计算最终结果.用户只需编写Map函数和Reduce函数,MapReduce框架在大规模集群上自动调度执行编写好的程序,扩展性、容错性等问题由系统解决,用户不必关心.

自从2004年Google首次发布该技术以来,MapReduce技术表现出了强大的穿透力(如图6所示):首先,工业界意识到了MapReduce的价值,一批新公司围绕MapReduce技术创建起来,提供大数据处理、分析和可视化的创新技术和解决方案;接着,并行计算研究领域迎来了第一波研究热潮(2006年~2009年);数据库研究领域紧随其后(2009年~2012年),掀起了另外一波研究热潮.随着MapReduce技术的影响力不断扩大,传统数据库厂家,包括强烈反对noSQL/MapReduce技术的一些厂家(如Oracle,VoltDB,Microsoft等)纷纷发布Big Data技术和产品战略. Oracle公司在2011年上半年还对MapReduce技术不以为然,而在2011年下半年则发布了Big Plan,全面接受和推广MapReduce技术.2009年,Dewitt宣称永远不会在任何产品中集成Hadoop的代码,微软拒绝了MapReduce技术.到了2010年和2011年,微软开始热情拥抱MapReduce技术;2012年,更是关闭了自己的Dryad项目(与MapReduce类似的大规模并行计算框架技术).反对MapReduce最为强烈的Mike Stonebraker,由其担任CTO的VoltDB数据库公司在2011年发布了VoltDB向Hadoop集成的产品策略,这从侧面说明了Stonebraker态度的改变,也说明了VoltDB不能解决所有问题,有些问题使用Hadoop解决更好.围绕MapReduce技术,新的数据分析生态系统正在形成,Hadoop技术成为大数据分析的事实标准.

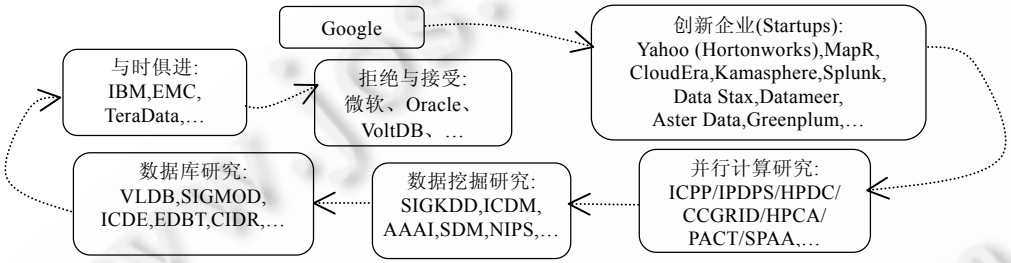


Fig.6 Penetration of MapReduce

图6 MapReduce技术的穿透力

MapReduce技术为大数据分析而生,以其高度的扩展性和容错性呈现出强大的生命力,获得了工业界和学术界的广泛关注.各大公司和各研究机构都投入力量,基于MapReduce框架展开一系列的研究.这些研究包括:

(1) 存储优化与数据类型支持,包括赋予MapReduce结构化存储模型(行存储和列存储),以便MapReduce能够有效处理结构化数据[50];为MapReduce的数据处理提供索引支持[51];对数据块的放置策略进行改进,加快连接操作[52];处理数据倾斜问题[53];利用MapReduce框架处理高维时空数据[54,55]、多媒体数据[56]等更丰富的数据类型.

(2) 对MapReduce框架的扩展,包括利用MapReduce框架处理流数据[57],利用MapReduce框架实现增量和持续的数据分析[58,59]以及对MapReduce进行扩展以便支持迭代式计算[60],充分利用大内存实现高性能数据分析[61]等.

(3) 连接算法以及查询算法优化,包括各种连接算法的优化[62−66]和查询算法优化[67].

(4) 调度算法优化,包括面向多核CPU[68,69]、GPU[70,71]、异构系统[72,73]以及云平台[74]等不同环境的调度策略[75].这些调度策略和执行算法的研究极大地提高了MapReduce的执行效率,改变了MapReduce性能低下的不足.

(5) 丰富MapReduce应用接口.为了帮助开发者和用户方便地使用MapReduce平台进行算法编写,研究人员还致力于应用编程接口——包括SQL接口[76]和统计分析、数据挖掘、机器学习编程接口的研发,比如面向操作型应用的HBase、面向分析型应用的Hive[77],Pig[78],System ML[79],Mahout[80]等.

System ML 是编写机器学习程序的声明性语言,提高了应用程序编写的效率.

(6) MapReduce 计算框架的安全[81]与节能[82]问题研究.

对 MapReduce 研究现状和进展的较为全面的介绍参见文献[83,84],这里仅介绍其中的几项工作.

在存储层面,研究人员借鉴关系数据库发展过程中积累的行存储、列存储、索引技术等经验,为 MapReduce 有效处理各类数据(包括结构化数据)展开了研究.这里介绍 RCFile[85],Wisconsin Columnar Storage[86], Hadoop++[51]等几项有特色的技术.RC File 技术是由 Ohio 州立大学、中国科学院、Facebook 公司合作研发的面向 Hadoop 平台的行列存储模型.RCFile 基于 HDFS 的块结构,维持 Hadoop 系统的扩展性和容错性不变,但是赋予 HDFS 的数据块更加精细的结构.这个工作借鉴了 RDBMS 的 PAX 存储技术,首先对大表进行横向划分,以便放入 HDFS 约定大小的数据块;然后在数据块里实现列存储,由于使用列存储,非常有利于对数据进行压缩,节省存储空间.该技术已经在 Facebook 得到了实际应用.Wisconsin Madison 大学提出了基于 MapReduce 平台的纯列存储模型,获得了比 RCFile 更高的性能.Hadoop++则通过嵌入索引寻址机制加快数据访问,从而提高数据处理的性能.在连接算法方面,著名数据库技术专家 Ullman 研究了面向 MapReduce 平台的两路、多路、模糊连接算法[62−64],Spyros Blanas 等人研究了面向日志处理的不同连接算法[65],Alper Okcan 则针对 Sita 连接[66]提出了有效的解决方案.来自 Massachusetts Amherst 大学的 Diao Yanlei 团队则基于 MapReduce 平台,利用 Hash 技术实现了流数据的实时(real time)处理[87].

经过研究人员的努力,大量的数据处理和分析算法被移植到 MapReduce 平台上,包括简单汇总和报表、OLAP 处理和多维分析[88]、数据挖掘[89]、人工智能和机器学习[90]、信息检索[91]、文本分析和情感计算[92]、科学数据处理(Bioinformatics/Physics)[93]、社会计算和图分析[94]等,并且,算法的性能得到不断的优化.

MapReduce 的应用领域不断扩展,大型银行(比如 JP Morgan Chase)、电信公司、科学研究机构(物理研究、生物研究等)都对 MapReduce 技术表现出浓厚的兴趣.

MapReduce 技术是一个巨大的倒退,还是一个重要的创新?

2008 年,数据库界著名的专家 Stonebraker(来自 MIT)和 Dewitt(来自微软)发布了《MapReduce: Major step backwards》博文,对 MapReduce 进行了批评,指出 MapReduce 技术没有模式、没有索引、依靠蛮力对数据进行处理,缺点一大堆,是一个倒退,毫无创新可言,是数据库界 25 年前已经淘汰的技术,注定没有前途.两位专家的观点掀起了激烈的争论.

20 世纪 70 年代初,关系数据库技术第一次提出来的时候,受到层次、网状数据库的拥护者(包括知名专家)的强烈批评.那次争论的结果是,关系数据库技术迎来了 40 年的大发展,确立了事务处理领域的霸主地位.近几年来,MapReduce 技术以其强大的生命力穿透了工业界和学术界,成为大数据分析的事实标准,是“21 世纪的瑞士军刀”(Media Guardian 媒体集团 2011 年授予 Hadoop 团队“年度创新大奖”的专家评语).

Stonebraker 和 Dewitt 是两位各自做出重大贡献的专家,其中,Stonebraker 是众多数据库技术、产品和公司的发明者和创立者,包括 PostgreSQL,Vertica,VoltDB,StreamBase,ScienceDB 等.这些技术和产品正好验证了 Stonebraker 早在 2005 年就提出来的“One Size cannot Fit All”的论断.比如,面向科学数据处理的 ScienceDB,采用多维数组(非关系数据模型)进行科学数据的管理,提供了超越 SQL 的复杂数据分析功能,包括协方差分析、矩阵奇异值分解等.Jim Gray 是另一位做出过重大贡献的数据库技术专家.早在 2005 年,在其论文《Scientific Data Management in the Coming Decade》[95]中就预见到,在接下来的 10 年中,MapReduce 技术将得到广泛的应用.

鉴于 SQL 语言的简单性及其已经被厂家和用户广为接受,面向操作型应用和面向分析型应用的 noSQL 数据库,一般提供 SQL 或者类似 SQL 的查询语言接口,方便用户进行数据的操作和简单汇总,比如 Hbase,Hive 等.

4 数据管理技术的新格局

图 7 从两个维度入手,即应用维度(操作型应用/分析型应用)和数据模型维度(关系模型/各种 noSQL 数据模型),展现了当前数据管理技术的竞争格局.

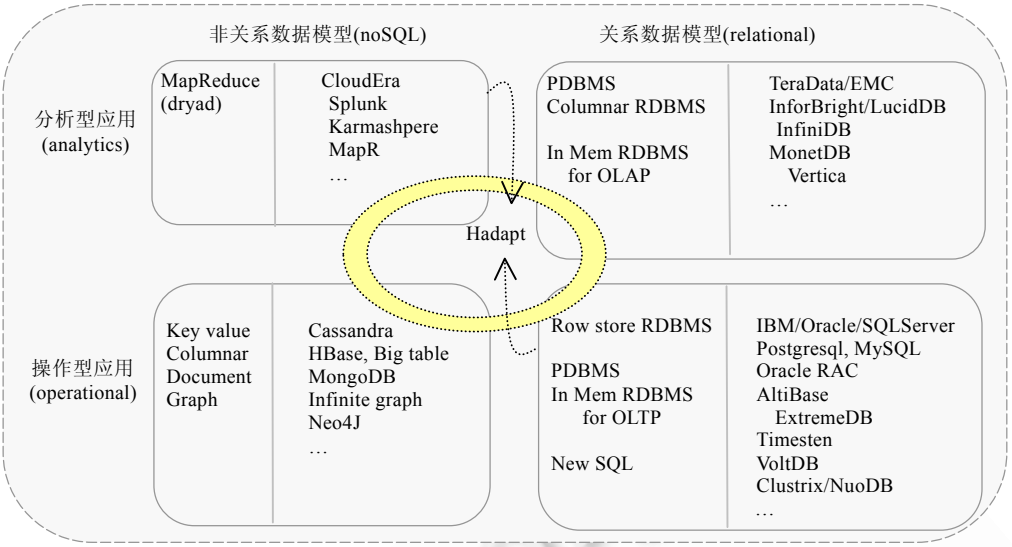


Fig.7 New landscape of data management technologies

图 7 数据管理技术的新格局

4.1 新的竞争格局

4.1.1 面向操作型应用的关系数据库技术

首先,传统的基于行存储的关系数据库系统,比如 IBM 的 DB2、Oracle、微软的 SQL Server 等,提供了高度的一致性、高度的精确度、系统的可恢复性等关键的特性,仍然是事务处理系统的核心引擎,无可替代;其次,面向实时计算的内存数据库系统,比如 Altibase,Timesten,Hana 等,通过把数据全部存储在内存里,并且针对内存存储进行了并发控制、查询处理和恢复技术的优化,获得了极高的性能,在电信、证券交易、军工、网络管理等特定领域获得了广泛应用,提供比磁盘数据库快 1 个数量级(10 倍、20 倍,甚至 30 倍)的性能.此外,以 VoltDB 为代表的新的面向 OLTP 应用的数据库系统(new SQL)采用了若干颠覆性的实现手段,包括串行执行事务(避免加锁开销)、全内存日志处理等(加速持久化过程)[6],获得了超过磁盘数据库 50 倍~60 倍的事务处理性能.其他宣称保持了 ACID 特性,同时具有 noSQL 扩展性的 newSQL 技术还有 Clustrix(http://www.clustrix.com/)和 NuoDB (http://www.nuodb.com/)等.

4.1.2 面向分析型应用的关系数据库技术

TeraData 是数据仓库领域的领头羊.TeraData 数据库采用了 Shared Nothing 的体系结构,支持较高的扩展性(参见第 4.2 节描述的 TeraData 对 Aster Data 的收购和战略的变化).面向分析型应用,列存储数据库的研究形成了一个重要的潮流.列存储数据库以其高效的压缩、更高的 I/O 效率等特点,在分析型应用领域获得了比行存储数据库高得多的性能.MonetDB 是一个典型的列存储数据库系统,此外还有 InforBright,InfiniDB,LucidDB, Vertica,SybaseIQ 等.MonetDB 和 Vertica 是基于列存储技术的内存数据库系统,主要面向分析型应用.

4.1.3 面向操作型应用的 noSQL 技术

noSQL 数据库系统相对于关系数据库系统具有两个明显的优势.一个是数据模型灵活、支持多样的数据类型(包括图数据),使用关系数据库系统对图数据进行建模、存储和分析,其性能、扩展性等很难与专门的图数据库抗衡(MySQL 和 Neo4J 的性能比较见表 1(无标度网络是度的分布符合幂律(power law)的网络),http://markorodriguez.com/2011/02/18/mysql-vs-neo4j-on-a-large-scale-graph-traversal/,http://www.slideshare.net/ClaudioMartella/presentation-7398682).另一个优势是高度的扩展性.从来没有一个关系数据库系统部署到超过 1 000 个节点的集群上,而 noSQL 技术通过灵活的数据模型、新的一致性约束机制,在大规模集群上获得了极高的性

能.比如,HBase 一天的吞吐量超过 200 亿个写操作(参见第 3.1.2 节),这个结果是在完全保证持久性的情况下获得的,也就是说,数据已经到达硬盘,真正实现了大数据的有效处理.基于关系模型的内存数据库系统也能获得极高的性能,但是在持久性完全保证的情况下,这个性能(写入性能)是要打折扣的.同时,其扩展性目前无法与 noSQL 匹敌,尚无法应付大数据的实时处理挑战.

Table 1 Performance comparison of graph data processing (MySQL vs. Neo4J)

表 1 对图数据进行处理的性能比较(MySQL vs. Neo4J)

一个基准评测(遍历操作)			
- 100 万个顶点 - 400 万条边 - 无标度网络 - 使用 Hash 索引和 BTree 索引 - MySQL vs. Neo4J	深度	关系数据库(ms)	图数据库(ms)
	1	100	30
	2	1 000	500
	3	10 000	3 000
	4	100 000	50 000
	5	N/A	100 000

操作型应用是一个比事务处理具有更广泛外延的概念,某些操作型应用并不需要 ACID 这样高强度的一致性约束,但是需要处理的数据量极大,对性能的要求也很高,必须依赖于大规模集群的并行处理能力来实现数据处理,弱一致性或者最终一致性是足够的.在这些应用场合,操作型 noSQL 技术大有用武之地.Facebook 从使用 RDBMS(MySQL)到弃用 RDBMS 转向 HBase,最后持续改进 HBase,作为其操作型应用数据处理架构的基础技术,具有标本意义.

4.1.4 面向分析型应用的 noSQL 技术

MapReduce 技术以其创新的设计理念、高度的扩展性和容错性,获得了学术界和工业界的青睐,围绕 MapReduce 的数据分析生态系统正在形成.近 5 年来,一批新公司(startup)涌现出来.比如提供 Hadoop 开源版本、相关培训和支持服务的 Cloudera 公司,提供具有更高性能的分布式文件系统的 MapR 公司,提供围绕 Hadoop 的完整工具套件的 Karmashpere 公司,致力于 Postgres 和 Hadoop 的集成、脱胎于 HadoopDB 大学项目的 Hadapt 公司,从 Yahooo 剥离出来、致力于 Hadoop 技术的研发和商用的 Hortonworks 公司等.与此同时,传统数据库厂商和数据分析套件厂商纷纷发布基于 Hadoop 技术的产品发展战略,这些公司包括 Microsoft,Oracle,SAS,IBM 等.微软公司已经关闭 Dryad 系统,全面投入 MapReduce 的研发.Oracle 在长期轻视 noSQL 和 MapReduce 技术以后,于 2011 年下半年发布 Big Plan 战略计划,全面进军大数据处理领域(当然少不了 Hadoop 技术).IBM 则早已捷足先登,IBM 的“沃森(Watson)”计算机,其软件系统即是基于 Hadoop 技术开发的,与此同时,IBM 发布了 Big Insights 计划,基于 Hadoop,Netezza 和 SPSS(统计分析、数据挖掘软件)等技术和产品构建大数据分析处理的技术框架.虽然有些公司的集成方案是肤浅的,只提供数据交换连接器[96],但是大家几乎不约而同地形成一个共识,那就是深度的集成方案是技术发展的方向.比如,TeraData 公司在 SIGMOD 2010 会议上展示了 Hadoop 和 TeraData 的连接器之后,毅然收购 Aster Data 公司,获得其 MapReduce 技术研发经验和复杂分析软件包,进而在新的产品计划中准备深度整合 Aster Data(SQL/MapReduce)和 TeraData,构造一个结构化数据和非结构化数据的统一处理平台.

4.2 各类技术的互相借鉴和发展

需要注意的是,上文对各类技术的划分是一种观察问题的角度.在具体的研究和开发实践中,理论界的研究人员和工业界的工程师不仅继续发展已有的技术和平台,同时不断地借鉴来自其他研究和技术的创新思想改进自身,或者提出兼具若干技术优点的混合技术架构.

2005 年以来,在数据管理技术领域最值得注意的两家新公司(startup)是 Aster Data 和 Greenplum,它们的技术手段相似,都是利用 MapReduce 技术对 PostgreSQL 数据库进行改造,使其可以运行在大规模集群上(MPP/Shared nothing),实现数据的高效处理,两者形成直接竞争关系.这两家公司的三板斧分别是 PostgreSQL 数据库、Shared Nothing 架构和 MapReduce 技术.Aster Data 和 Greenplum 数据库的核心引擎不仅能够执行 SQL 查询,同时也作为 MapReduce Job 的执行引擎,在同一架构下实现了 SQL/MapReduce 的统一处理.本质上,这两家公司

的技术都是在 RDBMS 的框架下实现了 MapReduce 技术.目前,Greenplum 公司已经被 EMC 收购,这一收购使 EMC 公司即刻获得了数据管理和分析的能力和产品.一夜之间,数据库市场上增加了一个重量级的竞争者(big player),给 Oracle 和 TeraData 形成巨大的压力.Aster Data 则被数据仓库领域的领头羊 TeraData 收购.我们可以思考,TeraData 收购 Aster Data,难道是想获得它的 PostgreSQL 数据库吗?还是想要获得它的 Shared Nothing 架构?其实,这两项技术 TeraData 是不缺的,TeraData 看中的是 Aster Data 的 MapReduce 技术经验及其分析软件包(基于 MapReduce 技术改写的并行数据分析软件包).目前,TeraData 正在着力于这样的任务,就是如何在 TeraData 中整合 MapReduce 技术,这是 RDBMS 对 MapReduce 技术的学习.

MapReduce 领域对 RDBMS 技术的借鉴是全方位的,包括存储、索引、查询优化、连接算法、应用接口、算法实现等各个方面.在这里再次提及的是 RCFile 系统.该技术在 HDFS 的存储框架下保留了 MapReduce 的扩展性和容错性(扩展性是大数据处理系统的首要要求,高度扩展的大规模集群必须关注容错保证),赋予 HDFS 数据块类似 PAX 的存储结构,通过借鉴 RDBMS 技术提高了 Hadoop 系统的分析处理性能,这是从 MapReduce 阵营借鉴 RDBMS 的技术和思想.该工作虽然不是原始创新,却是在 Hadoop 平台上实现列存储的第一项工作,被广为引用.更重要的是,RCFile 在 Facebook 公司获得了应用,解决了实际问题,是工业界和学术界协同创新的范例.

HadoopDB[97,98]是富有特色的一项工作.HadoopDB 试图把 MapReduce 的优点和 RDBMS 的优点结合起来.该系统具有清晰的两层结构,在上层利用 Hadoop 平台支持高度的系统扩展性能,在下层利用 RDBMS (PostgreSQL)实现数据的高性能处理.此外,HadoopDB 利用 HDFS 实现非结构化数据的处理,打造结构化数据和非结构化数据的统一处理平台.目前,HadoopDB 已经进行商业化(hadapt),首轮融资获得近 1 000 万美元投资,说明风险投资看好它的商业前途.仔细分析,HadoopDB 存在若干局限性,把数据保存到 PostgreSQL 表中,丧失了 Hadoop 的容错特性,要获得足够的容错性能,必须对 Hadoop 系统进行必要的修改;数据按照某个属性列进行划分后保存到 PostgreSQL 数据库,当连接查询涉及其他数据列时,HadoopDB 的性能将受到严重影响.目前,HadoopDB 系统也在不断改进和完善之中[99].

noSQL 技术的高度扩展性能给人们留下了深刻印象.VoltDB 团队当然没有无动于衷,他们的主要工作之一是根据数据模型对数据进行适当分区[100],即利用数据库表的主外键关系对数据进行分区,把数据分布到多个节点,尽量在 1 个节点上满足查询的要求,减少节点间的数据交换,并且避免数据倾斜,从而提高系统的扩展性.VoltDB 不仅保持传统关系数据库的优点,包括易用的 SQL 接口、严格的关系模型、完全的 ACID 保证等,同时通过增强的扩展能力,应对大数据处理的挑战.Stonebraker 统称 VoltDB 所使用的若干创新技术(包括数据分区方法、全内存日志处理、新的并发控制协议等)为 newSQL,并预测 newSQL 将在 2012 年流行开来(http://siliconangle.com/blog/2011/12/29/newsql-will-prevail-in-2012-says-mits-michael-stonebraker/).他们的预测是否准确有待验证,但是我们看到,noSQL 技术突飞猛进的发展促进了 RDBMS 的自我革新,即 SQL→noSQL→newSQL,以便应对大数据处理的新挑战.

各种技术将继续互相借鉴,持续发展(如图 8 所示).面向事务处理的 RDBMS 技术和面向操作型应用的 noSQL 技术将各自获得新的发展.在分析型应用方面,在围绕 RDBMS 的生态系统旁边生长出了围绕 Hadoop 技术的新生态系统,这两个生态系统的目标是一致的,即数据分析.关系数据库技术经过几十年的积淀和发展,擅长结构化数据的处理,性能高,但遇到扩展能力的困难;而 MapReduce 技术则在系统的扩展能力、数据的多样性、数据装载速度、分析和数据紧密结合/靠近数据进行分析(in situ data analytics)、分析的复杂度等方面见长.我们认为,在不久的将来,两项技术和生态系统最终走向融合,一个基于 HDFS/Hadoop 技术的大数据统一分析平台(unified big data analytics platform)和生态圈将最终形成.这个框架的核心技术是 HDFS/Hadoop,包括关系数据模型和查询处理技术被借鉴过来,溶解在这个框架里.该框架不仅提供关系数据的存储、处理和分析,还能采集、处理和分析类型多样的数据,并且在各个类型数据之间建立必要的联系(bridging unstructured and structured data),对其进行联合分析(together analysis),获得更全面的数据分析结果.在不久的将来,来自 TeraData(aster data),EMC(greenplum),Hadapt(HadoopDB)等公司的大数据分析技术方案,经过不断演化,将生长得越来越彼此相似(参见第 4.3.2 节).值得注意的是,有些研究试图在 RDBMS 领域把 OLTP 和 OLAP 结合起来,通过一个统一

的存储模型和查询处理框架实现事务处理和多维分析[101,102].目前,这种尝试尚未成为主流.

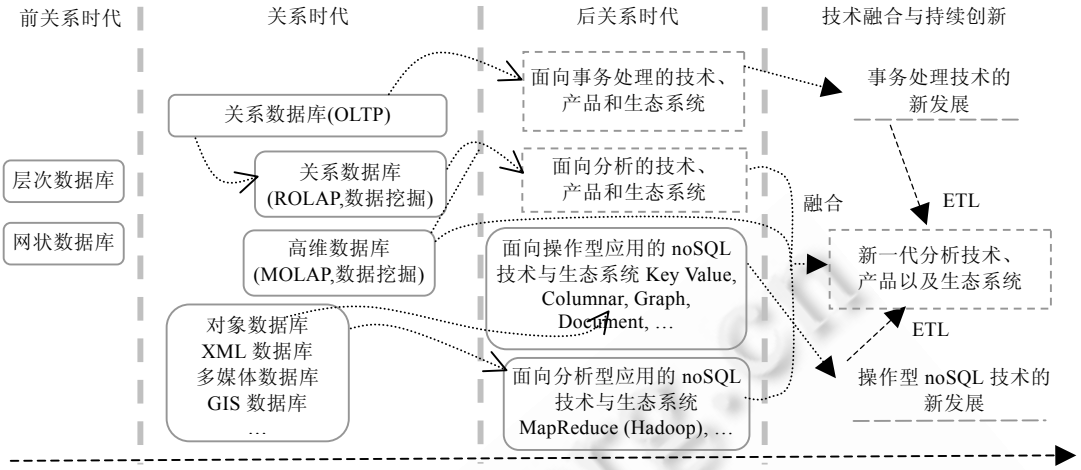


Fig.8 Mutual absorption, fusion and continuous development of various technologies

图 8 各类技术的互相借鉴、融合和持续发展

4.3 新的研究机会

面向数据管理的新挑战,研究和开发工作展现出空前繁荣的景象.研究的主题始终是更高的性能、更多样的数据和更复杂的分析.下文仅列举两个我们认为重要而且有趣的研究.

4.3.1 面向云平台的数据管理技术[103]

云平台整合了计算、存储、网络等硬件资源,通过虚拟化技术供用户共享,使得用户以更加经济的方式使用这些资源.云计算的主要特点是虚拟化和动态伸缩性.利用云平台实现面向操作型应用和分析型应用的数据管理仍然面临众多的挑战,比如,如何在具有动态伸缩性(elastic)的云平台上支持事务处理(transaction processing on the cloud)、支持云平台部署的新的存储模型[104]、数据的放置和容错[105]、支持云平台部署的新的索引结构、面向云平台的查询处理技术、如何把数据库当作一个服务部署到云平台上等,有待继续研究和应用验证.面向云平台的数据管理还要解决节能[106]、隐私保护和安全等关键问题才能使用户对其产生信任,真正实现大规模的产业化.

4.3.2 大数据统一处理平台

正如上文已经分析的,围绕 RDBMS 已经产生了一个分析生态系统,而围绕 Hadoop 技术正在产生另外一个分析生态系统.这两个系统的目的是重叠的,一个自然的想法就是,两项技术和生态系统是否将融合到一起?我们相信,经过理论界和工业界的努力,一个统一的大数据处理框架以及生态系统将会形成.仍然有些问题期待更深入的研究,比如,如何把类型多样的数据整合到一个存储层上(数据组织方式);如何建立更加智能的存储层(存储模型、数据压缩、数据去重、索引技术、查询的路由选择、在存储层执行一定的计算等);如何改进查询的调度算法、执行算法,使其适应多核、GPU、异构环境、超级计算机、廉价集群等硬件环境,并能够运行在云平台上;如何超越 SQL,提供扩展的编程和应用接口,为各类用户(包括普通用户、高级用户、统计学家、数学家等)提供灵活多样的数据分析环境和工具;如何对大数据的处理结果进行可视化;如何引导用户对大数据进行探索式的查询(Exploratory Query),以便建立分析模型,然后进行深入分析.

下一代 MapReduce(MapReduce 2.0/YARN)技术对系统的扩展性、性能、可用性等各个方面做了大幅度的调整和增强.通过把 MapReduce 计算模型从资源管理架构中剥离出来,新的框架支持更多的应用类型(通过 Application Manager 实现),包括流数据处理(stream processing)、图数据处理(graph processing)、BSP 计算模型(bulk synchronous processing)、MPI 计算模型(message passing interface)等(http://www.businesswire.com/news/

home/20120119005825/en/Hortonworks-Deliver-Next-Generation-Apache-Hadoop).其支持的节点规模从目前的4 000左右增加到6 000~10 000,并发的任务数从目前的40 000增加到100 000.这样的框架扩展和改进,预示着大数据统一处理平台初见端倪.利用内存处理技术,使得MapReduce可以支持数据的交互式处理[107].

4.4 我们的研究

4.4.1 多核GPU上的OLAP[108]

在OLAP应用中,立方体(cube)的计算是最耗时的操作,已经被研究人员深入地研究.我们的工作之一是通过设计并行的算法,利用GPU强大的多核处理能力加快Cube的计算过程.我们首先提出一种集成自底向上策略和宽度优先的分区顺序的、Cache感知的算法CC-BUC(cache conscious bottom up computation),分别处理每个维度的信息.在每个维度信息的处理过程中,宽度优先的扫描操作减少了内存I/O,提高了Cache的局部性(locality),降低了Cache缺失率.基于CC-BUC算法,我们提出了完整的Cube计算方法MC-Cubing.不仅各个分区可以并行处理,每个分区内部也可以进行并行处理,适应多核架构的硬件处理能力(包括多核CPU和GPU),具有很高的并行性.同时,我们调整微观数据结构,充分利用SIMD指令(单指令多数据)实现数据的处理.MC-Cubing已经在多核CPU和GPU上实现.实验结果显示,在实际数据集上,我们的算法比普通的BUC算法获得了6倍以上的性能提升.

4.4.2 社交网络与Graph分析[109,110]

社交网络的发展引起了人们的研究兴趣,对图进行分析是重要的基础研究.我们选择在一个大图中衡量节点之间的相似度(SimRank)作为我们的一个研究支点.已有的SimRank计算方法有两个局限:1) 计算代价太大;2) 只能应用到静态图上.我们的研究是:首先,利用GPU内在的并行性和高带宽设计新的并行算法,加速大图(large graph)上的SimRank计算;其次,SimRank计算本质上是一个一阶Markov链问题,基于这样的观察,我们专门设计了并行算法,把Markov链的耦合度打开,通过迭代式聚集实现节点相似度打分计算.这种算法可以应用到动态图上,不仅能够处理连接更新,而且能够处理节点更新.在人工数据集和实际数据集上进行的一系列实验验证了该算法的有效性和效率.

4.4.3 高度可扩展的OLAP[111]

面向多维分析的数据仓库一般按照星型模型进行数据建模.在星型模型中,事实表的数据量一般比较大,而维表的数据量一般较小.为了解决OLAP应用的数据仓库数据膨胀问题,我们考虑使用大规模集群进行处理.如果把事实表和维表都分布到集群上,则执行查询时连接操作将引发大量的网络传输;如果把维表全复制到各个节点,则对于大规模集群来说,又将带来庞大的空间开销.我们的策略是:首先,把各个维表的维度信息进行层次编码,代替事实表的外键;然后,把事实表横向分割,分布到大规模集群上,以便并行处理.在各个节点上,无须参考维表信息进行查询处理(即无需连接操作),各个节点的局部聚集结果由主节点进行合并,生成最终的结果集.这个特性使得该技术具有极高的扩展能力.主节点上的最终结果需要与维表进行连接,以便反向查找相关信息.由于最终结果集一般较小,连接操作的性能得到了保证.我们已经基于PostgreSQL进行了技术原型LinearDB的实现,正在Hadoop框架下移植层次编码方法及其查询处理方法,充分利用Hadoop的节点管理能力和扩展性,提高数据仓库星型查询的性能.初步实验结果显示(实验基于14台普通PC,其中1台作为name node),新的原型系统Dumbo在500GB SSB(star schema benchmark)数据集上取得了比HadoopDB接近1个数量级到超过1个数量级(不同查询的加速效果不一样)的性能提升.

5 总 结

大数据时代正在向我们走来,数据管理技术研究进入了新的阶段.本文追随技术进步的历史脉络,介绍了数据管理技术的发展进程,展开当今数据管理技术的新画卷.本文试图标定变革时期[112]几个值得研究的重要问题.最后,通过本文与读者分享我们的研究及其背后的思想.

致谢 感谢《软件学报》评审专家细心、专业的评审,使得本文的表述更加清楚、准确和客观.感谢EMC中国

研究院周宝曜博士和曹逾博士、人人网首席科学家陈继东博士对本文提出的建议,使得本文内容更加全面、结构更加合理.感谢中国人民大学新闻学院韦英平博士对文章的润色,使得文章更加容易阅读.

References:

[1] Abadi DJ, Boncz PA, Harizopoulos S. Column-Oriented database systems. VLDB 2009 Tutorial, 2009. http://cs-www.cs.yale.edu/homes/dna/talks/Column_Store_Tutorial_VLDB09.pdf

[2] Datta A, Thomas H. Querying compressed data in data warehouses. Journal of Information Technology and Management, 2002, 3(4):353−386. [doi: 10.1023/A:1019772807859]

[3] Bhuiyan MM, Hoque ASML. High performance SQL queries on compressed relational database. Journal of Computers, 2009, 3(12):1263−1274.

[4] O'Connell SJ, Winterbottom N. Performing joins without decompression in a compressed database system. SIGMOD Record, 2003,32(1):6−11. [doi: 10.1145/640990.640991]

[5] Olofson C. Feature: The database revolution. 2012. http://www.ibm.com/developerworks/data/library/dmmag/DMMag_2011_Issue1/FeatureHistory/

[6] Kallman R, Kimura H, Natkins J, Pavlo A, Rasin A, Zdonik S, Jones EPC, Madden S, Stonebraker M, Zhang Y, Hugg J, Abadi DJ. H-Store: A high-performance, distributed main memory transaction processing system. Proc. of the VLDB Endowment, 2008,1(2): 1496−1499.

[7] Lu HJ, Ng YY, Tian ZP. T-Tree or B-tree: Main memory database index structure revisited. In: Orlowska ME, ed. Proc. of the Australasian Database Conf. 2000. Canberra: IEEE Computer Society, 2000. 65−73. [doi: 10.1109/ADC.2000.819815]

[8] Shatdal A, Kant C, Naughton JF. Cache conscious algorithms for relational query processing. In: Bocca JB, Jarke M, Zaniolo C, eds. Proc. of the VLDB'94. Chile: Morgan Kaufmann Publishers, 1994. 510−521.

[9] Luan H, Du XY, Wang S. Cache-Conscious data cube computation on a modern processor. Journal of Computer Science and Technology, 2009,24(4):708−722.

[10] Rao J, Ross KA. Cache conscious indexing for decision-support in main memory. In: Atkinson MP, Orlowska ME, Valduriez P, Zdonik SB, Brodie ML, eds. Proc. of the VLDB'99. Edinburgh: Morgan Kaufmann Publishers, 1999. 78−89.

[11] He BS, Luo Q. Cache-Oblivious query processing. In: Hellerstein J, Stonebraker M, Weikum G, eds. Proc. of the CIDR 2007. Asilomar: CIDR Program Committee, 2007. 44−55.

[12] He BS, Luo Q. Cache-Oblivious databases: Limitations and opportunities. ACM Trans. on Database Systems, 2008,33(2):1−42. [doi: 10.1145/1366102.1366105]

[13] Bender MA, Farach-Colton M, Fineman JT, Fogel YR, Kuszmaul BC, Nelson J. Cache-Oblivious streaming B-trees. In: Gibbons PB, Scheideler C, eds. Proc. of the SPAA 2007. Munich: ACM Press, 2007. 81−92. [doi: 10.1145/1248377.1248393]

[14] Pandis I, Tozün P, Johnson R, Ailamaki A. PLP: Page latch-free shared-everything OLTP. Proc. of the VLDB Endowment, 2011, 4(10):610−621.

[15] Lee RB, Ding XN, Chen F, Lu QD, Zhang XD. MCC-DB: Minimizing cache conflicts in multi-core processors for databases. Proc. of the VLDB Endowment, 2009,2(1):373−384.

[16] Bakkum P, Skadron K. Accelerating SQL database operations on a GPU with CUDA. In: Kaeli DR, Leeser M, eds. Proc. of the GPGPU 2010. Pittsburgh: ACM Int'l Conf. Proc. Series, 2010. 94−103. [doi: 10.1145/1735688.1735706]

[17] Lin CF, Yuan SM. The design and evaluation of GPU based memory database. In: Watada J, Chung PC, Ho KC, eds. Proc. of the Int'l Conf. on Genetic and Evolutionary Computing 2011. Kinmen, Xiamen: IEEE Computer Society, 2011. 224−231. [doi: 10.1109/ICGEC.2011.61]

[18] Govindaraju NK, Gray J, Kumar R, Manocha D. GPUTeraSort: High performance graphics coprocessor sorting for large database management. In: Chaudhuri S, Hristidis V, Polyzotis N, eds. Proc. of the SIGMOD 2006. Chicago: ACM Press, 2006. 325−336. [doi: 10.1145/1142473.1142511]

[19] He BS, Yu JX. HighThroughput transaction executions on graphics processors. Proc. of the VLDB Endowment, 2011,4(5): 314−325.

[20] Volk PB, Habich D, Lehner W. GPU based speculative query processing for database operations. In: Bordawekar R, Lang CA, eds. Proc. of the VLDB Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures 2010. Singapore: VLDB Endowment, 2010. Article No.7.

[21] Lee SW, Moon B. Design of flash-based DBMS: An in-page logging approach. In: Chan CY, Ooi BC, Zhou AY, eds. Proc. of the SIGMOD 2007. Beijing: ACM Press, 2007. 55−66. [doi: 10.1145/1247480.1247488]

[22] Koltsidas I, Viglas SD. Flashing up the storage layer. Proc. of the VLDB Endowment, 2008,1(1):514−525.

[23] Bonnet P, Bouganim L, Koltsidas I, Viglas SD. System co-design and data management for flash devices. VLDB 2011 Tutorial, 2011. http://www.vldb.org/pvldb/vol4/p1504-bonnet-tutorial2.pdf

[24] Qin XP, Xiao YQ, Cao W, Wang S. A parallel recovery scheme for update intensive main memory database systems. In: Rountree N, ed. Proc. of the PDCAT 2008. Dunedin: IEEE Computer Society, 2008. 509−516. [doi: 10.1109/PDCAT.2008.69]

[25] Lee J, Kim K, Cha SK. Differential logging: A commutative and associative logging scheme for highly parallel main memory database. In: Georgakopoulos D, Buchmann A, eds. Proc. of the ICDE 2001. Heidelberg: IEEE Computer Society, 2001. 173−182. [doi: 10.1109/ICDE.2001.914826]

[26] Stonebraker. How to do complex analytics. 2012. http://www.slideshare.net/MassTLC/mike1

[27] Ni MX, Luo WM. Technology revolution in the age of data exploding. CCF Communications, 2011,7(7):12−20 (in Chinese with English abstract).

[28] Brewer EA. Towards robust distributed systems. PODC 2000 Keynote Speech, 2000. http://openstorage.gunadarma.ac.id/~mwiryana/Kuliah/Database/PODC-keynote.pdf

[29] Lynch N, Gilbert S. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant Web services. ACM SIGACT News, 2002,33(2):51−59. [doi: 10.1145/564585.564601]

[30] FAL Labs. Tokyo cabinet: A modern implementation of DBM. 2012. http://fallabs.com/tokyocabinet/

[31] Citrusbyte. Redis database. 2012. http://redis.io/

[32] Voldemort Team. Voldemort database. 2012. http://project-voldemort.com/

[33] Oracle. Oracle Berkeley DB 11g. 2012. http://www.oracle.com/technetwork/products/berkeleydb/overview/index.html

[34] De Candia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W. Dynamo: Amazon's highly available key-value store. ACM SIGOPS Operating Systems Review, 2007,41(6):205−220. [doi: 10.1145/1323293.1294281]

[35] Apache Foundation. Apache Cassandra. 2012. http://cassandra.apache.org/

[36] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: A distributed storage system for structured data. In: Bershad BN, Mogul JC, eds. Proc. of the OSDI 2006. Seattle: USENIX Association, 2006. 15−22.

[37] Apache Foundation. Apache HBase. 2012. http://hbase.apache.org/

[38] Borthakur D, Gray J, Sarma JS, Muthukkaruppan K, Spiegelberg N, Kuang HR, Ranganathan K, Molkov D, Menon A, Rash S, Schmidt R, Aiyer A. Apache hadoop goes realtime at Facebook. In: Sellis TK, Miller RJ, Kementsietsidis A, Velegrakis Y, eds. Proc. of the SIGMOD 2011. Athens: ACM Press, 2011. 1071−1080. [doi: 10.1145/1989323.1989438]

[39] Apache Foundation. CouchDB. 2012. http://couchdb.apache.org/

[40] MongoDB Team. MongDB. 2012. http://www.mongodb.org/

[41] Riak Team. Riak database. 2012. http://basho.com/products/riak-overview/

[42] Neo4j Team. Neo4j database. 2012. http://neo4j.org/

[43] NetMesh. InforGrid Web graph database. 2012. http://infogrid.org/trac/

[44] Objectivity. InfiniteGraph database. 2012. http://www.infinitegraph.com/

[45] HyperGraphDB Team. HyperGraphDB. 2012. http://www.kobrix.com/hgdb.jsp

[46] Marcus A. The NoSQL EcoSystem. 2012. http://www.aosabook.org/en/nosql.html

[47] Baker J, Bond C, Corbett JC, Furman JJ, Khorlin A, Larson J, Leon JM, Li YW, Lloyd A, Yushprakh V. Megastore: Providing scalable, highly available storage for interactive services. In: Ailamaki A, Franklin M, Hellerstein J, eds. Proc. of the CIDR 2011. Asilomar: Online Proc., 2011. 223−234.

[48] Das S, Agrawal D, Abbadi AE. G-Store: A scalable data store for transactional multi key access in the cloud. In: Hellerstein JM, Chaudhuri S, Rosenblum M, eds. Proc. of the SOCC 2010. Indianapolis: ACM Press, 2010. 163−174. [doi: 10.1145/1807128.1807157]

[49] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. In: Brewer E, Chen P, eds. Proc. of the OSDI 2004. San Francisco: USENIX Association, 2004. 137−150.

[50] Kaldewey T, Shekita EJ, Tata S. Clydesdale: Structured data processing on MapReduce. In: Rundensteiner EA, Markl V, Manolescu I, Amer-Yahia S, Naumann F, Ari I, eds. Proc. of the EDBT 2012. Berlin: ACM Press, 2012. 15−25. [doi: 10.1145/2247596.2247600]

[51] Jindal A, Quiane-Ruiz JA, Dittrich J. Trojan data layouts: Right shoes for a running elephant. In: Chase JS, Abbadi AE, Babu S, Romano P, eds. Proc. of the SOCC. Cascais: ACM Press, 2011. Article No.21. [doi: 10.1145/2038916.2038937]

[52] Eltabakh MY, Tian YY, Ozcan F, Gemulla R, Krettek A, McPherson J. CoHadoop: Flexible data placement and its exploitation in Hadoop. Proc. of the VLDB Endowment, 2011,4(9):575−585.

[53] Ananthanarayanan G, Agarwal S, Kandula S, Greenberg A, Stoica I, Harlan D, Harris E. Scarlett: Coping with skewed content popularity in mapreduce clusters. In: Kirsch CM, Heiser G, eds. Proc. of the EuroSys 2011. Salzburg: ACM Press, 2011. 287−300. [doi: 10.1145/1966445.1966472]

[54] Ma Q, Yang B, Qian WN, Zhou AY. Query processing of massive trajectory data based on mapreduce. In: Meng XF, Wang HX, Chen Y, eds. Proc. of the CloudDB 2009. New York: ACM Press, 2009. 9−16. [doi: 10.1145/1651263.1651266]

[55] Chandramouli B, Goldstein J, Duan SY. Temporal analytics on big data for Web advertising. In: Kementsietsidis A, Salles MAV, eds. Proc. of the ICDE 2012. Washington: IEEE Computer Society, 2012. 90−101. [doi: 10.1109/ICDE.2012.55]

[56] White B, Yeh T, Lin J, Davis L. Web-Scale computer vision using MapReduce for multimedia data mining. In: Hua XS, Ngo CW, eds. Proc. of the KDD Workshop on Multimedia Data Mining 2010. New York: ACM Press, 2010. Article No.9. [doi: 10.1145/1814245.1814254]

[57] Neumeyer L, Robbins B, Nair A, Kesari A. S4: Distributed stream computing platform. In: Fan W, Hsu W, Webb GI, Liu B, Zhang CQ, Gunopulos D, Wu XD, eds. Proc. of the ICDM Workshops 2010. Sydney: IEEE Computer Society, 2010. 170−177. [doi: 10.1109/ICDMW.2010.172]

[58] Chen QM, Hsu M. Continuous mapreduce for In-DB stream analytics. In: Meersman R, Dillon TS, Herrero P, eds. Proc. of the OTM 2010. Hersonissos: Springer-Verlag, 2010. 16−34. [doi: 10.1007/978-3-642-16961-8_9]

[59] Olston C, Chiou G, Chitnis L, Liu F, Han YP, Larsson M, Neumann A, Rao VBN, Sankarasubramanian V, Seth S, Tian C, ZiCornell T, Wang XD. Nova: Continuous Pig/Hadoop workflows. In: Sellis TK, Miller RJ, Kementsietsidis A, Velegrakis Y, eds. Proc. of the SIGMOD 2011. Athens: ACM Press, 2011. 1081−1090. [doi: 10.1145/1989323.1989439]

[60] Bu YY, Howe B, Balazinska M, Ernst MD. HaLoop: Efficient iterative data processing on large clusters. Proc. of the VLDB Endowment, 2010,3(1-2):285−296.

[61] Zhang SB, Han JZ, Liu ZY, Wang K, Feng SZ. Accelerating MapReduce with distributed memory cache. In: Pan Y, ed. Proc. of the ICPADS 2009. Shenzhen: IEEE, 2009. 472−478. [doi: 10.1109/ICPADS.2009.88]

[62] Afrati FN, Ullman JD. Optimizing joins in a map-reduce environment. In: Manolescu I, Spaccapietra S, Teubner J, Kitsuregawa M, Léger A, Naumann F, Ailamaki A, Özcan F, eds. Proc. of the EDBT 2010. Lausanne: ACM Int'l Conf. Proc. Series, 2010. 99−110. [doi: 10.1145/1739041.1739056]

[63] Afrati FN, Ullman JD. Optimizing multiway joins in a Map-Reduce environment. IEEE Trans. on Knowledge and Data Engineering, 2011,23(9):1282−1298. [doi: 10.1109/TKDE.2011.47]

[64] Afrati FN, Sarma AD, Menestrina D, Parameswaran A, Ullman JD. Fuzzy joins using MapReduce. Technical Report, #1006, Stanford: Stanford University, InfoLab, 2012.

[65] Blanas S, Patel JM, Ercegovac V, Rao J, Shekita EJ, Tian YY. A comparison of join algorithms for log processing in MapReduce. In: Elmagarmid AK, Agrawal D, eds. Proc. of the SIGMOD 2010. Indianapolis: ACM Press, 2010. 975−986. [doi: 10.1145/1807167.1807273]

[66] Okcan A, Riedewald M. Processing theta-joins using MapReduce. In: Sellis TK, Miller RJ, Kementsietsidis A, Velegrakis Y, eds. Proc. of the SIGMOD 2011. Athens: ACM Press, 2011. 949−960. [doi: 10.1145/1989323.1989423]

[67] Nykiel T, Potamias M, Mishra C, Kollios G, Koudas N. MRShare: Sharing across multiple queries in MapReduce. Proc. of the VLDB Endowment, 2010,3(1-2):494−505.

[68] Chu CT, Kim SK, Lin YA, Yu YY, Bradski G, Ng AY, Olukotun K. Map-Reduce for machine learning on multicore. In: Schölkopf B, Platt JC, Hoffman T, eds. Proc. of the NIPS 2006. Vancouver: MIT Press, 2006. 281−288.

[69] Chen R, Chen HB, Zang BY. Tiled-MapReduce: Optimizing resource usages of data-parallel applications on multicore with tiling. In: Salapura V, Gschwind M, Knoop J, eds. Proc. of the PACT 2010. Vienna: ACM Press, 2010. 523−534. [doi: 10.1145/1854273.1854337]

[70] He BS, Fang WB, Govindaraju NK, Luo Q, Wang TY. Mars: A MapReduce framework on graphics processors. In: Moshovos A, Tarditi D, Olukotun K, eds. Proc. of the PACT 2008. Toronto: ACM Press, 2008. 260−269. [doi: 10.1145/1454115.1454152]

[71] Hong CT, Chen DH, Chen WG, Zheng WM, Lin HB. MapCG: Writing parallel program portable between CPU and GPU. In: Salapura V, Gschwind M, Knoop J, eds. Proc. of the PACT 2010. Vienna: ACM Press, 2010. 217−226. [doi: 10.1145/1854273.1854303]

[72] Polo J, Carrera D, Becerra Y, Beltran V, Torres J, Ayguade E. Performance management of accelerated MapReduce workloads in heterogeneous clusters. In: Qin F, ed. Proc. of the ICPP 2010. San Diego: IEEE Computer Society, 2010. 653−662. [doi: 10.1109/ICPP.2010.73]

[73] Papagiannis A, Nikolopoulos DS. Rearchitecting MapReduce for heterogeneous multicore processors with explicitly managed memories. In: Qin F, ed. Proc. of the ICPP 2010. San Diego: IEEE Computer Society, 2010. 121−130. [doi: 10.1109/ICPP.2010.21]

[74] You HH, Yang CC, Huang JL. A load-aware scheduler for MapReduce framework in heterogeneous cloud environments. In: Chu WC, Wong WE, Palakal MJ, Hung CC, eds. Proc. of the SAC 2011. Taichung: ACM Press, 2011. 127−132. [doi: 10.1145/1982185.1982218]

[75] Zaharia M, Borthakur D, Sarma JS, Elmeleegy K, Shenker S, Stoica I. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In: Morin C, Muller G, eds. Proc. of the EuroSYS 2010. Paris: ACM Press, 2010. 265−278.

[76] Chattopadhyay B, Lin L, Liu WR, Mittal S, Aragonda P, Lychagina V, Kwon YH, Wong M. Tenzing a SQL implementation on the MapReduce framework. Proc. of the VLDB Endowment, 2011,4(4):1318−1327.

[77] Thusoo A, Sarma JS, Jain N, Shao Z, Chakka P, Anthony S, Liu H, Wyckoff P, Murthy R. Hive a warehousing solution over a MapReduce framework. Proc. of the VLDB Endowment, 2009,2(2):938−941.

[78] Olston C, Reed B, Srivastava U, Kumar R, Tomkins A. Pig latin: A not-so-foreign language for data processing. In: Wang JTL, ed. Proc. of the SIGMOD 2008. Vancouver: ACM Press, 2008. 1099−1110. [doi: 10.1145/1376616.1376726]

[79] Ghoting A, Krishnamurthy R, Pednault E, Reinwald B, Sindhwani V, Tatikonda S, Tian YY, Vaithyanathan S. SystemML: Declarative machine learning on MapReduce. In: Abiteboul S, Böhm K, Koch C, Tan KL, eds. Proc. of the ICDE 2011. Hannover: IEEE Computer Society, 2011. 231−242. [doi: 10.1109/ICDE.2011.5767930]

[80] Apache Foundation. Mahout. 2012. http://mahout.apache.org/

[81] Roy I, Ramadan HE, Setty STV, Kilzer A, Shmatikov V, Witchel E. Airavat: Security and privacy for MapReduce. In: Castro M, Snoeren AC, eds. Proc. of the NSDI 2010. San Jose: USENIX Association, 2010. 297−312.

[82] Lang W, Patel JM. Energy management for MapReduce clusters. Proc. of the VLDB Endowment, 2010,3(1-2):129−139.

[83] Qin XP, Wang HJ, Du XY, Wang S. Big data analysis—Competition and symbiosis of RDBMS and MapReduce. Ruanjian Xuebao/Journal of Software, 2012,23(1):32−45 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4091.htm [doi: 10.3724/SP.J.1001.2012.04091]

[84] Lee KH, Lee YJ, Choi H, Chung YD, Moon BK. Parallel data processing with MapReduce: A survey. SIGMOD Record, 2011, 40(4):11−20. [doi: 10.1145/2094114.2094118]

[85] He YQ, Lee RB, Huai Y, Shao Z, Jain N, Zhang XD, Xu ZW. RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems. In: Abiteboul S, Böhm K, Koch C, Tan KL, eds. Proc. of the ICDE 2011. Hannover: IEEE Computer Society, 2011. 1199−1208. [doi: 10.1109/ICDE.2011.5767933]

[86] Floratou A, Patel JM, Shekita EJ, Tata S. Column-Oriented storage techniques for MapReduce. Proc. of the VLDB Endowment, 2011,4(7):419−429.

[87] Li BD, Mazur E, Diao YL, McGregor A, Shenoy P. A platform for scalable one-pass analytics using MapReduce. In: Sellis TK, Miller RJ, Kementsietsidis A, Velegrakis Y, eds. Proc. of the SIGMOD 2011. Athens: ACM Press, 2011. 985−996. [doi: 10.1145/1989323.1989426]

[88] Abelló A, Ferrarons J, Romero O. Building cubes with MapReduce. In: Song IY, Ordonez C, eds. Proc. of the DOLAP 2010. New York: ACM Press, 2010. 17−24. [doi: 10.1145/2064676.2064680]

[89] Bose JH, Andrzejak A, Hogqvist M. Beyond online aggregation: Parallel and incremental data mining with online Map-Reduce. In: Nambiar U, McPherson J, Konopnicki D, eds. Proc. of the WWW Workshop on Massive Data Analytics on the Cloud 2010. New York: ACM Press, 2010. Article No.3. [doi: 10.1145/1779599.1779602]

[90] Omalley O. See what Yahoo! and Jeopardy! have in common. 2012. http://developer.yahoo.com/blogs/hadoop/posts/2011/02/i%E2%80%99ll-take-hadoop-for-400-alex/

[91] Zhang CJ, Ma Q, Wang XL, Zhou AY. Distributed SLCA-based XML keyword search by Map-Reduce. In: Yoshikawa M, Meng XF, Yumoto T, Ma Q, Sun LF, Watanabe C, eds. Proc. of the DASFAA 2010. Tsukuba: Springer-Verlag, 2010. 386−397. [doi: 10.1007/978-3-642-14589-6_40]

[92] Lin J, Dyer C. Data-Intensive Text Processing with MapReduce. San Rafael: Morgan and Claypool Publishers, 2010. 40−56.

[93] Wiley K, Connolly A, Gardner JP, Krughof S, Balazinska M, Howe B, Kwon YC, Bu YY. Astronomy in the cloud: Using MapReduce for image coaddition. Publications of the Astronomical Society of the Pacific, 2011,123(901):366−380. [doi: 10.1086/658877]

[94] Bahmani B, Kumar R, Vassilvitskii S. Densest subgraph in streaming and MapReduce. Proc. of the VLDB Endowment, 2012,5(5):454−465.

[95] Gray J, Liu DT, Nieto-Santisteban M, Szalay A, DeWitt DJ, Heber G. Scientific data management in the coming decade. SIGMOD Record, 2005,34(4):34−41. [doi: 10.1145/1107499.1107503]

[96] Xu Y, Kostamaa P, Gao LK. Integrating Hadoop and parallel DBMS. In: Elmagarmid AK, Agrawal D, eds. Proc. of the SIGMOD 2010. Indianapolis: ACM Press, 2010. 969−974. [doi: 10.1145/1807167.1807272]

[97] Abouzeid A, Bajda-Pawlikowski K, Abadi D, Silberschatz A, Rasin A. HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads. Proc. of the VLDB Endowment, 2009,2(1):922−933.

[98] Abouzied A, Bajda-Pawlikowski K, Huang JW, Abadi DJ, Silberschatz A. HadoopDB in action: Building real world applications. In: Elmagarmid AK, Agrawal D, eds. Proc. of the SIGMOD 2010. Indianapolis: ACM Press, 2010. 1111−1114. [doi: 10.1145/1807167.1807294]

[99] Bajda-Pawlikowski K, Abadi DJ, Silberschatz A, Paulson E. Efficient processing of data warehousing queries in a split execution environment. In: Sellis TK, Miller RJ, Kementsietsidis A, Velegrakis Y, eds. Proc. of the SIGMOD 2011. Athens: ACM Press, 2011. 1165−1176. [doi: 10.1145/1989323.1989447]

[100] Pavlo A, Curino C, Zdonik S. Skew-Aware automatic database partitioning in shared-nothing, parallel OLTP systems. In: Candan KS, Chen Y, Snodgrass RT, Gravano L, Fuxman A, eds. Proc. of the SIMGOD 2012. Scottsdale: ACM Press, 2012. 61−72. [doi: 10.1145/2213836.2213844]

[101] Cao Y, Chen C, Guo F, Jiang DW, Lin YT, Ooi BC, Vo HT, Wu S, Xu QQ. ES2: A cloud data storage system for supporting both OLTP and OLAP. In: Abiteboul S, Böhm K, Koch C, Tan KL, eds. Proc. of the ICDE 2011. Hannover: IEEE Computer Society, 2011. 291−302. [doi: 10.1109/ICDE.2011.5767881]

[102] Kemper A, Neumann T. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In: Abiteboul S, Böhm K, Koch C, Tan KL, eds. Proc. of the ICDE 2011. Hannover: IEEE Computer Society, 2011. 195−206. [doi: 10.1109/ICDE.2011.5767867]

[103] Lin ZY, Lai YX, Lin C, Xie Y, Zou Q. Research on cloud databases. Ruanjian Xuebao/Journal of Software, 2012,23(5):1148−1166 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4195.htm [doi: 10.3724/SP.J.1001.2012.04195]

[104] Wang YJ, Sun WD, Zhou S, Pei XQ, Li XY. Key technologies of distributed storage for cloud computing. Ruanjian Xuebao/Journal of Software, 2012,23(4):962−986 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4175.htm [doi: 10.3724/SP.J.1001.2012.04175]

[105] Zheng P, Cui LZ, Wang HY, Xu M. A data placement strategy for data intensive applications in cloud. Chinese Journal of Computers, 2010,33(8):1472−1480 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2010.01472]

[106] CompuGreen. Green 500 list. 2012. http://www.green500.org/

[107] Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Franklin MJ, Shenker S, Stoica I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Gribble S, Katabi D, eds. Proc. of the NSDI 2012. San Jose: USENIX, 2012. 2−15.

[108] Zhou GL, Chen H. Parallel cube computation on modern CPUs and GPUs. The Journal of Supercomputing, 2012,61(3):394−417. [doi: 10.1007/s11227-011-0575-7]

[109] He GM, Feng HJ, Li CP, Chen H. Parallel SimRank computation on large graphs with iterative aggregation. In: Rao B, Krishnapuram B, Tomkins A, Yang Q, eds. Proc. of the SIGKDD 2010. Washington: ACM Press, 2010. 543−552. [doi: 10.1145/1835804.1835874]

[110] Li CP, Han JW, He GM, Jin X, Sun YZ, Yu YT, Wu TY. Fast computation of SimRank for static and dynamic information networks. In: Manolescu I, Spaccapietra S, Teubner J, Kitsuregawa M, Léger A, Naumann F, Ailamaki A, Özcan F, eds. Proc. of the EDBT 2010. Lausanne: ACM, 2010. 465−476. [doi: 10.1145/1739041.1739098]

[111] Wang H, Qin X, Zhang Y, Wang S, Wang Z. LinearDB: A relational approach to make data warehouse scale like MapReduce. In: Yu JX, Kim MH, Unland R, eds. Proc. of the DASFAA 2011. Hong Kong: Springer-Verlag, 2011. 306−320. [doi: 10.1007/978-3-642-20152-3_23]

[112] Bloor R. The coming database revolution. 2012. http://www.slideshare.net/Dataversity/thu-0830-bloorrobincolor

附中文参考文献:

[27] 倪明选,罗吴蔓.数据爆炸时代的技术变革.计算机学会通讯,2011,7(7):12−20.

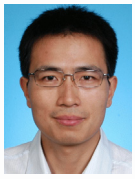
[83] 覃雄派,王会举,杜小勇,王珊.大数据分析——RDBMS 与 MapReduce 的竞争与共生.软件学报,2012,23(1):32−45. http://www.jos.org.cn/1000-9825/4091.htm [doi: 10.3724/SP.J.1001.2012.04091]

[103] 林子雨,赖永炫,林琛,谢怡,邹权.云数据库研究.软件学报,2012,23(5):1148−1166. http://www.jos.org.cn/1000-9825/4195.htm [doi: 10.3724/SP.J.1001.2012.04195]

[104] 王意洁,孙伟东,周松,裴晓强,李小勇.云计算环境下的分布存储关键技术.软件学报,2012,23(4):962−986. http://www.jos.org.cn/1000-9825/4175.htm [doi: 10.3724/SP.J.1001.2012.04175]

[105] 郑湃,崔立真,王海洋,徐猛.云计算环境下面向数据密集型应用的数据布局策略与方法.计算机学报,2010,33(8):1472−1480. [doi: 10.3724/SP.J.1016.2010.01472]

覃雄派(1971－),男,广西百色人,博士,讲师,主要研究领域为高性能数据库,大规模数据分析.
E-mail: qxp1990@sina.com



王会举(1979－),男,博士,主要研究领域为云计算,高性能数据库.
E-mail: wanghj@comp.nus.edu.sg

李芙蓉(1989－),女,博士生,主要研究领域为高性能数据库,云计算.
E-mail: furongli.cn@gmail.com

李翠平(1971－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据仓库,数据挖掘,信息网络,数据流.
E-mail: licuiping@ruc.edu.cn

陈红(1965－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据仓库,数据挖掘,流数据分析和管理,传感器网络数据管理.
E-mail: chong@ruc.edu.cn

周烜(1979－),男,博士,副教授,CCF 会员,主要研究领域为高性能数据库,内存数据库.
E-mail: xuan.zhou.mail@gmail.com

杜小勇(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为智能信息检索,高性能数据库,知识工程.
E-mail: duyong@ruc.edu.cn

王珊(1944－),女,教授,博士生导师,CCF 高级会员,主要研究领域为高性能数据库,知识工程,数据仓库,数据分析.
E-mail: swang@ruc.edu.cn