计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
50(1): 146-169, 2013

大数据管理:概念、技术与挑战

孟小峰　　慈　祥
(中国人民大学信息学院　北京　100872)
(xfmeng@ruc.edu.cn)

Big Data Management: Concepts, Techniques and Challenges

Meng Xiaofeng and Ci Xiang
(*School of Information, Renmin University of China, Beijing* 100872)

Abstract Data type and amount in human society is growing in amazing speed which is caused by emerging new services such as cloud computing, internet of things and social network, the era of big data has come. Data has been fundamental resource from simple dealing object, and how to manage and utilize big data better has attracted much attention. Evolution or revolution on database research for big data is a problem. This paper discusses the concept of big data, and surveys its state of the art. The framework of big data is described and key techniques are studied. Finally some new challenges in the future are summarized.

Key words big data; data analysis; cloud computing

摘　要　云计算、物联网、社交网络等新兴服务促使人类社会的数据种类和规模正以前所未有的速度增长,大数据时代正式到来.数据从简单的处理对象开始转变为一种基础性资源,如何更好地管理和利用大数据已经成为普遍关注的话题.大数据的规模效应给数据存储、管理以及数据分析带来了极大的挑战,数据管理方式上的变革正在酝酿和发生.对大数据的基本概念进行剖析,并对大数据的主要应用作简单对比.在此基础上,阐述大数据处理的基本框架,并就云计算技术对于大数据时代数据管理所产生的作用进行分析.最后归纳总结大数据时代所面临的新挑战.

关键词　大数据;数据分析;云计算

中图法分类号　TP311

随着以博客、社交网络、基于位置的服务 LBS 为代表的新型信息发布方式的不断涌现,以及云计算、物联网等技术的兴起,数据正以前所未有的速度在不断地增长和累积,大数据时代已经来到.学术界、工业界甚至于政府机构都已经开始密切关注大数据问题,并对其产生浓厚的兴趣.就学术界而言,《Nature》早在 2008 年就推出了 Big Data 专刊[1].计算社区联盟(Computing Community Consortium)在 2008 年发表了报告"Big data computing: Creating revolutionary breakthroughs in commerce, science, and society"[2],阐述了在数据驱动的研究背景下,解决大数据问题所需的技术以及面临的一些挑战.《Science》在 2011 年 2 月推出专刊"Dealing with Data"[3],主要围绕着科学研究中大数据的问题展开讨论,说明大数据对于科学研究的重要性.美国一些知名的数据管理领域的专家学者则从专业的研究角度出发,联合发布了一份白皮书《Challenges and Opportunities with Big Data》[4].该白皮书从学术的角度出发介绍了大数据的产生,分析了大数据的处理流程,并提出大数据所面临的若干挑战.

收稿日期:2012-11-30;修回日期:2012-12-04
基金项目:国家自然科学基金项目(61070055,91024032,91124001,60833005);中国人民大学科学研究基金项目(11XNL010);国家"八六三"高技术研究发展计划基金项目(2012AA010701)

全球知名的咨询公司麦肯锡(McKinsey)2011年6月份发布了一份关于大数据的详尽报告"Big data: The next frontier for innovation, competition, and productivity"[5],对大数据的影响、关键技术和应用领域等都进行了详尽的分析.进入2012年以来,大数据的关注度与日俱增.1月份的达沃斯世界经济论坛上,大数据是主题之一,该次会议还特别针对大数据发布了报告"Big data, big impact: New possibilities for international development"[6],探讨了新的数据产生方式下,如何更好地利用数据来产生良好的社会效益.该报告重点关注了个人产生的移动数据与其他数据的融合与利用.3月份美国奥巴马政府发布了"大数据研究和发展倡议"[7](Big data research and development initiative),投资2亿以上美元,正式启动"大数据发展计划".计划在科学研究、环境、生物医学等领域利用大数据技术进行突破.奥巴马政府的这一计划被视为美国政府继信息高速公路(Information Highway)计划之后在信息科学领域的又一重大举措.与此同时,联合国一个名为"Global Pulse"的倡议项目在今年5月发布报告"Big data for development: Challenges & opportunities"[8],该报告主要阐述大数据时代各国特别是发展中国家在面临数据洪流(data deluge)的情况下所遇到的机遇与挑战,同时还对大数据的应用进行了初步的解读.《纽约时报》的文章"The age of big data"[9]则通过主流媒体的宣传使普通民众开始意识到大数据的存在,以及大数据对于人们日常生活的影响.

大数据的火热并不意味着对于大数据的了解深入,反而表明大数据存在过度炒作的危险.大数据的基本概念、关键技术以及对其利用上均存在很多的疑问和争议.本文从大数据问题背后的本质出发,对现有的大数据研究资料进行全面的归纳和总结.首先简要介绍大数据的基本概念,阐述其同传统数据库的区别.在此基础上,对大数据处理框架进行详细解析.我们认为大数据的发展离不开云计算技术,云计算支撑着大数据存储、管理以及数据分析等.因此本文展开介绍了大数据时代不可或缺的云计算技术和工具.最后全面阐述大数据时代面临的新挑战.

1 大数据的基本概念、来源与应用

1.1 大数据的基本概念

大数据本身是一个比较抽象的概念,单从字面来看,它表示数据规模的庞大.但是仅仅数量上的庞大显然无法看出大数据这一概念和以往的"海量数据"(massive data)、"超大规模数据"(very large data)等概念之间有何区别.对于大数据尚未有一个公认的定义,不同的定义基本是从大数据的特征出发,通过这些特征的阐述和归纳试图给出其定义.在这些定义中,比较有代表性的是3V定义[10],即认为大数据需满足3个特点:规模性(volume)、多样性(variety)和高速性(velocity).除此之外,还有提出4V定义的,即尝试在3V的基础上增加一个新的特性.关于第4个V的说法并不统一,国际数据公司(International Data Corporation, IDC)认为大数据还应当具有价值性(value)[11],大数据的价值往往呈现出稀疏性的特点.而IBM认为大数据必然具有真实性(veracity)[12].维基百科对大数据的定义[13]则简单明了:大数据是指利用常用软件工具捕获、管理和处理数据所耗时间超过可容忍时间的数据集.

眼下在大数据定义问题上很难达成一个完全的共识,这点和云计算的概念刚提出时的情况是相似的.在面对实际问题时,不必过度地拘泥于具体的定义之中,在把握3V定义的基础上适当地考虑4V特性即可.

1.2 从数据库(database, DB)到大数据(big data, BD)

从数据库到大数据,看似只是一个简单的技术演进,但细细考究不难发现两者有着本质上的差别.大数据的出现必将颠覆传统的数据管理方式.在数据来源、数据处理方式和数据思维等方面都会对其带来革命性的变化.

如果要用简单的方式来比较传统的数据库和大数据的区别,我们认为"池塘捕鱼"和"大海捕鱼"是个很好的类比."池塘捕鱼"代表着传统数据库时代的数据管理方式,而"大海捕鱼"则对应着大数据时代的数据管理方式,"鱼"是待处理的数据."捕鱼"环境条件的变化导致了"捕鱼"方式的根本性差异.这些差异主要体现在如下几个方面:

1) 数据规模."池塘"和"大海"最容易发现的区别就是规模."池塘"规模相对较小,即便是先前认为比较大的"池塘",譬如VLDB(very large database),和"大海"XLDB(extremely large database)相比仍旧偏小."池塘"的处理对象通常以MB为基本单位,而"大海"则常常以GB,甚至是TB,PB为基本处理单位.

2) 数据类型.过去的"池塘"中,数据的种类单一,往往仅仅有一种或少数几种,这些数据又以结构

化数据为主. 而在“大海”中数据的种类繁多,数以千计,而这些数据又包含着结构化、半结构化以及非结构化的数据,并且半结构化和非结构化数据所占份额越来越大.

3) 模式(schema)和数据的关系. 传统的数据库都是先有模式,然后才会产生数据. 这就好比是先选好合适的“池塘”,然后才会向其中投放适合在该“池塘”环境生长的“鱼”. 而大数据时代很多情况下难以预先确定模式,模式只有在数据出现之后才能确定,且模式随着数据量的增长处于不断的演变之中. 这就好比先有少量的鱼类,随着时间推移,鱼的种类和数量都在不断地增长. 鱼的变化会使大海的成分和环境处于不断的变化之中.

4) 处理对象. 在“池塘”中捕鱼,“鱼”仅仅是其捕捞对象. 而在“大海”中,“鱼”除了是捕捞对象之外,还可以通过某些“鱼”的存在来判断其他种类的“鱼”是否存在. 也就是说传统数据库中数据仅作为处理对象. 而在大数据时代,要将数据作为一种资源来辅助解决其他诸多领域的问题.

5) 处理工具. 捕捞“池塘”中的“鱼”,一种渔网或少数几种基本就可以应对,也就是所谓的 One size fits all. 但是在“大海”中,不可能存在一种渔网能够捕获所有的鱼类,也就是说 No size fits all.

从“池塘”到“大海”不仅仅是规模的变大. 传统的数据库代表着数据工程(data engineering)的处理方式,大数据时代的数据已不仅仅只是工程处理的对象,需要采取新的数据思维来应对. 图灵奖获得者、著名数据库专家 Jim Gray 博士观察并总结人类自古以来,在科学研究上,先后历经了实验、理论和计算 3 种范式. 当数据量不断增长和累积到今天,传统的 3 种范式在科学研究,特别是一些新的研究领域已经无法很好地发挥作用,需要有一种全新的第 4 种范式来指导新形势下的科学研究. 基于这种考虑,Jim Gray 提出了一种新的数据探索型研究方式,被他自己称之为科学研究的“第 4 种范式”(The Fourth Paradigm)[14].

4 种范式的比较如表 1[14] 所示. 第 4 种范式的实质就是从以计算为中心转变到以数据处理为中心,也就是我们所说的数据思维. 这种方式需要我们从根本上转变思维. 正如前面提到的“捕鱼”,在大数据时代,数据不再仅仅是“捕捞”的对象,而应当转变成一种基础资源,用数据这种资源来协同解决其他诸多领域的问题. 计算社会科学(computational social science)[15] 基于特定社会需求,在特定的社会理论指导下,收集、整理和分析数据足迹(data print),以便进行社会解释、监控、预测与规划的过程和活动. 计算社会科学是一种典型的需要采用第 4 种范式来作指导的科学研究领域. Watts 在《Nature》杂志上的文章“A twenty-first century science”[16] 也指出,借助于社交网络和计算机分析技术,21 世纪的社会科学有可能实现定量化的研究,从而成为一门真正的自然科学.

Table 1 Four Science Paradigms

表 1 科学发现的 4 种范式

Science Paradigms	Time	Methodology
Empirical	Thousand years ago	Describing natural phenomena
Theoretical	Last few hundred years	Using models, generalizations
Computational	Last few decades	Simulating complex phenomena
Data Exploration (eScience)	Today	Data captured by instruments or generated by simulator; Processed by software; Information stored in computer; Scientist analyzes database

1.3 大数据的产生和应用

人类历史上从未有哪个时代和今天一样产生如此海量的数据. 数据的产生已经完全不受时间、地点的限制. 从开始采用数据库作为数据管理的主要方式开始,人类社会的数据产生方式大致经历了 3 个阶段,而正是数据产生方式的巨大变化才最终导致大数据的产生.

1) 运营式系统阶段. 数据库的出现使得数据管理的复杂度大大降低,实际中数据库大都为运营系统所采用,作为运营系统的数据管理子系统,比如超市的销售记录系统、银行的交易记录系统、医院病人的医疗记录等. 人类社会数据量第 1 次大的飞跃正是建立在运营式系统开始广泛使用数据库. 这个阶段最主要特点是数据往往伴随着一定的运营活动而产生并记录在数据库中,比如超市每销售出一件产品就会在数据库中产生相应的一条销售记录. 这种

数据的产生方式是被动的.

2) 用户原创内容阶段. 互联网的诞生促使人类社会数据量出现第 2 次大的飞跃. 但是真正的数据爆发产生于 Web 2.0 时代,而 Web 2.0 的最重要标志就是用户原创内容(user generated content, UGC). 这类数据近几年一直呈现爆炸性的增长,主要有两方面的原因:首先是以博客、微博为代表的新型社交网络的出现和快速发展,使得用户产生数据的意愿更加强烈;其次就是以智能手机、平板电脑为代表的新型移动设备的出现,这些易携带、全天候接入网络的移动设备使得人们在网上发表自己意见的途径更为便捷,这个阶段数据的产生方式是主动的.

3) 感知式系统阶段. 人类社会数据量第 3 次大的飞跃最终导致了大数据的产生,今天我们正处于这个阶段. 这次飞跃的根本原因在于感知式系统的广泛使用. 随着技术的发展,人们已经有能力制造极其微小的带有处理功能的传感器,并开始将这些设备广泛地布置于社会的各个角落,通过这些设备来对整个社会的运转进行监控. 这些设备会源源不断地产生新数据,这种数据的产生方式是自动的.

简单来说,数据产生经历了被动、主动和自动 3 个阶段. 这些被动、主动和自动的数据共同构成了大数据的数据来源,但其中自动式的数据才是大数据产生的最根本原因.

正如 Google 的首席经济学家 Hal Varian 所说[17],数据是广泛可用的,所缺乏的是从中提取出知识的能力. 数据收集的根本目的是根据需求从数据中提取有用的知识,并将其应用到具体的领域之中. 不同领域的大数据应用有不同的特点,表 2 列举了若干具有代表性的大数据应用及其特征:

Table 2 Comparison between Typical Big Data Applications

表 2 典型大数据应用的比较

Applications	Examples	Number of Users	Response Time	Data Scale	Reliability	Accuracy
Scientific Computing	Bioinformatics	Small	Slow	TB	Moderate	Very High
Finance	High-frequency trading	Large	Very Fast	GB	Very High	Very High
Social network	Facebook	Very Large	Fast	PB	High	High
Mobile Data	Mobile phone	Very Large	Fast	TB	High	High
Internet of Things	Sensor network	Large	Fast	TB	High	High
Web Data	News website	Very Large	Fast	PB	High	High
Multimedia	Video site	Very Large	Fast	PB	High	Moderate

正是由于大数据的广泛存在才使得大数据问题的解决很具挑战性. 而它的广泛应用则促使越来越多的人开始关注和研究大数据问题.

2 大数据处理框架

2.1 大数据处理模式

大数据的应用类型有很多,主要的处理模式可以分为流处理(stream processing)和批处理(batch processing)两种[18-19]. 批处理是先存储后处理(store-then-process),而流处理则是直接处理(straight-through processing).

2.1.1 流处理

流处理的基本理念是数据的价值会随着时间的流逝而不断减少,因此尽可能快地对最新的数据作出分析并给出结果是所有流数据处理模式的共同目标. 需要采用流数据处理的大数据应用场景主要有网页点击数的实时统计、传感器网络、金融中的高频交易等.

流处理的处理模式将数据视为流,源源不断的数据组成了数据流. 当新的数据到来时就立刻处理并返回所需的结果. 图 1[18] 是流处理中基本的数据流模型:

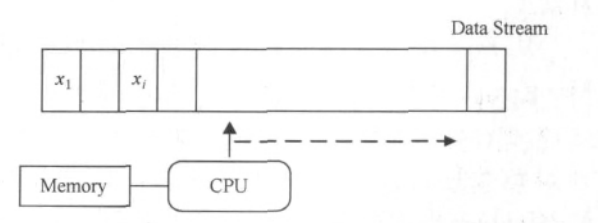

Fig. 1 Basic data stream model.

图 1 基本的数据流模型

数据的实时处理是一个很有挑战性的工作,数据流本身具有持续达到、速度快且规模巨大等特点,因此通常不会对所有的数据进行永久化存储,而且

数据环境处在不断的变化之中,系统很难准确掌握整个数据的全貌.

由于响应时间的要求,流处理的过程基本在内存中完成,其处理方式更多地依赖于在内存中设计巧妙的概要数据结构(synopsis data structure),内存容量是限制流处理模型的一个主要瓶颈.以 PCM(相变存储器)为代表的储存级内存(storage class memory, SCM)设备的出现或许可以使内存未来不再成为流处理模型的制约.

数据流的理论及技术研究已经有十几年的历史,目前仍旧是研究热点.与此同时很多实际系统也已开发和得到广泛的应用,比较代表性的开源系统如 Twitter 的 Storm[20]、Yahoo 的 S4[21]以及 Linkedin 的 Kafka[22]等.

2.1.2 批处理

Google 公司在 2004 年提出的 MapReduce[23]编程模型是最具代表性的批处理模式.一个完整的 MapReduce 过程如图 2[23]所示:

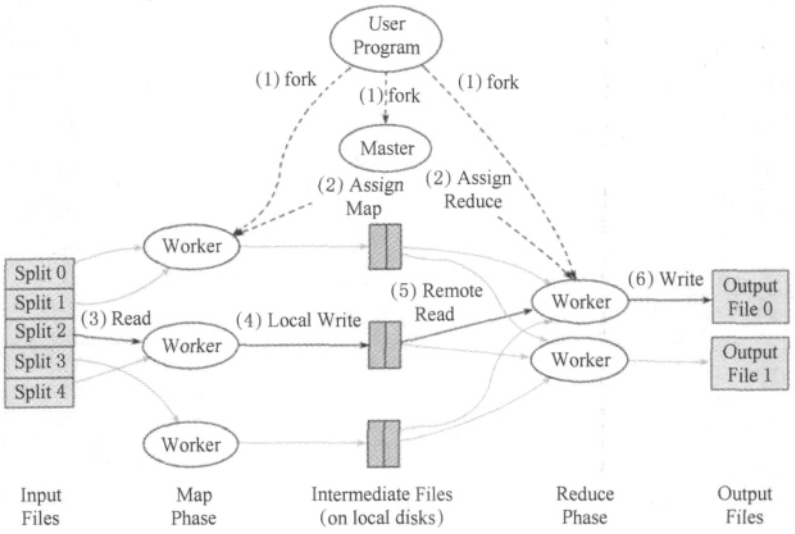

Fig. 2 Execution overview of MapReduce.

图 2 MapReduce 执行流程图

MapReduce 模型首先将用户的原始数据源进行分块,然后分别交给不同的 Map 任务区处理.Map 任务从输入中解析出链/值(Key/Value)对集合,然后对这些集合执行用户自行定义的 Map 函数得到中间结果,并将该结果写入本地硬盘.Reduce 任务从硬盘上读取数据之后会根据 key 值进行排序,将具有相同 Key 值的组织在一起.最后用户自定义的 Reduce 函数会作用于这些排好序的结果并输出最终结果.

从 MapReduce 的处理过程我们可以看出,MapReduce 的核心设计思想在于:1)将问题分而治之;2)把计算推到数据而不是把数据推到计算,有效地避免数据传输过程中产生的大量通信开销.MapReduce 模型简单,且现实中很多问题都可用 MapReduce 模型来表示.因此该模型公开后立刻受到极大的关注,并在生物信息学、文本挖掘等领域得到广泛的应用.

无论是流处理还是批处理都是大数据处理的可行思路.大数据的应用类型很多,在实际的大数据处理中,常常并不是简单地只使用其中的某一种,而是将二者结合起来.互联网是大数据最重要的来源之一,很多互联网公司根据处理时间的要求将自己的业务划分为在线(online)、近线(nearline)和离线(offline),比如著名的职业社交网站 Linkedin[24],这种划分方式是按处理所耗时间来划分的.其中在线的处理时间一般在秒级甚至是毫秒级,因此通常采用上面所说的流处理.离线的处理时间可以以天为基本单位,基本采用批处理方式,这种方式可以最大限度地利用系统 I/O.近线的处理时间一般在分钟级或者是小时级,对其处理模型并没有特别的要求,可以根据需求灵活选择,但在实际中多采用批处理模式.

2.2 大数据处理的基本流程

大数据的数据来源广泛,应用需求和数据类型都不尽相同,但是最基本的处理流程一致.海量 Web 数据的处理是一类非常典型的大数据应用,从中可以归纳出大数据处理的最基本流程.ScholarSpace[25]由中国人民大学网络与移动数据管理实验室(WAMDM)开发,目标是建立一个“以人为本”,即

以作者为中心来展示多学科中文文献的集成数据库系统. 该系统已经从最初的计算机领域扩展到包括经济、法律等人文社会科学在内的多领域,从数据抽取和集成,一直到最终的结果展示,ScholarSpace 完整地体现出大数据处理的一般流程. 在其基础上我们归纳出大数据的基本流程,如图 3 所示:

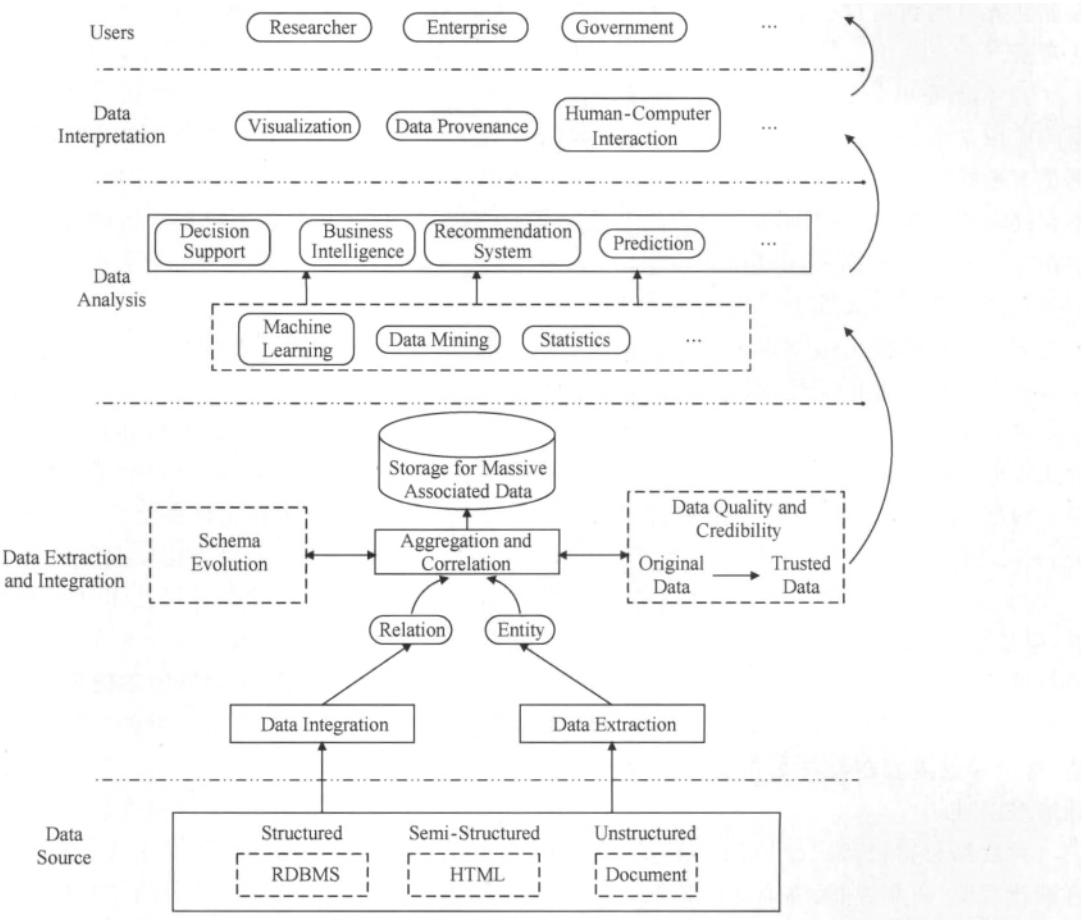

Fig. 3 Basic framework of big data processing.

图 3 大数据处理基本流程

整个大数据的处理流程可以定义为在合适工具的辅助下,对广泛异构的数据源进行抽取和集成,结果按照一定的标准统一存储. 利用合适的数据分析技术对存储的数据进行分析,从中提取有益的知识并利用恰当的方式将结果展现给终端用户. 具体来说可以分为数据抽取与集成、数据分析以及数据解释.

2.2.1 数据抽取与集成

大数据的一个重要特点就是多样性,这就意味着数据来源极其广泛,数据类型极为繁杂,这种复杂的数据环境给大数据的处理带来极大的挑战. 要想处理大数据,首先必须对所需数据源的数据进行抽取和集成,从中提取出关系和实体,经过关联和聚合之后采用统一定义的结构来存储这些数据. 在数据集成和提取时需要对数据进行清洗,保证数据质量及可信性. 同时还要特别注意前面提及的大数据时代模式和数据的关系,大数据时代的数据往往是先有数据再有模式,且模式是在不断的动态演化之中的.

数据抽取和集成技术不是一项全新的技术,传统数据库领域已对此问题有了比较成熟的研究. 随着新的数据源的涌现,数据集成方法也在不断的发展之中. 从数据集成模型来看,现有的数据抽取与集成方式可以大致分为以下 4 种类型[26]:基于物化或 ETL 方法的引擎(materialization or ETL engine)、基于联邦数据库或中间件方法的引擎(federation engine or mediator)、基于数据流方法的引擎(stream engine)及基于搜索引擎的方法(search engine).

2.2.2 数据分析

数据分析是整个大数据处理流程的核心,因为大数据的价值产生于分析过程. 从异构数据源抽取和集成的数据构成了数据分析的原始数据. 根据不同应用的需求可以从这些数据中选择全部或部分进行分析. 传统的分析技术如数据挖掘、机器学习、统计分析等在大数据时代需要作出调整,因为这些技术在大数据时代面临着一些新的挑战,主要有:

1) 数据量大并不一定意味着数据价值的增加，相反这往往意味着数据噪音的增多. 因此在数据分析之前必须进行数据清洗等预处理工作，但是预处理如此大量的数据对于机器硬件以及算法都是严峻的考验.

2) 大数据时代的算法需要进行调整. 首先大数据的应用常常具有实时性的特点，算法的准确率不再是大数据应用的最主要指标. 很多场景中算法需要在处理的实时性和准确率之间取得一个平衡，比如在线的机器学习算法(online machine learning)；其次云计算是进行大数据处理的有力工具，这就要求很多算法必须作出调整以适应云计算的框架，算法需要变得具有可扩展性；最后在选择算法处理大数据时必须谨慎，当数据量增长到一定规模以后，可以从小量数据中挖掘出有效信息的算法并一定适用于大数据. 统计学中的邦弗朗尼原理(Bonferroni's principle)[27]① 就是一个典型的例子.

3) 数据结果好坏的衡量. 得到分析结果并不难，但是结果好坏的衡量却是大数据时代数据分析的新挑战. 大数据时代的数据量大、类型庞杂，进行分析时往往对整个数据的分布特点掌握的不太清楚，这会导致最后在设计衡量的方法以及指标时遇到诸多困难.

大数据分析已被广泛应用于诸多领域，典型的有推荐系统、商业智能、决策支持等.

2.2.3 数据解释

数据分析是大数据处理的核心，但是用户往往更关心结果的展示. 如果分析的结果正确但是没有采用适当的解释方法，则所得到的结果很可能让用户难以理解，极端情况下甚至会误导用户. 数据解释的方法很多，比较传统的就是以文本形式输出结果或者直接在电脑终端上显示结果. 这种方法在面对小数据量时是一种很好的选择. 但是大数据时代的数据分析结果往往也是海量的，同时结果之间的关联关系极其复杂，采用传统的解释方法基本不可行. 可以考虑从下面两个方面提升数据解释能力.

1) 引入可视化技术. 可视化作为解释大量数据最有效的手段之一率先被科学与工程计算领域采用. 通过对分析结果的可视化用形象的方式向用户展示结果，而且图形化的方式比文字更易理解和接受. 常见的可视化技术有标签云(tag cloud)、历史流(history flow)、空间信息流(spatial information flow)等. 可以根据具体的应用需要选择合适的可视化技术.

2) 让用户能够在一定程度上了解和参与具体的分析过程. 这个既可以采用人机交互技术，利用交互式的数据分析过程来引导用户逐步地进行分析，使得用户在得到结果的同时更好地理解分析结果的由来. 也可以采用数据起源技术[28]，通过该技术可以帮助追溯整个数据分析的过程，有助于用户理解结果.

3 关键技术分析

大数据价值的完整体现需要多种技术的协同. 文件系统提供最底层存储能力的支持. 为了便于数据管理，需要在文件系统之上建立数据库系统. 通过索引等的构建，对外提供高效的数据查询等常用功能. 最终通过数据分析技术从数据库中的大数据提取出有益的知识.

3.1 云计算：大数据的基础平台与支撑技术

如果将各种大数据的应用比作一辆辆“汽车”，支撑起这些“汽车”运行的“高速公路”就是云计算. 正是云计算技术在数据存储、管理与分析等方面的支撑，才使得大数据有用武之地.

在所有的“高速公路”中，Google 无疑是技术最为先进的一个. 需求推动创新，面对海量的 Web 数据，Google 于 2006 年首先提出了云计算的概念. 支撑 Google 内部各种大数据应用的正是其自行研发的一系列云计算技术和工具. 难能可贵的是 Google 并未将这些技术完全封闭，而是以论文的形式逐步公开其实现. 正是这些公开的论文，使得以 GFS，MapReduce，Bigtable 为代表的一系列大数据处理技术被广泛了解并得到应用，同时还催生出以 Hadoop[29] 为代表的一系列云计算开源工具. 云计算所涉及到的技术很多，但是通过 Google 云计算技术的介绍能够快速、完整地把握云计算技术的核心和精髓. 本节以 Google 的相关技术介绍为主线，详细介绍 Google 以及其他众多学者和研究机构在大数据技术方面已有的一些工作. 根据 Google 已公开的论文及相关资料，结合大数据处理的需求，我们对 Google 的技术演化进行了整理，如图 4② 所示：

① 邦弗朗尼原理表明并非给定数据集和挖掘任务就肯定能挖掘出合理的结果。具体内容见文献[27].

② 上面所列的系统绝大部分都已经有论文公布其大致实现，虽然 Colossus 和 Caffeine 系统并没有论文公开，但是可以确定其存在. 图 4 中所列时间如无特别标明，均为论文发表时间，并不代表其在 Google 内部的正式部署和使用时间.

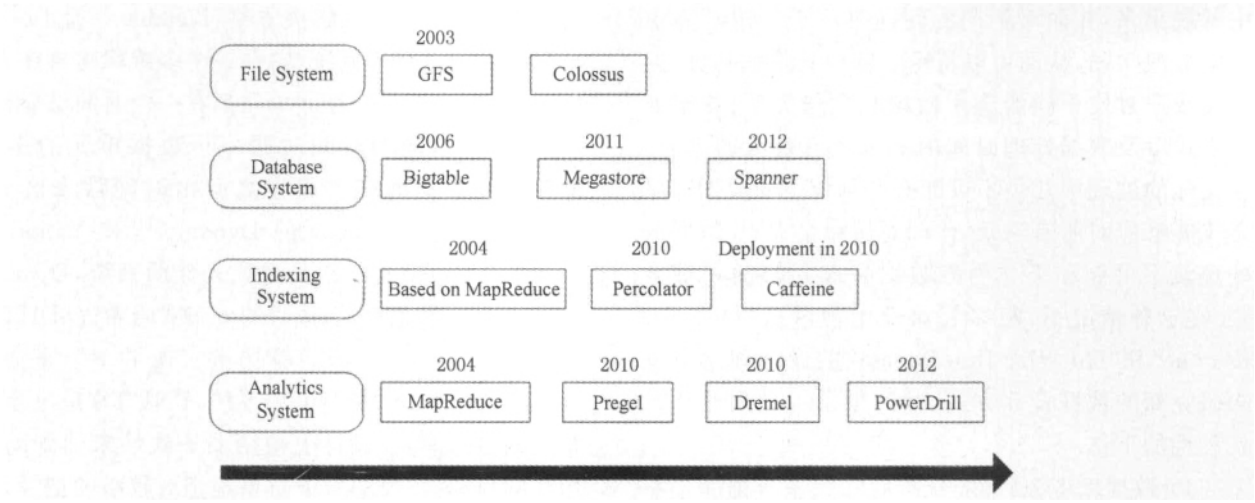

Fig. 4 Technology evolution in Google.

图 4 Google 技术演化图

3.1.1 文件系统

文件系统是支撑上层应用的基础. 在 Google 之前,尚未有哪个公司面对过如此多的海量数据. 因此对于 Google 而言并没有完全成熟的存储方案可以直接使用. Google 认为系统组件失败是一种常态而不是异常,基于此思想 Google 自行设计开发了 Google 文件系统 GFS[30](Google file system). GFS 是构建在大量廉价服务器之上的一个可扩展的分布式文件系统,GFS 主要针对文件较大,且读远大于写的应用场景,采用主从(Master-Slave)结构. 通过数据分块、追加更新(append-only)等方式实现了海量数据的高效存储. 随着时间推移,GFS 的架构逐渐开始无法适应需求. Google 对 GFS 进行了重新的设计,该系统正式的名称为 Colosuss,具体实现尚未公开,但是从 ACM 对 GFS 团队核心工程师的访谈[31]可以了解其一些新的特性. 其中 GFS 的单点故障(指仅有一个主节点容易成为系统的瓶颈)、海量小文件的存储等问题在 Colosuss 中均得到了解决.

除了 Google,众多企业和学者也从不同方面对满足大数据存储需求的文件系统进行了详尽的研究. 微软自行开发的 Cosmos[32]支撑着其搜索、广告等业务. HDFS[33]和 CloudStore[34]都是模仿 GFS 的开源实现. GFS 类的文件系统主要是针对较大文件设计的,而在图片存储等应用场景,文件系统主要存储海量小文件,此时 GFS 等文件系统因为频繁读取元数据等原因,效率很低. 针对这种情况,Facebook 推出了专门针对海量小文件的文件系统 Haystack[35],通过多个逻辑文件共享同一个物理文件、增加缓存层、部分元数据加载到内存等方式有效地解决了 Facebook 海量图片存储问题. 淘宝推出了类似的文件系统 TFS[36](Tao file system),通过将小文件合并成大文件、文件名隐含部分元数据等方式实现了海量小文件的高效存储. FastDFS[37]针对小文件的优化类似于 TFS.

3.1.2 数据库系统

原始的数据存储在文件系统之中,但是用户习惯通过数据库系统来存取文件. 因为这样会屏蔽掉底层的细节,且方便数据管理. 直接采用关系模型的分布式数据库并不能适应大数据时代的数据存储,主要因为:

1) 规模效应所带来的压力. 大数据时代的数据量远远超过单机所能容纳的数据量,因此必须采用分布式存储的方式. 这就需要系统具有很好的扩展性,但这恰恰是传统数据库的弱势之一. 因为传统的数据库产品对于性能的扩展更倾向于 scale-up(纵向扩展)的方式,而这种方式对于性能的增加速度远低于需要处理数据的增长速度,且性能提升存在上限. 适应大数据的数据库系统应当具有良好的 scale-out(横向扩展)能力,而这种性能扩展方式恰恰是传统数据库所不具备的. 即便是性能最好的并行数据库产品其 scale-out 能力也相对有限.

2) 数据类型的多样化. 传统的数据库比较适合结构化数据的存储,但是数据的多样性是大数据时代的显著特征之一. 这也就是意味着除了结构化数据,半结构化和非结构化数据也将是大数据时代的重要数据类型组成部分. 如何高效地处理多种数据类型是大数据时代数据库技术面临的重要挑战之一.

3) 设计理念的冲突. 关系数据库追求的是“One size fits all”的目标,希望将用户从繁杂的数据管理

中解脱出来,在面对不同的问题时不需要重新考虑数据管理问题,从而可以将重心转向其他的部分.但在大数据时代不同的应用领域在数据类型、数据处理方式以及数据处理时间的要求上有极大的差异.在实际的处理中几乎不可能有一种统一的数据存储方式能够应对所有场景.比如对于海量 Web 数据的处理就不可能和天文图像数据采取同样的处理方式.在这种情况下,很多公司开始尝试从"One size fits one"和"One size fits domain"的设计理念出发来研究新的数据管理方式,并产生了一系列非常有代表性的工作.

4) 数据库事务特性.众所周知关系数据库中事务的正确执行必须满足 ACID 特性,即原子性(atomicity)、一致性(consistency)、隔离性(isolation)和持久性(durability).对于数据强一致性的严格要求使其在很多大数据场景中无法应用.这种情况下出现了新的 BASE 特性,即只要求满足 basically available(基本可用), soft state(柔性状态)和 eventually consistent(最终一致).从分布式领域著名的 CAP 理论[38]① 的角度来看,ACID 追求一致性 C,而 BASE 更加关注可用性 A.正是在事务处理过程中对于 ACID 特性的严格要求,使得关系型数据库的可扩展性极其有限.

面对这些挑战,以 Google 为代表的一批技术公司纷纷推出了自己的解决方案. Bigtable[39] 是 Google 早期开发的数据库系统,它是一个多维稀疏排序表,由行和列组成,每个存储单元都有一个时间戳,形成三维结构.不同的时间对同一个数据单元的多个操作形成数据的多个版本之间由时间戳来区分.除了 Bigtable, Amazon 的 Dynamo[40] 和 Yahoo 的 PNUTS[41] 也都是非常具有代表性的系统. Dynamo 综合使用了键/值存储、改进的分布式哈希表(DHT)、向量时钟(vector clock)等技术实现了一个完全的分布式、去中心化的高可用系统. PNUTS 是一个分布式的数据库,在设计上使用弱一致性来达到高可用性的目标,主要的服务对象是相对较小的记录,比如在线的大量单个记录或者小范围记录集合的读和写访问.不适合存储大文件、流媒体等. Bigtable, Dynamo, PNUTS 等的成功促使人们开始对关系数据库进行反思,由此产生了一批未采用关系模型的数据库,这些方案现在被统一称为 NoSQL(not only SQL). NoSQL 并没有一个准确的定义,但一般认为 NoSQL 数据库应当具有以下的特征[42]:模式自由(schema-free)、支持简易备份(easy replication support)、简单的应用程序接口(simple API)、最终一致性(或者说支持 BASE 特性,不支持 ACID)、支持海量数据(huge amount of data). NoSQL 和关系型数据库的简单比较如表 3 所示:

Table 3 Comparison between NoSQL Database and RDBMS

表 3 NoSQL 数据库和关系数据库的对比

Objects of Comparison	RDBMS	NoSQL	Notes
Rationale	Perfect	Imperfect	RDBMS is based on mathematical model; NoSQL has no such model.
Data Scale	Large	Extremely Large	Performance of RDBMS will degrade as the data increase, so it's usually not appropriate for extremely large data; NoSQL can increase the volume of storage by adding more devices.
Schema	Fixed	Flexible	RDBMS must define schema at first; NoSQL is schema-free.
Query	Fast	Simple query is efficient	RDBMS will build index, so it can well support point query and range query; NoSQL has no index, although the query processing can be accelerated by MapReduce, it is still less efficient.
Consistency	Strong consistency	Weak consistency	RDBMS obey ACID; NoSQL obey BASE.
Scalability	Moderate	Good	RDBMS is difficult to scale; NoSQL can easily scale out by adding new nodes.
Availability	Good	Very Good	Availability of RDBMS is relatively weak when the volume of data is very large because of its limitation of strong consistency; NoSQL can achieve better availability by relaxing ACID semantics.
Standard	Yes	No	RDBMS has standard(SQL); NoSQL has no such standard.
Technical Support	High	Low	Technical support for RDBMS is high; Technical support for NoSQL is low.
Maintenance	Complex	Complex	RDBMS should be maintained by DBA; NoSQL is not sophisticated now, so its maintenance is also difficult.

① CAP 理论指出:一个分布式系统不可能同时满足一致性、可用性(availability)和分区容错性(partition tolerance),最多只能同时满足其中两个.

典型的 NoSQL 数据库分类如表 4[43]所示:

Table 4 Typical NoSQL Databases

表 4 典型 NoSQL 数据库

Category	Matching Databases	Performance	Scalability	Flexibility	Complexity	Advantages	Disadvantages
Key-Value	Redis Riak	High	High	High	None	Query is efficient	Stored data lack structure
Column	HBase Cassandra	High	High	Moderate	Low	Query is efficient	Functionality is minimal
Document	CouchDB MongoDB	High	Variable	High	Low	Little limits on data structure	Performance of query is low
Graph	Neo4J OrientDB	Variable	Variable	High	High	Graph algorithms are sophisticated	Data scale is relatively low

Bigtable 的模型简单,但是相较传统的关系数据库其支持的功能非常有限,不支持 ACID 特性.因此 Google 开发了 Megastore[44]系统,虽然其底层数据存储依赖 Bigtable,但是它实现了类似 RDBMS 的数据模型,同时提供数据的强一致性解决方案.Megastore 将数据进行细粒度的分区,数据更新会在机房间进行同步复制.Spanner[45]是已知的 Google 的最新的数据库系统,Google 在 OSDI2012 上公开了 Spanner 的实现.Spanner 是第 1 个可以实现全球规模扩展(global scale)并且支持外部一致的事务(support externally-consistent distributed transactions)的数据库.通过 GPS 和原子时钟(atomic clocks)技术,Spanner 实现了一个时间 API.借助该 API,数据中心之间的时间同步能够精确到 10 ms 以内.Spanner 类似于 Bigtable,但是它具有层次性的目录结构以及细粒度的数据复制.对于数据中心之间不同操作会分别支持强一致性或弱一致性,且支持更多的自动操作.Spanner 的目标是控制 100 万到 1 000 万台服务器,最多包含大约 10 万亿目录和 1 000 万亿字节的存储空间.另外在 SIGMOD2012 上,Google 公开了用于其广告系统的新数据库产品 F1[46],作为一种混合型数据库 F1 融合兼有 Bigtable 的高扩展性以及 SQL 数据库的可用性和功能性.该产品的底层存储正是采用 Spanner,具有很多新的特性,包括全局分布式、同步跨数据中心复制、可视分片和数据移动、常规事务等.

有些比较激进的观点认为"关系数据库已死",我们认为关系数据库和 NoSQL 并不是矛盾的对立体,而是可以相互补充的、适用于不同应用场景的技术.例如实际的互联网系统往往都是 ACID 和 BASE 两种系统的结合.近些年来,以 Spanner 为代表的若干新型数据库的出现,给数据存储带来了 SQL,NoSQL 之外的新思路.这种融合了一致性和可用性的 NewSQL 或许会是未来大数据存储新的发展方向.

3.1.3 索引与查询技术

数据查询是数据库最重要的应用之一,而索引则是解决数据查询问题的有效方案.就 Google 自身而言,索引的构建是提供搜索服务的关键部分.Google 最早的索引系统是利用 MapReduce 来更新的.根据更新频率进行层次划分,不同的层次对应不同的更新频率.每次需要批量更新索引,即使有些数据并未改变也需要处理掉.这种索引更新方式效率较低.随后 Google 提出了 Percolator[47],这是一种增量式的索引更新器,每次更新不需要替换所有的索引数据,效率大大提高.虽然不是所有的大数据应用都需要索引,但是这种增量计算的思想非常值得我们借鉴.Google 当前正在使用的索引系统为 Caffeine[48],其具体实现尚未公布.但是可以确定 Caffeine 是构建在 Spanner 之上,采用 Percolator 更新索引.效率相较上一代索引系统而言有大幅度提高.

关系数据库也是利用对数据构建索引的方式较好地解决了数据查询的问题.不同的索引方案使得关系数据库可以满足不同场景的要求.索引的建立以及更新都会耗费较多的时间,在面对传统数据库的小数据量时这些时间和其所带来的查询便利性相比是可以接受的,但是这些复杂的索引方案基本无法直接应用到大数据之上.表 5[49]是对一些索引方案直接应用在 Facebook 上的性能估计.

从表 5 可以看出不太可能将已有的成熟索引方案直接应用于大数据.NoSQL 数据库针对主键的查询效率一般较高,因此有关的研究集中在 NoSQL 数据库的多值查询优化上.针对 NoSQL 数据库上的查询优化研究主要有两种思路:

Table 5 Query Index Examples

表 5 查询索引的案例

Algorithms	Index Size for Facebook	Index Time for Facebook	Query Time on Facebook/s
Ullmann	—	—	>1 000
VF2	—	—	>1 000
RDF-3X	1 TB	>20 d	>48
BitMat	2.4 TB	>20d	>269
Subdue	—	>67 years	—
SpiderMine	—	>3 years	—
R-Join	>175 TB	$>10^{15}$ years	>200
Distance-Join	>175 TB	$>10^{15}$ years	>4 000
GraphQL	>13 TB(r=2)	>600 years	>2 000
Zhao	>12 TB(r=2)	>600 years	>600
GADDI	$>2\times10^5$ TB(L=4)	$>4\times10^5$ years	>400

1) 采用 MapReduce 并行技术优化多值查询：当利用 MapReduce 并行查询 NoSQL 数据库时，每个 MapTask 处理一部分的查询操作，通过实现多个部分之间的并行查询来提高多值查询的效率. 此时每个部分的内部仍旧需要进行数据的全扫描.

2) 采用索引技术优化多值查询：很多的研究工作尝试从添加多维索引的角度来加速 NoSQL 数据库的查询速度. 表 6 列举了已有的一些解决方案的对比.

Table 6 Comparison between Multiple Fields Query Using Index

表 6 采用索引加速多值查询的方案对比

Types of indexes	Advantages	Disadvantages	Examples
Multiple one-dimensional Indexes	Achievement and maintenance are easy	Less efficient multidimensional query and high space redundancy	ITHbase IHbase CCIndex Asynchronous views
Multidimensional Index	Good scalability	Achievement and maintenance of index are complex	RT-CAN QT-Chord EMINC A-Tree
Spatial Index	High write throughput and low cost of maintenance	Consistency maintenance is complex	MD-HBase UQE-Index

ITHbase[50], IHbase[51], CCIndex[52] 和 Asynchronous views[53] 是典型的采用多个一维二级索引来加速多值查询优化的实现方案. 其中 ITHbase 和 IHbase 是两个开源的实现方案，ITHbase 主要关注于数据一致性，事务性是其重要特性. IHbase 与 ITHbase 类似，从 HBase 源码级别进行了扩展，重新定义和实现了 Server，Client 端处理逻辑. CCIndex(complemental clustering index)是中国科学院提出的另外一种索引结构，它在索引中既存储索引项，也存储记录的其他列的数据，以便在查询时直接在索引表中通过顺序扫描找到相应的数据，大幅度减少查询时间. 该方法本质是以空间代价来换取查询效率. CCIndex 的索引更新代价比较高，会影响系统的吞吐量. 索引创建以后不能够动态增加或修改. Asynchronous views 以异步视图的方式来实现非主键的查询，提出了两种视图方案：远端视图表(remote view tables, RVTs)和局部视图表(local view tables, LVTs).

RT-CAN[54] 采用多维索引加速多值查询. 其局部索引采用 R-tree，全局索引中采用了能够支持多维查询的 CAN 覆盖网络. QT-Chord[55] 是另一种双层索引结构，它的局部索引采用的是改进的四叉树 IMX-CIF quad-tree，全局索引采用的 Chord 覆盖网络. EMINC[56] 针对每个局部节点建立一个 KD-tree，然后选择 KD-tree 的部分节点作为全局索引. 每一个局部索引节点被看成是一个多个维度组成的

立方体,然后在全局索引中用 R-tree 对这些立方体进行索引. A-Tree[57]提出了另外一种方案. 基本思路是:针对每一个存储节点构建 R-tree,同时创建一个 Bloom filter(布隆过滤器). 这样在进行点查询时,首先通过 Bloom filter 进行验证,如果查询点不在其中,则不再进行 R-tree 查询,否则继续进行 R-tree 查询.

MD-HBase[58]提出一种基于空间目标排序的索引方案. 基于空间目标排序的索引方法的基本思想是:按照一定规则将覆盖整个研究区的范围划分为大小相等的格子,并给每一格网分配一个编号,用这些编号为空间目标生成一组具有代表意义的数字. 其实质是将 k 维空间的实体映射到一维空间,因此可以利用现有数据库管理系统中比较成熟的一维索引技术. UQE-Index[59]主要针对海量物联网应用场景的时空特性,在时间维度上把数据分成当前数据和历史数据,对当前数据和历史数据进行不同粒度的索引,对当前数据,在时间段和子空间上进行索引,从而减少索引更新的次数,降低索引维护的代价,提高系统的吞吐量;对历史数据,批量地建立记录级别的索引;在建立子空间索引时,为了确保数据分布均匀,采用 KD Tree 进行动态划分. 但是如果所有的数据都需要经过 KD Tree 来索引也会带来较高的代价,会影响数据的插入速度,因此可以对数据进行采样,对采样得到的数据利用 KD Tree 进行索引,从而得到空间上的划分方案.

就已有方案来看,针对 NoSQL 数据库上的查询优化技术都并不成熟,仍有很多关键性问题亟待解决.

3.1.4 数据分析技术

数据分析是 Google 最核心业务,每一次简单的网络点击背后都需要进行复杂的分析过程,因此 Google 对其分析系统进行不断的升级改造之中. MapReduce 是 Google 最早采用的计算模型,适用于批处理,其具体内容已在上一节介绍. 图是真实社会中广泛存在的事物之间联系的一种有效表示手段,因此对图的计算是一种常见的计算模式,而图计算会涉及到在相同数据上的不断更新以及大量的消息传递,如果采用 MapReduce 去实现,会产生大量不必要的序列化和反序列化开销. 现有的图计算系统并不适用于 Google 的应用场景,因此 Google 设计并实现了 Pregel 图计算模型. Pregel[60]是 Google 继 MapReduce 之后提出的又一个计算模型,与 MapReduce 的离线批处理模式不同,它主要用于图的计算. 该模型的核心思想源于著名的 BSP[61]计算模型. Dremel[62]是 Google 提出的一个适用于 Web 数据级别的交互式数据分析系统,通过结合列存储和多层次的查询树,Dremel 能够实现极短时间内的海量数据分析. Dremel 支持着 Google 内部的一些重要服务,比如 Google 的云端大数据分析平台 Big Query[63]. Google 在 VLDB 2012 发表的文献[64]中介绍了一个内部名称为 PowerDrill 的分析工具,PowerDrill 同样采用了列存储,且使用了压缩技术将尽可能多的数据装载进内存. PowerDrill 与 Dremel 均是 Google 的大数据分析工具,但是其关注的应用场景不同,实现技术也有很大差异. Dremel 主要用于多数据集的分析,而 PowerDrill 则主要应用于大数据量的核心数据集分析,数据集的种类相较于 Dremel 的应用场景会少很多. 由于 PowerDrill 是设计用来处理少量的核心数据集,因此对数据处理速度要求极高,所以其数据应当尽可能地驻留在内存,而 Dremel 的数据则存储在磁盘中. 除此之外,PowerDrill 与 Dremel 在数据模型、数据分区等方面都有明显的差别. 从实际的执行效率来看,Dremel 可以在几秒内处理 PB 级的数据查询,而 PowerDrill 则可以在 30~40 s 里处理 7820 亿个单元格的数据,处理速度快于 Dremel. 二者的应用场景不同,可以相互补充.

微软提出了一个类似 MapReduce 的数据处理模型,称之为 Dryad[65],Dryad 模型主要用来构建支持有向无环图(directed acycline graph, DAG)类型数据流的并行程序. Cascading[66]通过对 Hadoop MapReduce API 的封装,支持有向无环图类型的应用. Sector/sphere[67]可以视为一种流式的 MapReduce,它由分布式文件系统 Sector 和并行计算框架 sphere 组成. Nephele/PACTs[68]则包括 PACTs(parallelization contracts)编程模型和并行计算引擎 Nephele. MapReduce 模型基本成为了批处理类应用的标准处理模型,很多应用开始尝试利用 MapReduce 加速其数据处理.

实时数据处理是大数据分析的一个核心需求. 很多研究工作正是围绕这一需求展开的. 前面介绍了大数据处理的两种基本模式,而在实时处理的模式选择中,主要有 3 种思路:

1) 采用流处理模式. 虽然流处理模式天然适合实时处理系统,但其适用领域相对有限. 流处理模型的应用主要集中在实时统计系统、在线状态监控等.

2) 采用批处理模式. 近几年来,利用批处理模

型开发实时系统已经成为研究热点并取得了很多成果. 从增量计算的角度出发,Google 提出了增量处理系统 Percolator[47],微软则提出了 Nectar[69] 和 DryadInc[70]. 三者均实现了大规模数据的增量计算,但是这些系统和 MapReduce 并不兼容,因此 Incoop[71] 和 IncMR[72] 实现了 MapReduce 框架下的增量计算. Yahoo 的 Nova[73] 则支持有状态的增量数据计算模式. HOP[74] 在 MapReduce 处理的过程中引入管道(pipeline)的概念. 在保证 Hadoop 容错性的前提下,使数据在各个任务间以管道的方式交互,增加了任务的并发性,提高了数据处理的实时性. 中国人民大学 WAMDM 实验室在 HOP 基础上开发的 COLA 系统[75] 在 HOP 系统的基础上增加了数据采样、结果估计、置信区间计算等功能模块,一定程度上提高了 HOP 的实时性. 原位分析可以避免将文件集中传输到分析服务器上的通信开销,大大提高了实时性. 文献[76-77]从原位分析的角度出发,分别实现了针对大规模日志分析的原位 MapReduce(In-situ MapReduce)和 Continuous MapReduce. 原始的 MapReduce 模型并不能很好地支持迭代计算,计算代价很大. 而迭代计算是图计算、数据挖掘、机器学习等领域常见的运算模式,不少研究工作通过改进 MapReduce 模型迭代计算的效率来提高其实时性. HaLoop[78] 通过在各个 task tracker 对数据进行缓存(cache)和创建索引(index)的方式来减少磁盘 IO,并提供了一套新的编程接口. 但是 HaLoop 的动静态数据无法分离,且没有一个客观的停止迭代的标准. Twister[79] 系统将 Hadoop 的全部数据存放在内存中,采用独立模块传递所有的消息和数据. 但是数据驻留内存的限制使其难以实用,且其计算模型的抽象度不高,支持的应用也很有限. Twister 仍处于初步的研究阶段. iMapReduce[80] 介绍了一种基于 MapReduce 的迭代模型,但是它的静态调度策略和粗粒度的 task 可能会导致资源利用不佳和负载不均. iHadoop[81] 实现了 MapReduce 的异步迭代,但是在 task 之间的复用上并无太大改进. PrIter[82] 是在 Hadoop 的基础上开发的,支持带优先级的迭代计算,能够保证迭代过程的快速收敛,适合 top-k 之类的在线查询. 最新版本的 PrIter 已经支持基于内存和基于文件的数据存储方式. Spark[83] 将中间结果存放在内存中,支持除 Map 和 Reduce 之外的多种操作类型. 但是 Spark 不适用异步细粒度更新状态的应用,同时在容错性方面有待提升. Facebook 结合自己的应用场景构建了实时的 Hadoop 系统[84],主要是实现了高可用的 NameNode,对并发读和实时负载性能进行了优化,改造 HBase 使其适合真实的实时生产环境.

3) 二者的融合. 有不少研究人员尝试将流处理和批处理模式进行融合,主要思路是利用 MapReduce 模型实现流处理. 文献[85]着重探讨了将 MapReduce 模型应用到流处理这种单遍分析(one-pass analytics)的应用时在架构上应当进行怎样的调整. 在此分析基础上,文献[86]介绍了一种利用 MapReduce 实现的适用于单遍分析的可扩展平台. DEDUCE 系统[87] 扩展了 IBM 的流处理软件 System S,使其支持 MapReduce. C-MR 系统[88] 通过 3 个方面的工作实现了支持流处理的持续型 MapReduce(Continuous-MapReduce):①将并行流处理中的窗口概念透明的扩展到 MapReduce 模型中;②有效结合了包括 CPU 和 GPU 在内的多种异构计算能力;③支持灵活、动态的工作流调度. M^3[89] 在 Hadoop 系统基础上进行扩展,绕开 HDFS 的限制,实现了一个全内存处理的高效流处理系统. StreamMapReduce[90] 结合事件流处理(event stream processing)的特点,对 MapReduce 中的 Mapper 和 Reducer 进行重新定义,增加了持续的、低延迟的数据处理能力. 文献[91]认为大数据时代快速数据(fast data)的处理是一个非常典型的场景,比如数据流. 在充分调研基础上,作者认为原始的 MapReduce 框架不适合处理快速数据. 结合快速数据的特点,文中设计了一个类似 MapReduce 的框架——MapUpdate,并在该框架基础上实现了一个原型系统 Muppet. 和上述这些系统相比,SSS[92] 最大的特点就是在支持快速流处理的同时也能够支持大规模静态数据的处理,也就是说兼具流处理和批处理. 文献[93]中提出名为离散流(discretized streams)的编程模型,并在 Spark 基础上实现了一个原型系统 Spark Streaming.

3.2 大数据处理工具

关系数据库在很长的时间里成为数据管理的最佳选择,但是在大数据时代,数据管理、分析等的需求多样化使得关系数据库在很多场景不再适用. 本节将对现今主流的大数据处理工具进行一个简单的归纳和总结.

Hadoop 是目前最为流行的大数据处理平台. Hadoop 最先是 Doug Cutting 模仿 GFS,MapReduce 实现的一个云计算开源平台,后贡献给 Apache. Hadoop 已经发展成为包括文件系统

(HDFS)、数据库(HBase、Cassandra)、数据处理(MapReduce)等功能模块在内的完整生态系统(Ecosystem).某种程度上可以说 Hadoop 已经成为大数据处理工具事实上的标准.

对 Hadoop 改进并将其应用于各种场景的大数据处理已经成为新的研究热点.主要的研究成果集中在对 Hadoop 平台性能的改进[94-97]、高效的查询处理[98-103]、索引构建和使用[104-105]、在 Hadoop 之上构建数据仓库[106-108]、Hadoop 和数据库系统的连接[109-111]、数据挖掘[112-115]、推荐系统[116-118]等.

除了 Hadoop,还有很多针对大数据的处理工具.这些工具有些是完整的处理平台,有些则是专门针对特定的大数据处理应用.

表 7 归纳总结了现今一些主流的处理平台和工具,这些平台和工具或是已经投入商业使用,或是开源软件.在已经投入商业使用的产品中,绝大部分也是在 Hadoop 基础上进行功能扩展,或者提供与 Hadoop 的数据接口.

Table 7 List of Big Data Tools

表 7 大数据工具列表

Category		Examples
Platform	Local	Hadoop, MapR, Cloudera, Hortonworks, InfoSphere BigInsights, ASTERIX
	Cloud	AWS, Google Compute Engine, Azure
Database	SQL	Greenplum, Aster Data, Vertica
	NoSQL	HBase, Cassandra, MongoDB, Redis
	NewSQL	Spanner, Megastore, F1
Data Warehouse		Hive, HadoopDB, Hadapt
Data Processing	Batch	MapReduce, Dryad
	Stream	Storm, S4, Kafka
Query Language		HiveQL, Pig Latin, DryadLINQ, MRQL, SCOPE
Statistic and Machine Learning		Mahout, Weka, R
Log Processing		Splunk, Loggly

4 大数据时代面临的新挑战

综上所述,大数据时代的数据存在着如下几个特点:多源异构;分布广泛;动态增长;先有数据后有模式.正是这些与传统数据管理迥然不同的特点,使得大数据时代的数据管理面临着新的挑战,

下面将对其中的主要挑战进行详细分析.

4.1 大数据集成

数据的广泛存在性使得数据越来越多地散布于不同的数据管理系统中,为了便于进行数据分析需要进行数据的集成.数据集成看起来并不是一个新的问题,但是大数据时代的数据集成却有了新的需求,因此也面临着新的挑战.

1) 广泛的异构性.传统的数据集成中也会面对数据异构的问题,但是在大数据时代这种异构性出现了新的变化.主要体现在:①数据类型从以结构化数据为主转向结构化、半结构化、非结构化三者的融合.②数据产生方式的多样性带来的数据源变化.传统的电子数据主要产生于服务器或者是个人电脑,这些设备位置相对固定.随着移动终端的快速发展,手机、平板电脑、GPS 等产生的数据量呈现爆炸式增长,且产生的数据带有很明显的时空特性.③数据存储方式的变化.传统数据主要存储在关系数据库中,但越来越多的数据开始采用新的数据存储方式来应对数据爆炸,比如存储在 Hadoop 的 HDFS 中.这就必然要求在集成的过程中进行数据转换,而这种转换的过程是非常复杂和难以管理的.

2) 数据质量.数据量大不一定就代表信息量或者数据价值的增大,相反很多时候意味着信息垃圾的泛滥.一方面很难有单个系统能够容纳下从不同数据源集成的海量数据;另一方面如果在集成的过程中仅仅简单地将所有数据聚集在一起而不作任何数据清洗,会使得过多的无用数据干扰后续的数据分析过程.大数据时代的数据清洗过程必须更加谨慎,因为相对细微的有用信息混杂在庞大的数据量中.如果信息清洗的粒度过细,很容易将有用的信息过滤掉.清洗粒度过粗又无法达到真正的清洗效果,因此在质与量之间需要进行仔细的考量和权衡.

4.2 大数据分析(analytics)

传统意义上的数据分析主要针对结构化数据展开,且已经形成了一整套行之有效的分析体系.首先利用数据库来存储结构化数据,在此基础上构建数据仓库,根据需要构建数据立方体进行联机分析处理(online analytical processing, OLAP),可以进行多个维度的下钻(drill-down)或上卷(roll-up)操作.对于从数据中提炼更深层次的知识的需求促使数据挖掘技术的产生,并发明了聚类、关联分析等一系列在实践中行之有效的方法.这一整套处理流程在处理相对较少的结构化数据时极为高效.但是随着大数据时代的到来,半结构化和非结构化数据量的迅猛增长,给传统的分析技术带来了巨大的冲击和挑战,主要体现在:

1) 数据处理的实时性(timeliness).随着时间的流逝数据中所蕴含的知识价值往往也在衰减,因此很多领域对于数据的实时处理有需求.随着大数据时代的到来,更多应用场景的数据分析从离线(offline)转向了在线(online),开始出现实时处理的需求,比如 KDD 2012 最佳文献[119]所探讨的实时广告竞价问题.大数据时代的数据实时处理面临着一些新的挑战,主要体现在数据处理模式的选择及改进.在实时处理的模式选择中主要有 3 种思路:即流处理模式、批处理模式以及二者的融合.相关研究成果在 3.1.4 节已经有详细介绍.虽然已有的研究成果很多,但是仍未有一个通用的大数据实时处理框架.各种工具实现实时处理的方法不一,支持的应用类型都相对有限,这导致实际应用中往往需要根据自己的业务需求和应用场景对现有的这些技术和工具进行改造才能满足要求.

2) 动态变化环境中索引的设计.关系数据库中的索引能够加速查询速率,但是传统的数据管理中模式基本不会发生变化,因此在其上构建索引主要考虑的是索引创建、更新等的效率.大数据时代的数据模式随着数据量的不断变化可能会处于不断的变化之中,这就要求索引结构的设计简单、高效,能够在数据模式发生变化时很快地进行调整来适应.前面也介绍了通过在 NoSQL 数据库上构建索引来应对大数据挑战的一些方案,但总的来说,这些方案基本都有特定的应用场景,且这些场景的数据模式不太会发生变化.在数据模式变更的假设前提下设计新的索引方案将是大数据时代的主要挑战之一.

3) 先验知识的缺乏.传统分析主要针对结构化数据展开,这些数据在以关系模型进行存储的同时就隐含了这些数据内部关系等先验知识.比如我们知道所要分析的对象会有哪些属性,通过属性我们又能大致了解其可能的取值范围等.这些知识使得我们在数据分析之前就已经对数据有了一定的理解.而在面对大数据分析时,一方面是半结构化和非结构化数据的存在,这些数据很难以类似结构化数据的方式构建出其内部的正式关系;另一方面很多数据以流的形式源源不断地到来,这些需要实时处理的数据很难有足够的时间去建立先验知识.

4.3 大数据隐私问题

隐私问题由来已久,计算机的出现使得越来越多的数据以数字化的形式存储在电脑中,互联网的发展则使数据更加容易产生和传播,数据隐私问题越来越严重.

1) 隐性的数据暴露.很多时候人们有意识地将自己的行为隐藏起来,试图达到隐私保护的目的.但是互联网尤其是社交网络的出现,使得人们在不同的地点产生越来越多的数据足迹.这种数据具有累积性和关联性,单个地点的信息可能不会暴露用户的隐私,但是如果有办法将某个人的很多行为从不同的独立地点聚集在一起时,他的隐私就很可能会暴露,因为有关他的信息已经足够多,这种隐性的数据暴露往往是个人无法预知和控制的.从技术层面来说,可以通过数据抽取和集成来实现用户隐私的获取.而在现实中通过所谓的"人肉搜索"的方式往往能更快速、准确地得到结果,这种人肉搜索的方式实质就是众包(crowdsourcing).大数据时代的隐私保护面临着技术和人力层面的双重考验.

2) 数据公开与隐私保护的矛盾.如果仅仅为了保护隐私就将所有的数据都加以隐藏,那么数据的价值根本无法体现.数据公开是非常有必要的,政府可以从公开的数据中来了解整个国民经济社会的运行,以便更好地指导社会的运转.企业则可以从公开的数据中了解客户的行为,从而推出针对性的产品和服务,最大化其利益.研究者则可以利用公开的数据,从社会、经济、技术等不同的角度来进行研究.因此大数据时代的隐私性主要体现在不暴露用户敏感信息的前提下进行有效的数据挖掘,这有别于传统的信息安全领域更加关注文件的私密性等安全属性.统计数据库数据研究中最早开展数据隐私性技术方面的研究,近年来逐渐成为相关领域的研究热点.文献[120]中提出保护隐私的数据挖掘(privacy preserving data mining)这一概念,很多学者开始致力于这方面的研究.主要集中于研究新型的数据发

布技术,尝试在尽可能少损失数据信息的同时最大化地隐藏用户隐私.但是数据信息量和隐私之间是有矛盾的,因此尚未出现非常好的解决办法.Dwork在2006年提出了新的差分隐私(differential privacy)[121]方法.差分隐私保护技术可能是解决大数据中隐私保护问题的一个方向,但是这项技术离实际应用还很远.

3)数据动态性.大数据时代数据的快速变化除了要求有新的数据处理技术应对之外,也给隐私保护带来了新的挑战.现有隐私保护技术主要基于静态数据集,而在现实中数据模式和数据内容时刻都在发生着变化.因此在这种更加复杂的环境下实现对动态数据的利用和隐私保护将更具挑战.

4.4 大数据能耗问题

在能源价格上涨、数据中心存储规模不断扩大的今天,高能耗已逐渐成为制约大数据快速发展的一个主要瓶颈.从小型集群到大规模数据中心都面临着降低能耗的问题,但是尚未引起足够多的重视,相关的研究成果也较少.在大数据管理系统中,能耗主要由两大部分组成:硬件能耗和软件能耗,二者之中又以硬件能耗为主.理想状态下,整个大数据管理系统的能耗应该和系统利用率成正比.但是实际情况并不像预期情况,系统利用率为0时仍然有能量消耗[122].针对这个问题,《纽约时报》和麦肯锡经过一年的联合调查,最终在《纽约时报》上发表文章"Power, pollution and the Internet"[123].调查显示Google数据中心年耗电量约为300万瓦,而Facebook则在60万瓦左右.最令人惊讶的是在这些巨大的能耗中,只有6%~12%的能量被用来响应用户的查询并进行计算.绝大部分的电能用以确保服务器处于闲置状态,以应对突如其来的网络流量高峰,这种类型的功耗最高可以占到数据中心所有能耗的80%.从已有的一些研究成果来看,可以考虑以下两个方面来改善大数据能耗问题:

1)采用新型低功耗硬件.从《纽约时报》的调查中可以知道绝大部分的能量都耗费在磁盘上.在空闲的状态下,传统的磁盘仍然具有很高的能耗,并且随着系统利用率的提高,能耗也在逐渐升高.新型非易失存储器件的出现,给大数据管理系统带来的新的希望.闪存、PCM等新型存储硬件具有低能耗的特性.虽然随着系统利用率的提高,闪存、PCM等的能耗也有所升高,但是其总体能耗仍远远低于传统磁盘.

2)引入可再生的新能源.数据中心所使用的电能绝大部分都是从不可再生的能源中产生的.如果能够在大数据存储和处理中引入诸如太阳能、风能之类的可再生能源,将在很大程度上缓解能耗问题.这方面的工作很少,文献[124]探讨了如何利用太阳能构建一个绿色环保的数据库.

4.5 大数据处理与硬件的协同

硬件的快速升级换代有力地促进了大数据的发展,但是这也在一定程度上造成了大量不同架构硬件共存的局面.日益复杂的硬件环境给大数据管理带来的主要挑战有:

1)硬件异构性带来的大数据处理难题.整个数据中心(集群)内部不同机器之间的性能会存在着明显的差别,因为不同时期购入的不同厂商的服务器在IOPS和CPU处理速度等性能方面会有很大的差异.这就导致了硬件环境的异构性(heterogeneous),而这种异构性会给大数据的处理带来诸多问题.一个典型的例子就是MapReduce任务过程中,其总的处理时间很大程度上取决于Map过程中处理时间最长的节点.如果集群中硬件的性能差异过大,则会导致大量的计算时间浪费在性能较好的服务器等待性能较差的服务器上.这种情况下服务器的线性增长并不一定会带来计算能力的线性增长,因为"木桶效应"制约了整个集群的性能.一般的解决方案是考虑硬件异构的环境下将不同计算强度的任务智能的分配给计算能力不同的服务器,但是当这种异构环境的规模扩展到数以万计的集群时问题将变得极为复杂.

2)新硬件给大数据处理带来的变革.所有的软件系统都是构建在传统的计算机体系结构之上,即CPU-内存-硬盘三级结构.CPU的发展一直遵循着摩尔定律,且其架构已经从单核转入多核.因此需要深入研究如何让软件更好地利用CPU多核心之间的并发机制.由于机械特性的限制,基于磁性介质的硬盘(hard disk drive, HDD)的读写速率在过去几十年中提升不大,而且未来也不太可能出现革命性的提升.基于闪存的固态硬盘(solid state disk, SSD)的出现从硬件层为存储系统结构的革新提供了支持,为计算机存储技术的发展和存储能效的提高带来了新的契机.SSD具有很多优良特性,主要包括极高的读写性能、抗震性、低功耗、体积小等,因此正得到越来越广泛的应用.但是直接将SSD应用到现有的软件上并不一定会带来软件性能的大幅提升.Lee等人的研究[125]表明虽然SSD的读写速率是HDD的60~150倍,基于SSD的数据库系统的

查询时间却仅仅提升了不到 10 倍. 二者之间的巨大差距主要是由 SSD 的一些特性造成的,这些特性包括:SSD 写前擦除特性导致的读写操作代价不对称、SSD 存储芯片的擦除次数有限等. 软件设计之时必须仔细考虑这些特性才能够充分利用 SSD 的优良特性. 与大容量磁盘和磁盘阵列相比,固态硬盘的存储容量相对较低,单位容量的价格远高于磁盘. 且不同类型的固态硬盘产品性能差异较大,将固态硬盘直接替换磁盘应用到现有的存储体系中难以充分发挥其性能. 因此现阶段可以考虑通过构建 HDD 和 SSD 的混合存储系统来解决大数据处理问题. 当前混合存储系统的实现主要有 3 种思路:HDD 作为内存的扩展充当 SSD 写缓冲[126-127];HDD 和 SSD 同作二级存储[128-132];SSD 用作内存的扩展充当 HDD 读写缓冲[133-137]. 国外的 Google,Facebook,国内的百度、淘宝等公司已经开始在实际运营环境中大规模地使用混合存储系统来提升整体性能. 在这三级结构之中,内存的发展处于一个相对缓慢的阶段,一直没有出现革命性的变化. 构建任何一个软件系统都会假设内存是一个容量有限的易失结构体. 随着以 PCM 为代表的 SCM 的出现,未来的内存极有可能会兼具现在内存和磁盘的双重特性,即处理速度极快且非易失. 虽然 PCM 尚未有可以大规模量产的产品推出,但是各大主流厂商都对其非常重视,三星电子在 2012 年国际固态电路会议(ISSCC 2012)上发表了采用 20nm 工艺制程的容量为 8 GB 的 PCM 元件. 一旦 PCM 能够大规模地投入使用,必将给现有的大数据处理带来一场根本性的变革. 譬如前面提到的流处理模式就可以不再将内存的大小限制作为算法设计过程中的一个主要考虑因素.

4.6 大数据管理易用性(usability)问题

从数据集成到数据分析,直到最后的数据解释,易用性应当贯穿整个大数据的流程. 易用性的挑战突出体现在两个方面:首先大数据时代的数据量大,分析更复杂,得到的结果形式更加多样化. 其复杂程度已经远远超出传统的关系数据库. 其次大数据已经广泛渗透到人们生活的各个方面,很多行业都开始有了大数据分析的需求. 但是这些行业的绝大部分从业者都不是数据分析的专家,在复杂的大数据工具面前,他们只是初级的使用者(naïve users). 复杂的分析过程和难以理解的分析结果限制了他们从大数据中获取知识的能力. 这两个原因导致易用性成为大数据时代软件工具设计的一个巨大挑战. 关于大数据易用性的研究仍处于一个起步阶段. 从设计学的角度来看易用性表现为易见(easy to discover)、易学(easy to learn)和易用(easy to use). 要想达到易用性,需要关注以下 3 个基本原则[138]:

1) 可视化原则(visibility). 可视性要求用户在见到产品时就能够大致了解其初步的使用方法,最终的结果也要能够清晰地展现出来. 针对 MapReduce 使用复杂的情况,文献[139-141]实现了 MapReduce 程序的自动调优,避免了复杂参数设置的过程,大大减轻了用户调试 MapReduce 程序的负担. Starfish 系统[142]架构 Hadoop 之上,尝试解决 Hadoop 系统性能自动调优的问题. 未来如何实现更多大数据处理方法和工具的简易化和自动化将是一个很大的挑战. 除了功能设计之外,最终结果的展示也要充分体现可视化的原则. 可视化技术是最佳的结果展示方式之一,通过清晰的图形图像展示直观地反映出最终结果. 但是超大规模的可视化却面临着诸多挑战,主要有[143]:原位分析,用户界面与交互设计,大数据可视化,数据库与存储,算法,数据移动、传输和网络架构,不确定性的量化,并行化,面向领域与开发的库、框架以及工具,社会、社区以及政府参与.

2) 匹配原则(mapping). 人的认知中会利用现有的经验来考虑新的工具的使用. 譬如一提到数据库,了解的人都会想到使用 SQL 语言来执行数据查询. 在新工具的设计过程中尽可能将人们已有的经验知识考虑进去,会使得新工具非常便于使用,这就是所谓的匹配原则. MapReduce 模型虽然将复杂的大数据处理过程简化为 Map 和 Reduce 的过程,但是具体的 Map 和 Reduce 函数仍需要用户自己编写,这对于绝大部分没有编程经验的用户而言仍过于复杂. 如何将新的大数据处理技术和人们已经习惯的处理技术和方法进行匹配将是未来大数据易用性的一个巨大挑战. 这方面现在已有一些初步的研究工作. 针对 MapReduce 技术缺乏类似 SQL 标准语言的弱点,研究人员开发出更高层的语言和系统. 典型代表有 Hadoop 的 HiveQL[106] 和 Pig Latin[144]、Google 的 Sawzall[145]、微软的 SCOPE[32] 和 DryadLINQ[146] 以及 MRQL[147] 等. SQL 查询有自动优化的过程,而 MapReduce 并没有. 针对这点,文献[148-149]实现了 MapReduce 的查询优化器. 文献[150]通过调研发现系统 I/O 冗余是由于查询之间的关联(correlations),为了解决这个问题,作者引入了 BSP(batched stream processing)模型,并在 DryadLINQ 中实现了查询优化系统 Comet. 还有部分学者的工作集中在将 SQL 语言自动转化成

MapReduce 任务.比较代表性的系统有 YSmart[151] 和 Tenzing[152] 等.还有一些其他的工作,比如 S4Latin[153] 在 S4 的基础上实现了一个新的数据处理框架,使得用户可以直接用类似查询的方式而不是编程的方式创建新的流应用.这在很大程度上改善了大数据流处理平台 S4 的易用性.

3) 反馈原则(feedback).带有反馈的设计使得人们能够随时掌握自己的操作进程.进度条就是一个体现反馈原则的经典例子.大数据领域关于这方面的工作较少,有部分学者开始关注 MapReduce 程序执行进程的估计[154-156].在传统的软件工程领域,程序出现问题之后有比较成熟的调试工具可以对错误的程序进行交互式的调试,相对容易找到错误的根源.但是大数据时代很多工具其内部结构复杂,对于普通用户而言这些工具近似于黑盒(black box),调试过程复杂,缺少反馈性.文献[157]通过可视化 MapReduce 程序产生的大量日志文件,辅助后期的程序调试过程.PerfXplain[158] 设计并实现了 MapReduce 的简便化调试系统.为了解决大数据云(big data cloud)中程序部署和调试的问题,文献[159]实现了一个可扩展的轻量级 Hadoop 性能分析器 HiTune.如果未来能够在大数据的处理中大范围地引入人机交互技术,使得人们能够较完整地参与整个分析过程,会有效地提高用户的反馈感,在很大程度上提高易用性.

满足以上 3 个基本原则的设计就能够达到良好的易用性.从技术层面来看,可视化、人机交互以及数据起源技术都可以有效地提升易用性.而在这些技术的背后,海量元数据管理的问题是需要我们特别关注的一个问题.元数据是关于数据的数据,数据之间的关联关系以及数据本身的一些属性大都是靠元数据来表示的.可视化技术离不开元数据的支持,因为如果无法准确的表征出数据之间的关系,就无法对数据进行可视化的展示.数据起源技术更是离不开元数据管理技术.因为数据起源需要利用元数据来记录数据之间包括因果关系在内的各种复杂关系,并通过这些信息来进行相关的推断.如何在大规模存储系统中实现海量元数据的高效管理将会对大数据的易用性产生重要影响.

4.7 性能的测试基准(benchmark)

关系数据库产品的成功离不开以 TPC 系列为代表的测试基准的产生.正是有了这些测试基准,才能够准确地衡量不同数据库产品的性能,并对其存在的问题进行改进.目前尚未有针对大数据管理的测试基准,构建大数据测试基准面临的主要挑战有[160-161]:

1) 系统复杂度高.大数据管理系统的类型非常多,很多公司针对自己的应用场景设计了相应的数据库产品.这些产品的功能模块各异,很难用一个统一的模型来对所有的大数据产品进行建模.

2) 用户案例的多样性.测试基准需要定义一系列具有代表性的用户行为,但是大数据的数据类型广泛,应用场景也不尽相同,很难从中提取出具有代表性的用户行为.

3) 数据规模庞大.这会带来了两方面的挑战.首先数据规模过大使得数据重现非常困难,代价很大.其次在传统的 TPC 系列测试中,测试系统的规模往往大于实际客户使用的数据集,因此测试的结果可以准确地代表系统的实际性能.但是在大数据时代,用户实际使用系统的数据规模往往大于测试系统的数据规模,因此能否用小规模数据的测试基准来代表实际产品的性能是目前面临的一个挑战.数据重现的问题可以尝试利用一定的方法来产生测试样例,而不是选择下载某个实际的测试数据集.但是这又涉及到如何使产生的数据集能真实反映原始数据集的问题.

4) 系统的快速演变.传统的关系数据库其系统架构一般比较稳定,但是大数据时代的系统为了适应数据规模的不断增长和性能要求的不断提升,必须不断地进行升级,这使得测试基准得到的测试结果很快就不能反映系统当前的实际性能.

5) 重新构建还是复用现有的测试基准.如果能够在现有的测试基准中选择合适的进行扩展,那么将大大减少构建新的大数据测试基准的工作量.可能的候选测试标准有 SWIM(Statistical Workload Injector for MapReduce)[162], MRBS[163], Hadoop 自带的 GridMix[164], TPC-DS[165], YCSB++[166] 等.

现在已经开始有研究工作开始关注大数据的测试基准的构建,比如一些针对大数据测试基准的会议 WBDB 2012 和 TPCTC 2012 等.但是也有观点认为当前讨论大数据测试基准的构建为时尚早.Chen 等人[161,167] 通过对 7 个应用 MapReduce 技术的实际产品的负载进行了跟踪和分析,认为现在根本无法确定大数据时代的典型用户案例.因此从这个角度来看并不适合构建大数据的测试基准,还有很多基础性的问题亟待解决.

总的来说,构建大数据的测试基准是有必要的.

但是面临的挑战非常多,要想构建一个类似 TPC 的公认的测试标准难度很大.

5 结 论

随着云计算、物联网等的发展,数据呈现爆炸式的增长,人们正被数据洪流所包围,大数据的时代已经到来.正确利用大数据给人们的生活带来了极大的便利,但与此同时也给传统的数据管理方式带来了极大的挑战.本文对最近几年国内外大数据相关的研究成果进行了全面的回顾和总结,介绍了大数据的基本概念,详细分析了大数据管理的关键技术,主要是阐述云计算技术对于大数据管理的基础性作用.本文还着重介绍了目前大数据研究面临的新挑战以及相应的一些研究成果.总的来说,目前对于大数据的研究仍处于一个非常初步的阶段,还有很多基础性的问题有待解决.大数据的几个特征中究竟哪个最重要?面对大数据管理我们需要的是简单的技术上的演变(evolution)还是彻底的变革(revolution)?不同学科的研究者之间怎样协作才能更有利于大数据问题的解决?诸如此类的问题还有许多,要解决大数据问题仍有很长的路要走,期望本文的介绍能给大数据研究同行学者提供一定的参考.

参 考 文 献

[1] Nature. Big Data [EB/OL]. [2012-10-02]. http://www.nature.com/news/specials/bigdata/index.html

[2] Bryant R E, Katz R H, Lazowska E D. Big-Data computing: Creating revolutionary breakthroughs in commerce, science, and society [R]. [2012-10-02]. http://www.cra.org/ccc/docs/init/Big_Data.pdf

[3] Science. Special online collection: Dealing with data [EB/OL]. [2012-10-02]. http://www.sciencemag.org/site/special/data/, 2011

[4] Agrawal D, Bernstein P, Bertino E, et al. Challenges and opportunities with big data—A community white paper developed by leading researchers across the United States [R/OL]. [2012-10-02]. http://cra.org/ccc/docs/init/bigdata whitepaper.pdf

[5] Manyika J, Chui M, Brown B, et al. Big data: The next frontier for innovation, competition, and productivity [R/OL]. [2012-10-02]. http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovation

[6] World Economic Forum. Big data, big impact: New possibilities for international development [R/OL]. [2012-10-02]. http://www3.weforum.org/docs/WEF_TC_MFS_BigDataBigImpact_Briefing_2012.pdf

[7] Big Data Across the Federal Government [EB/OL]. [2012-10-02]. http://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_fact_sheet_final_1.pdf

[8] UN Global Pulse. Big Data for Development: Challenges & Opportunities [R/OL]. [2012-10-02]. http://www.unglobalpulse.org/projects/BigDataforDevelopment

[9] Times N Y. The age of big data [EB/OL]. [2012-10-02]. http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html?pagewanted=all

[10] Grobelnik M. Big-data computing: Creating revolutionary breakthroughs in commerce, science, and society [R/OL]. [2012-10-02]. http://videolectures.net/eswc2012_grobelnik_big_data/

[11] Barwick H. The "four Vs" of Big Data. Implementing Information Infrastructure Symposium [EB/OL]. [2012-10-02]. http://www.computerworld.com.au/article/396198/iiis_four_vs_big_data/

[12] IBM. What is big data? [EB/OL]. [2012-10-02]. http://www-01.ibm.com/software/data/bigdata/

[13] Big data [EB/OL]. [2012-10-02]. http://en.wikipedia.org/wiki/Big_data

[14] Hey T, Tansley S, Tolle K. The Fourth Paradigm: Data-intensive Scientific Discovery [M/OL]. Microsoft Research, Redmond, Washington (2009) http://research.microsoft.com/en-us/collaboration/fourthparadigm/

[15] Lazer D, et al. Computational social science [J]. Science. 2009, 323: 721-723

[16] Watts D J. A twenty-first century science [J]. Nature, 2007, 445(7127): 489

[17] The Economist. Data, data, everywhere—A special report on managing information [EB/OL]. [2012-10-02]. http://www.economist.com/node/15557443

[18] Kumar R. Two computational paradigm for big data [EB/OL]. [2012-10-02]. KDD summer school, 2012. http://kdd2012.sigkdd.org/sites/images/summerschool/Ravi-Kumar.pdf

[19] InformationWeek Report. The big data management challenge [R/OL]. [2012-10-02]. http://reports.informationweek.com/abstract/81/8766/business-intelligence-and-information-management/research-the-big-data-management-challenge.html

[20] Storm [EB/OL]. [2012-10-02]. https://github.com/nathanmarz/storm

[21] Neumeyer L, Robbins B, Nair A, et al. S4: Distributed Stream Computing Platform [C] //Proc of ICDM Workshops 2010. Piscataway, NJ: IEEE, 2010: 170-177

[22] Goodhope K, Koshy J, Kreps J, et al. Building LinkedIn's Real-time Activity Data Pipeline [J]. Data Engineering, 2012, 35(2): 33-45

[23] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters [C] //Proc of OSDI 2004. Berkeley, CA: USENIX Association, 2004: 137-150

[24] Das S. Data Infrastructure at LinkedIn [C/OL] //Proc of the 5th Extremely Large Databases Conf. http://www-conf.slac.stanford.edu/xldb2011/talks/xldb2011__tue__1005__LinkedIn.pdf. 2011

[25] ScholarSpace [EB/OL]. [2012-10-02]. http://www.cdblp.cn/

[26] Haas L. Integrating Extremely Large Data is Extremely Challenging [C/OL] //Proc of XLDB Asia 2012. http://idke.ruc.edu.cn/xldb/www.xldb-asia.org/program.html

[27] Rajaraman A, Jeff Ullman. Mining of Massive Datasets [M/OL]. [2012-10-02]. http://i.stanford.edu/ullman/mmds.html.

[28] Chapman A, Allen M D, Blaustein B. It's About the Data: Provenance as a Tool for Assessing Data Fitness [C] //Proc of the 4th USENIX Workshop on the Theory and Practice of Provenance. Berkeley, CA: USENIX Association, 2012

[29] Hadoop [EB/OL]. [2012-10-02]. http://hadoop.apache.org/index.html

[30] Ghemawat S, Gobioff H, Leung S T. The Google file system [C] //Proc of SOSP 2003. New York: ACM, 2003: 29-43

[31] McKusick K, Quinlan S. GFS: Evolution on fast-forward [J]. Communication of ACM, 2010, 53(3): 42-49

[32] Chaiken R, Jenkins B, Larson P-Å, et al. SCOPE: Easy and efficient parallel processing of massive data sets [J]. PVLDB, 2008, 1(2): 1265-1276

[33] HDFS Architecture Guide [EB/OL]. [2012-10-02]. http://hadoop.apache.org/docs/hdfs/r0.22.0/hdfs_design.html

[34] CloudStore [EB/OL]. [2012-10-02]. http://code.google.com/p/kosmosfs/

[35] Beaver D, Kumar S, Li H C, et al. Finding a Needle in Haystack: Facebook's Photo Storage [C] //Proc of OSDI 2010. Berkeley, CA: USENIX Association, 2010: 47-60

[36] TFS [EB/OL]. [2012-10-02]. http://code.taobao.org/p/tfs/wiki/index/

[37] FastDFS [EB/OL]. [2012-10-02]. http://code.google.com/p/fastdfs/w/list

[38] Brewer E A. Towards robust distributed systems (Invited Talk)[C] //Proc of PODC 2000. New York: ACM, 2000

[39] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data [C] //Proc of OSDI 2006. Berkeley, CA: USENIX Association, 2006: 205-218

[40] DeCandia G, Hastorun D, Jampani M, et al. Dynamo: Amazon's highly available key-value store [C] //Proc of SOSP 2007. New York: ACM, 2007: 205-220

[41] Cooper B F, Ramakrishnan R, Srivastava U, et al. PNUTS: Yahoo!'s hosted data serving platform [J]. PVLDB, 2008, 1(2): 1277-1288

[42] NOSQL Databases [EB/OL]. [2012-10-02]. http://nosql-database.org/

[43] Strauch C. NoSQL Databases [EB/OL]. [2012-10-02]. http://www.christof-strauch.de/nosqldbs.pdf

[44] Baker J, Bond C, Corbett J, et al. Megastore: Providing Scalable, Highly Available Storage for Interactive Services [C] //Proc of CIDR. 2011: 223-234

[45] Corbett J C, Dean J, Epstein M, et al. Spanner: Google's globally-distributed database [C/OL] //Proc of OSDI 2012. Berkeley, CA: USENIX Association, 2012. [2012-10-10]. http://static.googleusercontent.com/external_content/untrusted__dlcp/research.google.com/zh-CN//archive/spanner-osdi2012.pdf

[46] Shute J, Oancea M, Ellner S, et al. F1: The fault-tolerant distributed RDBMS supporting google's ad business [C] //Proc of SIGMOD 2012. New York: ACM, 2012: 777-778

[47] Peng D, Dabek F. Large-scale incremental processing using distributed transactions and notifications [C] //Proc of OSDI 2010. Berkeley, CA: USENIX Association, 2010: 1-15

[48] Iyer S C, Utts M. Help test some next-generation infrastructure [EB/OL]. [2012-10-02]. http://googlewebmastercentral.blogspot.com/2009/08/help-test-some-next-generation.html, 2009

[49] Wang Haixun. KDD summer school, 2012. Managing and Mining Billion-Node Graphs [EB/OL]. [2012-10-02]. http://kdd2012.sigkdd.org/sites/images/summerschool/Haixun-Wang.pdf

[50] ITHbase. [EB/OL]. [2012-10-02]. https://github.com/hbase-trx/hbase-transactional-tableindexed

[51] IHbase. [EB/OL]. [2012-10-02]. http://github.com/ykulbak/ihbase

[52] Zou Yongqiang, Liu Jia, Wang Shicai, et al. CCIndex: A complemental clustering index on distributed ordered tables for multi-dimensional range queries [C] // Proc of NPC 2010. Berlin: Springer, 2010: 247-261

[53] Agrawal P, Silberstein A, Cooper B F, et al. Asynchronous view maintenance for VLSD databases [C] // Proc of SIGMOD 2009. New York: ACM, 2009: 179-192

[54] Wang Jinbao, Wu Sai, Gao Hong, et al. Indexing multi-dimensional data in a cloud system [C] // Proc of SIGMOD 2010. New York: ACM, 2010: 591-602

[55] Ding Linlin, Qiao Baiyou, Wang Guoren, et al. An efficient quad-tree based index structure for cloud data management [C] // Proc of WAIM 2011. Berlin: Springer, 2011: 238-250

[56] Zhang Xiangyu, Ai Jing, Wang Zhongyuan, et al. An efficient multi-dimensional index for cloud data management [C] // Proc of CloudDB 2009. New York: ACM, 2009: 17-24

[57] Papadopoulos A, Katsaros D. A-Tree: Distributed indexing of multidimensional data for cloud computing environments [C] // Proc of CloudCom 2011. Piscataway, NJ: IEEE, 2011: 407-414

[58] Nishimura S, Das S, Agrawal D, et al. MD-HBase: A scalable multi-dimensional data infrastructure for location aware services [C] // Proc of MDM 2011. Piscataway, NJ: IEEE, 2011: 7-16

[59] Ma Youzhong, Rao Jia, Hu Weisong, et al. An efficient index for massive IOT data in cloud environment [C] //Proc of CIKM 2012. New York: ACM, 2012

[60] Malewicz G, Austern M H, Bik A J C, et al. Pregel: A system for large-scale graph processing [C] // Proc of SIGMOD 2010. New York: ACM, 2010:135-146

[61] Leslie G. Valiant: A bridging model for parallel computation [J]. Communication of ACM, 1990, 33(8): 103-111

[62] Melnik S, Gubarev A, Long Jingjing, et al. Dremel: Interactive analysis of web-scale datasets [J]. PVLDB, 2010, 3(1): 330-339

[63] Google BigQuery. [EB/OL]. [2012-10-02]. https://cloud.google.com/products/big-query.html

[64] Hall A, Bachmann O, Büssow R, et al. Processing a trillion cells per mouse click [J]. PVLDB, 2012, 5(11): 1436-1446

[65] Isard M, Budiu M, Yu Yuan, et al. Dryad: Distributed data-parallel programs from sequential building blocks [C] // Proc of EuroSys 2007. New York: ACM, 2007: 59-72

[66] Cascading. [EB/OL]. [2012-10-02]. http://www.cascading.org/

[67] Gu Y, Grossman R L. Sector and sphere: The design and implementation of a high-performance data cloud [J]. Philosophical Transactions of the Royal Society A: Physical, Mathematical and Engineering Sciences, 2009, 367(1897): 2429-2445

[68] Battré D, Ewen S, Hueske F, et al. Nephele/PACTs: A programming model and execution framework for web-scale analytical processing [C] //Proc of SoCC 2010. New York: ACM, 2010: 119-130

[69] Gunda P K, Ravindranath L, Thekkath C A, et al. Nectar: Automatic management of data and computation in datacenters [C] //Proc of OSDI 2010. Berkeley, CA: USENIX Association, 2010: 75-88

[70] Popa L, Budiu M, et al. DryadInc: Reusing work in large-scale computations [C] //Proc of HotCloud 2009. Berkeley, CA: USENIX Association, 2009

[71] Bhatotia P, Wieder A, Rodrigues R, et al. Incoop: MapReduce for incremental computations [C] //Proc of SOCC 2011. New York: ACM, 2011

[72] Yan Cairong, Yang Xin, Yu Ze, et al. IncMR: Incremental data processing based on MapReduce [C] // Proc of Cloud 2012. Piscataway, NJ: IEEE, 2012: 534-541

[73] Olston C, Chiou G, Chitnis L, et al. Nova: Continuous Pig/Hadoop workflows [C] //SIGMOD 2011. New York: ACM, 2011: 1081-1090

[74] Condie T, Conway N, Alvaro P, et al. MapReduce Online [C] // Proc of NSDI 2010. Berkeley, CA: USENIX Association, 2010: 313-328

[75] Shi Yingjie, Meng Xiaofeng, Wang Fusheng, et al. You can stop early with COLA: Online processing of aggregate queries in the cloud [C] //Proc of CIKM 2012. New York: ACM, 2012

[76] Logothetis D, Trezzo C, Webb K, et al. In-situ mapreduce for log processing [C] //Proc of USENIX ATC 2011. Berkeley, CA: USENIX Association, 2011

[77] Trezzo C J. Continuous mapreduce: An architecture for large-scale in-situ data processing [D]. USA: University of California, San Diego, 2010. http://christrezzo.com/ctrezzo-thesis.pdf

[78] Bu Yingyi, Howe B, Balazinska M, et al. HaLoop: Efficient iterative data processing on large clusters [J]. PVLDB, 2010, 3(1): 285-296

[79] Ekanayake J, Li Hui, Zhang Bingjing, et al. Twister: A runtime for iterative MapReduce [C] // Proc of HPDC 2010. New York: ACM, 2010: 810-818

[80] Zhang Y, Gao Q, Gao L, et al. iMapReduce: A distributed computing framework for iterative computation [C] //Proc of IPDPS 2011 DataCloud Workshop. Piscataway, NJ: IEEE, 2011: 1112-1121

[81] Elnikety E, Elsayed E, Ramadan H E. iHadoop: Asynchronous iterations for mapreduce [C] // Proc of CloudCom 2011. Piscataway, NJ: IEEE, 2011: 81-90

[82] Zhang Yanfeng, Gao Qixin, Gao Lixin, et al. PrIter: A distributed framework for prioritized iterative computations [C] // Proc of SoCC 2011. New York: ACM, 2011: 1-14

[83] Zaharia M, Chowdhury M, Franklin M, et al. Spark: Cluster computing with working sets [C] //Proc of HotCloud 2010. Berkeley, CA: USENIX Association, 2010

[84] Borthakur D, Gray J, Sarma J S, et al. Apache hadoop goes realtime at Facebook [C] // Proc of SIGMOD 2011. New York: ACM, 2011: 1071-1080

[85] Mazur E, Li Boduo, Diao Yanlei, et al. Towards scalable one-pass analytics using MapReduce [C] // Proc of IPDPS Workshops 2011. Piscataway, NJ: IEEE, 2011: 1102-1111

[86] Li Boduo, Mazur E, Diao Yanlei, et al. A platform for scalable one-pass analytics using MapReduce [C] // Proc of SIGMOD 2011. New York: ACM, 2011: 985-996

[87] Kumar V, Andrade H, Gedik B, et al. DEDUCE: At the intersection of MapReduce and stream processing [C] // Proc of EDBT 2010. New York: ACM, 2010: 657-662

[88] Backman N, Pattabiraman K, Fonseca R, et al. C-MR: Continuously executing MapReduce workflows on multi-core processors [C] // Proc of the 3rd Int Workshop on MapReduce and Its Applications. New York: ACM, 2012: 1-8

[89] Aly A M, SallamA, Gnanasekaran B M, et al. M3: Stream processing on main-memory MapReduce [C] // Proc of ICDE 2012. Piscataway, NJ: IEEE, 2012: 1253-1256

[90] Brito A, Martin A, Knauth T, et al. Scalable and low-latency data processing with stream MapReduce [C] // Proc of CloudCom 2011. Piscataway, NJ: IEEE, 2011: 48-58

[91] Wang Lam, Liu Lu, Prasad S, et al. Muppet: MapReduce-style processing of fast data [J]. PVLDB, 2012, 5(12): 1814-1825

[92] Ogawa H, Nakada H, Takano R, et al. SSS: An implementation of key-value store based MapReduce framework [C] //Proc of CloudCom 2010. Piscataway, NJ: IEEE, 2010: 754-761

[93] Zaharia M, Das T, Li H, et al. Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters [C] // Proc of HotCloud 2012. Berkeley, CA: USENIX Association, 2012

[94] Seo S, Jang I, Woo K, et al. HPMR: Prefetching and pre-shuffling in shared MapReduce computation environment [C] // Proc of CLUSTER 2009. Piscataway, NJ: IEEE, 2009: 1-8

[95] Dittrich J, Quiané-Ruiz J-A, Jindal A, et al. Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing) [J]. PVLDB, 2010, 3(1): 518-529

[96] Babu S. Towards automatic optimization of MapReduce programs [C] // Proc of SoCC 2010. New York: ACM, 2010: 137-142

[97] Shafer J, Rixner S, Cox A L. The Hadoop distributed filesystem: Balancing portability and performance [C] // Proc of ISPASS 2010. Piscataway, NJ: IEEE, 2010: 122-1

[98] Lu Wei, Shen Yanyan, Chen Su, et al: Efficient processing of *k* nearest neighbor joins using MapReduce [J]. PVLDB, 2012, 5(10): 1016-1027

[99] Zhang Xiaofei, Chen Lei, Wang Min. Efficient multi-way theta-join processing using MapReduce [J]. PVLDB, 2012, 5(11): 1184-1195

[100] Pansare N, Borkar V R, Jermaine C, et al. Online aggregation for large MapReduce jobs [J]. PVLDB, 2011, 4(11): 1135-1145

[101] Silva Y N, Reed J M. Exploiting MapReduce-based similarity joins [C] // Proc of SIGMOD 2012. New York: ACM, 2012: 693-696

[102] Okcan I, Riedewald M. Processing theta-joins using MapReduce [C] // Proc of SIGMOD 2011. New York: ACM, 2011: 949-960

[103] Afrati F N, Das S A, Menestrina D, et al. Fuzzy joins using MapReduce [C] //Proc of ICDE 2012. Piscataway, NJ: IEEE, 2012: 498-509

[104] Gudmundsson G Þ, Amsaleg L, Jónsson B Þ. Distributed high-dimensional index creation using Hadoop, HDFS and C++[C] // Proc of CBMI 2012. Piscataway, NJ: IEEE, 2012: 1-6

[105] Liao Haojun, Han Jizhong, Fang Jinyun. Multi-dimensional index on Hadoop distributed file system [C] //Proc of NAS. Piscataway, NJ: IEEE, 2010: 240-249

[106] Thusoo A, Sarma J S, Jain N, et al. Hive—A petabyte scale data warehouse using Hadoop [C] // Proc of ICDE 2010. Piscataway, NJ: IEEE, 2010: 996-1005

[107] Abouzied A, Bajda-Pawlikowski K, Huang Jiewen, et al. HadoopDB in action: Building real world applications [C] // Proc of SIGMOD 2010. New York: ACM, 2010: 1111-1114

[108] Chen Songting. Cheetah: A high performance, custom data warehouse on top of MapReduce [J]. PVLDB, 2010, 3(2): 1459-1468

[109] Su Xueyuan, Swart G. Oracle in-database hadoop: When mapreduce meets RDBMS [C] // Proc of SIGMOD 2012. New York: ACM, 2012: 779-790

[110] Özcan F, Hoa D, Beyer K S, et al. Emerging trends in the enterprise data analytics: Connecting Hadoop and DB2 warehouse [C] // Proc of SIGMOD 2011. New York: ACM, 2011: 1161-1164

[111] Xu Yu, Kostamaa P, Gao Like. Integrating hadoop and parallel DBMs [C] // Proc of SIGMOD 2010. New York: ACM, 2010: 969-974

[112] Yang Yan, Ni Xianhua, Wang Hongjun, et al. Parallel implementation of ant-based clustering algorithm based on Hadoop [C] // Proc of ICSI 2012. Berlin: Springer, 2012: 190-197

[113] Yang Lai, Shi Zhongzhi, Xu L D, et al. DH-TRIE frequent pattern mining on Hadoop using JPA [C] // Proc of GrC 2011. Piscataway, NJ: IEEE, 2011: 875-878

[114] Nair S, Mehta J. Clustering with Apache Hadoop [C] // Proc of ICWET 2011. New York: ACM, 2011: 505-509

[115] Yang Lai, Shi Zhongzhi. An efficient data mining framework on Hadoop using Java persistence API [C] // Proc of CIT 2010. Piscataway, NJ: IEEE, 2010: 203-209

[116] Chiky R, Ghisloti R, Kazi-Aoul Z. Development of a distributed recommender system using the Hadoop framework [C] // Proc of EGC. 2012: 495-500

[117] Jiang Jing, Lu Jie, Zhang Guangquan, et al. Scaling-up item-based collaborative filtering recommendation algorithm based on Hadoop [C] // Proc of SERVICES 2011. Piscataway, NJ: IEEE, 2011: 490-497

[118] De Pessemier T, Vanhecke K, Dooms S, et al. Content-based recommendation algorithms on the Hadoop MapReduce framework [C] // Proc of WEBIST 2011. New York: ACM, 2011: 237-240

[119] Perlich C, Dalessandro B, Hook R, et al. Bid optimizing and inventory scoring in targeted online advertising [C] // Proc of KDD 2012. New York: ACM, 2012: 804-812

[120] Agrawal R, Srikant R. Privacy preserving data mining [C] // Proc of SIGMOD 2000. New York: ACM, 2000: 439-450

[121] Dwork C. Differential privacy [C] //Proc of ICALP 2006. Berlin: Springer, 2006: 1-12

[122] Härder T, Hudlet V, Ou Y, et al. Energy efficiency is not enough, energy proportionality is needed! [C] //Proc of DASFAA 2011. Berlin: Springer, 2011: 226-239

[123] Times N Y. Power, Pollution and the Internet [EB/OL]. [2012-10-02]. http://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html?pagewanted=all

[124] Chen Cheng, He Bingsheng, Tang Xueyan, et al. Green databases through integration of renewable energy [C/OL]. // Proc of CIDR 2013. [2012-10-02]. http://www.cidrdb.org/2013

[125] Lee S-W, Moon B. Design of flash-based DBMS: An in-page logging approach [C] //Proc of SIGMOD 2007. New York: ACM, 2007: 55-66

[126] Soundararajan G, Prabhakaran V, Balakrishnan M, et al. Extending SSD lifetimes with disk-based write caches [C] // Proc of FAST 2010, Berkeley, CA: USENIX Association, 2010

[127] Yang Puyuan, Jin Peiquan, Yue Lihua. Hybrid storage with disk based write cache [C] //Proc of FlashDB 2011. Berlin: Springer, 2011: 190-201

[128] Koltsidas I, Viglas S D. Flashing up the storage layer [J]. PVLDB, 2008, 1(1): 514-525

[129] Yang Qing, Ren Jin. I-CASH: Intelligently coupled array of SSD and HDD [C] //Proc of HPCA 2011. Piscataway, NJ: IEEE, 2011: 278-289

[130] Chen Feng, Koufaty D A, Zhang Xiaodong. Hystor: Making the best use of solid state drives in high performance storage systems [C] //Proc of ICS 2011. New York: ACM, 2011: 22-32

[131] Payer H, Sanvido M A, Bandic Z Z, et al. Combo drive: Optimizing cost and performance in a heterogeneous storage device [C] //Proc of WISH 2009. New York: ACM, 2009

[132] Zhang Ning, Tatemura Junichi, Patel J M, et al. Towards cost-effective storage provisioning for DBMSs [J]. PVLDB, 2011, 5(4): 274-285

[133] Canim M, Mihaila G A, Bhattacharjee B, et al. SSD bufferpool extensions for database systems [J]. PVLDB, 2010, 3(1/2): 1435-1446

[134] Do Jaeyoung, Zhang Donghui, Patel Jignesh M, et al. Turbocharging DBMS buffer pool using SSDs [C] //Proc of SIGMOD 2011. New York: ACM, 1113-1124

[135] Ou Y, Härder T. Improving database performance using a flash-based write cache [C] //Proc of Proc of FlashDB 2011. Berlin: Springer, 2012: 2-13

[136] Luo T, Lee R, Mesnier M, et al. hStorage-DB: Heterogeneity-aware data management to exploit the full capability of hybrid storage systems [J]. PVLDB, 2012, 5(10): 1076-1087

[137] Kang W-H, Lee S-W, Moon B. Flash-based extended cache for higher throughput and faster recovery [J]. PVLDB, 2012, 5(11): 1615-1626

[138] Norman D A. The Design of Everyday Things [M]. New York: Basic Books, 2002

[139] Babu S. Towards automatic optimization of MapReduce programs [C] // Proc of SoCC 2010. New York: ACM, 2010: 137-142

[140] Jahani E, Cafarella M J, Ré C. Automatic optimization for MapReduce programs [J]. PVLDB, 2011, 4(6): 385-396

[141] Cafarella M J, Ré C. Manimal: Relational optimization for data-intensive programs [C] // Proc of WebDB 2010. New York: ACM, 2010

[142] Herodotou H, Lim H, Luo G, et al. Starfish: A self-tuning system for big data analytics [C] // Proc of CIDR. 2011: 261-272

[143] Wong P C, Shen H-W, Johnson C R, et al. The top 10 challenges in extreme-scale visual analytics [J]. Computer Graphics and Applications, 2012, 32(4): 63-67

[144] Olston C, Reed B, Srivastava U, et al. Pig Latin: A not-so-foreign language for data processing [C] // Proc of SIGMOD 2008. New York: ACM, 2008: 1099-1110

[145] Pike R, Dorward S, Griesemer R, et al. Interpreting the data: Parallel analysis with Sawzall [J]. Scientific Programming, 2005, 13(4): 277-298

[146] Isard M, Yu Y. Distributed data-parallel computing using a high-level programming language [C] // Proc of SIGMOD 2009. New York: ACM, 2009: 987-994

[147] Fegaras L, Li C, Gupta U, et al. XML query optimization in MapReduce [C] //Proc of WebDB 2011. New York: ACM, 2011

[148] Iu M-Y, Zwaenepoel W. HadoopToSQL: A MapReduce query optimizer [C] //Proc of EuroSys 2010. New York: ACM, 2010: 251-264

[149] Fegaras L, Li C, Gupta U. An optimization framework for map-reduce queries [C] //Proc of EDBT 2012. New York: ACM, 2012: 26-37

[150] He Bingsheng, Yang Mao, Guo Zhenyu, et al. Comet: Batched stream processing for data intensive distributed computing [C] //Proc of SoCC 2010. New York: ACM, 2010: 63-74

[151] Lee R, Luo Tian, Huai Yin, et al. YSmart: Yet another SQL-to-MapReduce translator. [C] // Proc of ICDCS 2011. Piscataway, NJ: IEEE, 2011: 25-36

[152] Chattopadhyay B, Lin Liang, Liu Weiran, et al. Tenzing a SQL implementation on the MapReduce framework [J]. PVLDB, 2011, 4(12): 1318-1327

[153] Stoellberger P. S4Latin: Language-based big data streaming [D/OL]. UK: University of Edinburgh, 2011. [2012-10-02]. http://analytical-labs.com/downloads/msc_BigDataStreams.pdf

[154] Morton K, Balazinska M, Grossman D. ParaTimer: A progress indicator for MapReduce DAGs [C] // Proc of SIGMOD 2010. New York: ACM, 2010: 507-518

[155] Morton K, Friesen A, Balazinska M, et al. KAMD: Estimating the progress of MapReduce pipelines [C] // Proc of ICDE 2010. Piscataway, NJ: IEEE, 2010: 681-684

[156] Huang Dachuan, Shi Xuanhua, Ibrahim Shadi, et al. MR-scope: A real-time tracing tool for MapReduce [C] // Proc of HPDC 2010. New York: ACM, 2010: 849-855

[157] Tan Jiaqi, Kavulya S, Gandhi R, et al. Visual, log-based causal tracing for performance debugging of MapReduce systems [C] // Proc of ICDCS 2010. Piscataway, NJ: IEEE, 2010: 795-806

[158] Khoussainova N, Balazinska M, Suciu D. PerfXplain: Debugging MapReduce job performance [J]. PVLDB, 2012, 5(7): 598-609

[159] Dai Jinquan, Huang Jie, Huang Shengsheng, et al. HiTune: Dataflow-based performance analysis for big data cloud [C] //Proc of USENIX ATC 2011. Berkeley, CA: USENIX Association, 2011

[160] Baru C, Bhandarkar M, Nambiar R, et al. Introducting the big data benchmarking community [EB/OL] // Proc of XLDB 2012. [2012-10-02]. http://www-conf. slac. stanford. edu/xldb2012/talks/xldb2012 _tue _LT07 _Baru _etal. pdf

[161] Chen Y. We don't know enough to make a big data benchmark suite—An academia-industry view [R/OL]. Berkeley: EECS Department, University of California, http://www. eecs. berkeley. edu/Pubs/TechRpts/2012/EECS-2012-71. pdf

[162] Chen Yanpei, Ganapathi A, Griffith R, et al. The case for evaluating MapReduce performance using workload suites [C] // Proc of MASCOTS 2011. Piscataway, NJ: IEEE, 2011: 390-399

[163] Sangroya A, Serrano D, Bouchenak S. MRBS: A comprehensive MapReduce benchmark suite [R/OL]. LIG Laboratory, University of Grenoble. [2012-10-02]. http://rr. liglab. fr/research_report/RR-LIG-024_orig. pdf

[164] GridMix. [EB/OL]. [2012-10-02]. http://hadoop. apache. org/docs/mapreduce/current/gridmix. html

[165] TPC-DS. [EB/OL]. [2012-10-02]. http://www. tpc. org/tpcds/

[166] Patil S, Polte M, Ren K, et al. YCSB++: Benchmarking and performance debugging advanced features in scalable table stores [C] //Proc of SoCC 2011. New York: ACM, 2011

[167] Chen Y, Alspaugh S, Katz R. Interactive query processing in big data systems: A cross-industry study of MapReduce workloads [J]. PVLDB, 2012, 5(12): 1802-1813

Meng Xiaofeng, born in 1964, PhD. Professor at Renmin University of China. Executive member of China Computer Federation. His main research interests include cloud data management, Web data management, flash-based databases, privacy protection etc.

Ci Xiang, born in 1986. PhD candidate at Renmin University of China. His main research interests include cloud data management, big data, etc.