软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(1):127−141 [doi: 10.13328/j.cnki.jos.005644]

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

大数据管理系统的历史、现状与未来*

杜小勇[1,2], 卢 卫[1,2], 张 峰[1,2]

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

[2](中国人民大学 信息学院,北京 100872)

通讯作者: 杜小勇, E-mail: duyong@ruc.edu.cn

摘 要: 大数据管理技术正在经历以软件为中心到以数据为中心的计算平台的变迁,传统的关系型数据库管理系统无法满足现在以数据为中心的大数据管理的需求,设计新型大数据管理系统迫在眉睫.首先回顾了数据管理技术的发展历史;之后,从大数据管理的存储、数据模型、计算模式、查询引擎等方面分析了大数据管理系统的现状,指出目前大数据管理系统具有模块化和松耦合的特点,并进一步介绍了大数据管理系统应具备的数据特征、系统特征和应用特征,指出大数据管理系统技术还在快速进化之中,预测未来的大数据管理系统应具备多数据模型并存、多计算模式融合、可伸缩调整、新硬件驱动、自适应调优等特点.

关键词: 大数据管理系统;数据存储;数据模型;模块化;松耦合

中图法分类号: TP311

中文引用格式: 杜小勇,卢卫,张峰.大数据管理系统的历史、现状与未来.软件学报,2019,30(1):127−141. http://www.jos.org.cn/1000-9825/5644.htm

英文引用格式: Du XY, Lu W, Zhang F. History, present, and future of big data management systems. Ruan Jian Xue Bao/Journal of Software, 2019,30(1):127−141 (in Chinese). http://www.jos.org.cn/1000-9825/5644.htm

History, Present, and Future of Big Data Management Systems

DU Xiao-Yong[1,2], LU Wei[1,2], ZHANG Feng[1,2]

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

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

Abstract: Big data management systems are migrating from software-centric computing platforms to data-centric computing platforms, and traditional relational databse management systems (a.b.a. RDBMS) do not entirely meet the need for data-centric data management. Hence, it is urgent to design a new kind of big data management systems. In thisThe paper, we first reviews the history of the development of data management systems. Second, we it analyzes the current situation of big data management systems in terms of data storage, data model, and query engines of big data management systems, and points out that current big data management systems have the characteristics of modularity and loose coupling. After that, the data characteristics and application characteristics of the big data management systems are we introduced the data characteristics and application characteristics of the big data management systems,, and it is pointed out that big data management systems are still rapidly evolving, but do not mature. Finally, the future of big data management systems is we predicted, i.e., the future of big data management systems. Big big data management systems in future should have the

* 基金项目: 国家重点研发计划(2018YFB1004401); 国家自然科学基金(61732014, 61502504, 61802412); 北京市科技计划(Z171100005117002)

Foundation item: National Key Research and Development Program of China (2018YFB1004401); National Natural Science Foundation of China (61732014, 61502504, 61802412); Beijing Municipal Science and Technology Project (Z171100005117002)

本文由“软件学科发展回顾特刊”特约编辑梅宏教授、金芝教授、郝丹副教授推荐.

收稿时间: 2018-07-04; 修改时间: 2018-08-21; 采用时间: 2018-09-17; jos 在线出版时间: 2018-11-20

CNKI 网络优先出版: 2018-11-21 09:52:30, http://kns.cnki.net/kcms/detail/11.2560.TP.20181121.0952.002.html

characteristics of multiple data models coexistence, multi-computation platform fusion, elastic adjustment, new hardware driven, self-adaptive tuning, and so on.

Key words: big data management system; data storage; data model; modularity; loose coupling

近年来,大数据的重要性凸显,世界上的许多国家都把大数据上升到国家战略的高度.实施国家大数据战略,离不开大数据技术的研究.回顾信息技术的发展历史,数据管理技术是信息应用技术的基础.与其他计算机学科相比,数据管理是整个领域为数不多的既有基础理论研究、又有系统软件研制、还有产业支撑的学科.专门从事数据管理系统软件和应用软件研制的甲骨文公司于 2013 年超越 IBM,成为继微软公司之后全球第二大软件公司.如今,历史似乎又正在重演,大数据管理在大数据技术中表现得越来越重要.

大数据管理系统正在经历以软件为中心到以数据为中心的计算平台的变迁.计算机的研制最初是为了满足军事和科学计算的需要,应用软件的开发和系统软件的研制均以硬件为中心开展,且依赖于特定的计算机硬件环境.自微软公司 1980 年推出 MS DOS(MicroSoft Disk Operating System)磁盘操作系统以来,MSDOS 作为当时 IBM 的 PC 和兼容机的基本软件,成为了个人计算机中最普遍的操作系统.随后,微软推出的首个带有图形界面的个人电脑操作系统 Windows 1.0,逐渐成为 PC 机的预装软件.微软操作系统成功推动了底层要素的标准化,即底层硬件的可替代性.具体来说,操作系统统一将硬件标准化为设备,使用同样的接口,这样,操作系统就可以运行在不同的硬件平台,使得底层硬件的可替代性得到了增强.与此同时,微软操作系统也成为新的中心,即,应用软件的开发和系统软件的研制此后均以操作系统软件为中心展开.

随着信息技术的发展,特别是以互联网为代表的大数据应用每天产生巨大的数据量,大数据管理系统也发生了以软件为中心到以数据为中心的计算平台的迁移.例如,谷歌、百度等搜索引擎公司存储的网页数据越来越多,逐渐成为网络数据的集中存放仓库,并以这些数据为中心开展各项服务.据统计,2013 年,谷歌大约存储了 10EB(Exabyte,10^{18} bytes)的磁盘数据.如何存储和管理如此巨量的数据,是目前研究的热点.不仅限于搜索引擎公司,其他信息技术公司也都面临同样的大数据管理需求.例如:阿里巴巴集团旗下的蚂蚁金服存储着巨量的交易数据,同时,以此为基础提供征信服务[1,2];腾讯公司的社交数据中心存有大量的用户会话、朋友圈等信息[3],并基于这些信息开发新型应用.总之,这些数据不可避免地成为了一个新平台,大数据时代要求我们在以数据为中心的平台上去开发新型数据管理系统和相应的应用系统.

大数据管理系统正在经历的另一个趋势是基础设施化,基础设施是指人们生活中不可或缺的设施服务,例如水电等公共服务、飞机公路等交通设施等.基础设施必须具备 3 个基本特征:第一,基础性,即社会运行不可缺少的东西;第二,规模性,只有达到了与社会经济状况相适应的规模才能提供有效的服务;第三,可靠性,不能经常出错,要能提供持续稳定的服务.我们正在步入数字经济和数字社会时代,软件作为一种使能技术,在数字社会中具有不可替代的作用,软件基础设施化的趋势也越来越明显,并具有其独特的表现:第一,基础性,计算作为数字社会中最重要、最基础的服务,需要通过软件来提供,计算能力通过软件的定义,可以呈现为丰富多彩的服务形态;第二,规模性,整个社会的计算能力,或通过软件重定义的服务能力,必须互联互通作为一个整体才能成为社会的基础设施;第三,可靠性.基础设施化的软件必须能够提供稳定的、持续的、高效的在线服务.大数据管理系统正在经历软件的基础设施化进程.软件服务的种类很多,其中最重要的服务就是数据服务.云计算不仅是计算资源的汇聚,更是数据资源的汇聚.这些数据资源之间通过数据库软件实现互联、互通,并向公众提供数据的存储与组织以及查询、分析、维护、安全性等管理服务.这样的数据库软件我们称其为大数据管理系统.大数据时代要求我们根据软件的基础设施化去开发大数据管理系统和相应的应用系统.

结合大数据时代所处的两个趋势特征,本文首先回顾数据管理技术的发展历史(见第 1 节).之后,从数据存储、组织模型、计算模式和分析引擎等维度深入剖析大数据管理系统的现状(见第 2 节),并指出大数据管理系统应具备的数据特征、系统特征和应用特征(见第 3 节).随后,本文进一步对大数据管理系统的未来发展趋势进行展望(见第 4 节).最后对全文进行总结(见第 5 节).

1 数据库管理系统的发展历史回顾

数据库管理系统的功能是伴随着对数据的组织和管理以及应用的需求而不断发展起来的.第 1 代系统的功能主要集中在数据的组织与存储,数据的组织以层次和网状模型为代表,多种链表结构作为存储方式.这个时期的数据库系统可以看作是一种数据组织与存取的工具.第 2 代系统主要围绕 OLTP 应用展开,在关系模型和存储技术的基础上,重点发展了事务处理子系统、查询优化子系统、数据访问控制子系统.第 3 代系统主要围绕 OLAP 应用展开,重点在于提出高效支持 OLAP 复杂查询的新型数据组织技术,包括 CUBE 和列存储等技术以及 OLAP 分析前端工具.第 4 代系统主要围绕大数据应用展开,重点在分布式可扩展、异地多备份高可用架构、多数据模型支持以及多应用负载类型支持等特性.

1.1 第1代:层次、网状数据库系统

数据库管理系统的发展可以上溯到 20 世纪 60 年代出现的层次数据库技术,当时,计算机已经开始在商业上获得应用,文件作为数据存储的主要设施,已经无法满足对商业应用(如银行业务)中数据项之间的复杂关系进行管理的需求,主要表现在以下几个方面:第一,文件系统是面向单一应用的,也就是说,根据每一个应用的需要,有针对性地设计文件的数据结构,因此,不同的应用要使用同一个文件结构会显得效率低下;第二,文件之间的数据是独立的,如果两个文件的数据存在内在的逻辑关系,要维护这种关系就非常困难甚至是不可能的;第三,文件的组织方式单一,难以满足不同的访问模式对数据高效率访问的需求.

因此,这个时期的信息系统对数据管理的核心需求是提供一种面向系统整体的数据组织与访问功能,简单来说就是存储和访问这两件事情.受制于当时的计算机技术水平,第 1 代的数据库管理系统是层次型的,之后又进一步扩展为网状型数据库.这里所谓的层次型或者网状型指的是系统中的数据组织方式是按照树或者(受限)图来组织的.树由于其每一个节点最多只有一个父节点,因此可以采用更加有效的手段(例如按照树遍历的顺序)来存储数据.网状模型则通过引入“基本层次联系”的概念,将图分解为一组基本层次联系的集合.而基本层次联系实际上就是一个命名的层次联系.因此,这两类数据库本质上还是一样的,都可以用“树”结构来表达和存储数据.尽管这个时期数据管理的功能集中在数据存储组织和数据访问等,但这是第 1 次将数据管理的功能从具体的应用逻辑中分离并独立出来,在数据管理系统的发展历史上是一件里程碑的事情.

对于层次/网状数据库,数据访问的最常见的模式就是根据某一个父节点的值检索子节点的全部或者部分值.例如,查询信息学院计算机系教师张丽的情况,数据的访问就需要从学院到系再到教师这样的路径进行.为了数据访问的高效率,最有效的数据存储方式就是按照树遍历(例如中序遍历)方式访问树节点,并将这些节点的数据邻近存储,兄弟节点之间则用指针进行链接.因此,这个时期的数据库看起来就是“玩”各种数据结构,指针、链表被大量使用.

分层结构最大的好处就是底层系统的稳定性,即,将变化尽可能地限制在单层内部,这会使得系统的稳定性大大增加,减少了系统的维护代价.这是非常重要的一个特征.举例来说,数据库应用由于有了三级模式结构,当外模式结构发生了变化,例如数据项的增减,可以通过重新定义外模式和模式之间的对应关系,而保持下层模式以及内模式不变,从而使数据库的数据存储组织也不需要变化.反过来,当管理员想重新调整数据的存储组织方式时(提高效率),可以通过重新定义模式到内模式的映射,而保持模式不变,从而外模式也无需变化.自然,基于外模式写的应用程序也无需改变.这就是所谓的程序对于数据的透明性,或者说数据对于程序的独立性.

总的来说,第 1 代系统的主要贡献就是首次将数据的存储与访问功能从应用程序中分离出来.

1.2 第2代:关系数据库系统

20 世纪 70 年代是关系数据库形成并实现产品化的年代,主要的代表人物就是 IBM 的埃德加·科德(Edgar F. Codd).1970 年,科德发表题为“大型共享数据库的关系模型”的论文,文中首次提出了数据库的关系模型[4]这一概念.由于关系模型简单明了、具有坚实的数学理论基础,操作语言是描述性的,不用像层次网状模型那样需要描述存取路径(即先访问哪个数据再访问哪个数据)这样的细节,给提高信息系统开发的生产率提供了极大的空间,所以一经提出就受到了学术界和产业界的高度重视和广泛响应.尽管一开始产业界还充斥了对关系数据库

系统性能的怀疑,但是经过科德所开发的 System R 系统的验证,证明关系数据库系统的性能是可以有保障的.这一结论极大地推动了关系数据库技术的发展,关系数据库产品化的活动进入一个高潮,IBM 公司自己也在 System R 系统的基础上推出了 DB2 产品,其他最著名的要数 ORACLE 公司的同名数据库产品了.可以说,20 世纪 80 年代以来,关系数据库迅速取代层次和网状数据库而占领市场.数据库的研究工作也是围绕关系数据库展开的,1975 年召开了第 1 届超大规模数据库大会(very large data bases Conf.,简称 VLDB),在会议录的前言中就曾提到,当今产业届已经出现了一张关系表中有 100 万条记录的“大表”,如何才能确保对这样大表的访问效率?由于科德杰出的贡献,1981 年的图灵奖很自然地授予了这位“关系数据库之父”.他的图灵演讲题目就是“关系数据库:提高生产率的实际基础”,说到了关系数据库成功的关键,那就是这项技术提高了生产率!这正是数据库技术成功背后的“看不见的手”.

为什么能做到这一点呢?

- 第一,描述性的关系数据语言功不可没.SQL 语言的基本结构由 3 个子句构成,分别用于指定关系表、行和列.FROM 子句指定表;WHERE 子句指定对行的选择条件,也就是哪些行符合查询要求,是对行的约束;SELECT 子句指定需要呈现的列,可以看作是对列的限制.从本质上来讲,一个查询就是从一张表中选择出需要的行和列.关系数据库采用了 SQL 语言,是提高信息系统开发效率的重要因素之一;
- 第二,关系数据库系统有完善的确保数据正确的功能,能够避免各种错误可能带来的数据库损害.例如,为了应对各种可能的故障,从程序运行错误,到停电,到存储介质损害等等,故障发生以后,如何使得数据不受到破坏,这是任何一个应用系统都需要考虑的问题,如果需要为每一个应用系统自己去完成这些代码,可想而知,应用的开发效率不可能高.如果数据库系统能以一定的方式确保数据库中的数据不会被各种故障所损害,那么开发应用的时候就可以不用关心这个问题.事实上,在数据库管理系统的代码中,真正用于处理 SQL 语句的部分并不多,大部分的代码都用于处理各种异常;
- 第三,有各种应用开发工具.一个数据库应用从设计到实施有复杂的过程,需要了解信息需求和功能需求,需要进行数据库模式设计,需要编写 SQL 语句等等,如果有各种开发工具,甚至是数据库模式的自动生成工具,以及 SQL 语句的自动生成工具,那么应用开发的效率自然就能成倍地提高.

在关系数据库的关键技术中,最为核心的有查询优化技术和事务管理两个方面,这是关系数据库走向实用必须首先要解决的难题.由于关系数据库语言是基于集合的描述性语言,典型代表就是 SQL,其查询的结果也是一个集合.如何将一个 SQL 语句转换为可以执行的程序(有点类似程序自动生成器),而且要在所有可能的执行计划(程序)中选择出一个效率足够好的加以执行,这就是查询优化器的作用.

由于关系数据库主要用于支撑各种业务系统,因此需要管理业务状态的变化,将这类应用称为 OLTP(online transaction processing),即联机事务处理.“事务”又称为交易,体现的是现实世界的业务逻辑.典型的例子就是银行的转账业务,如果某客户希望将账号 *A* 的钱转到账号 *B* 上去,那么必须保证账号 *A* 和 *B* 的存款数之和在转账前后是一样的,既不能多出来也不能少掉.这是数据库系统必须保障的,否则数据库就无法使用.对于单机系统,这种业务逻辑的维护还比较容易,但当数据库系统是多用户系统时,这件事情就变得非常复杂,需要认真对待和解决.这些问题如果不能圆满解决,无论哪个公司的数据库产品都无法进入实用,最终不能被用户所接受.

事务管理(TM)是数据库的重要部件,它提供对并发事务的调度控制和故障恢复,确保数据库系统的正确运行.詹姆斯·尼古拉·格雷(James Nicholas Gray)在解决这些重大技术问题上发挥了十分关键的作用,为 DBMS 成熟并顺利进入市场做出了重要贡献.其成就汇聚成一部厚厚的专著《Transaction Processing: Conceptsand Techniques》[5],他也众望所归地获得了 1998 年度的图灵奖.

在关系数据库时代,对于数据库系统做出重要贡献的还有一位学者——迈克尔·斯通布雷克(Michael Stonebraker).他在加州大学伯克利分校计算机科学系任教达 29 年,在此期间,领导开发了关系数据库系统 Ingres、对象-关系数据库系统 Postgres、联邦数据库系统 Mariposa,同时还创立了多家数据库公司,包括 Ingres、Illustra、Cohera、StreamBase Systems 和 Vertica 等,将大量研究成果和原型系统实现商业化.他在“One size does not fit all”的思想指导下,开发了一系列的“专用”关系数据库产品,例如流数据管理系统、内存数据库管理系统、

列存储关系数据库系统、科学数据库管理系统等.因“对现代数据库系统底层的概念与实践所做出的基础性贡献”,他在 2015 年获得了图灵奖.

1.3 第3代:数据仓库系统

这个阶段也可以看成是关系数据库的一个自然延伸.随着数据库技术的普及应用,越来越多的数据被存储在数据库中,除了支持业务处理以外,如何让这些数据发挥更大的作用,则是一个亟待解决的问题.由于对这些数据而言,主要是分析,因此这类应用也称为 OLAP(online analytical processing)应用[6],即联机分析处理应用.例如,对于电话详单数据,因为是通话记录,因此一旦发生就成为历史的记录,很少发生事后修改的情况.但是,许多业务员会对电话详单数据感兴趣.例如,按照时间轴去分析不同时间区间通话的数量变化情况,也可以按照区域去分析通话数量的情况等等.尽管关系数据库也能实现上述要求,但是如何让这样的复杂分析高效地执行?需要有特殊的数据组织模式.

星型模型是最常用的数据仓库的数据组织模型,特别适合于联机分析类应用,即 OLAP 应用.所谓星型模型也称为多维模型,就是选定一些属性作为分析的维度,另一些属性作为分析的对象.维属性通常根据值的包含关系会形成一个层次,例如,时间属性可以根据年月周日形成一个层次,地区属性也可以形成街道-区-市这样的层次.为了实现快速分析,可以预先计算出不同粒度的统计结果.例如,如果预先计算了按照周和区为单位某连锁超市的销售额,就可以快速、方便地分析展示各区按照周的顺序的销售额的变化情况.这种采用预先计算的方法可以获得快速联机分析的性能.

数据仓库可以用关系数据库实现(relational OLAP,简称 ROLAP),分别用事实表和维表来存储统计结果和维度结构.也可以用特别的数据模型(CUBE)来实现(multidimensional OLAP,简称 MOLAP),列存储的技术也在这个过程中被提出和应用.同时,支持 OLAP 的前端分析工具也很重要,使得普通用户可以方便地使用数据仓库.

1.4 第4代:大数据管理系统

关系数据库成熟并得以广泛应用后,数据库研究和开发一度走入一段迷茫期.数据库界一直无法打破关系数据库的魔咒,被关系模型和系统的“完美”所陶醉,无法自我突破.提出的一些新概念,例如面向对象数据库系统,很快就被关系数据库所消化,未能形成气候.整个 20 世纪 90 年代都是在这样的气氛中渡过的.斯通布雷克也不能免俗,也难以逃脱关系数据库的束缚.MapReduce[7]出现之后,他曾经激烈地批判过 MapReduce,认为是对数据库技术的巨大倒退(从某些方面来讲,也确实是这样的).所以,大数据处理平台 MapReduce 并不是数据库学者首先提出来的,而是做系统的人提出来的,据说最早的论文是投到数据库顶级会议上,被无情地拒了.这也是数据库学术界需要认真反思的地方.

由于信息技术的高度发展,信息系统所积累的数据越来越多,数据类型也越来越丰富,而且产生的速度非常快.传统的数据库技术“存不下”、无法建模、无法及时入库等问题凸显出来,难以满足应用的需要.在这样的背景下,Google 公司发展了 GFS(Google file system)[8]、MapReduce 和 Bigtable[9].

Google 的这 3 件“武器”后来在 Apache 基金会下面有一个对应的实现,这就是 Hadoop 生态系统.它包括实现了一个分布式文件系统 HDFS(hadoop distributed file system)[10]、一个计算框架 MapReduce 和一个数据库 HBase[11].HDFS 有高容错性的特点,并且可部署在低廉的服务器上,适合那些有着超大数据规模的应用.MapReduce 为海量的数据提供了一个可容错的、高可扩展的、分布式的计算框架,HBase 是一个基于键值对组织模型(逻辑上可以看成是宽表)的分布式数据库,数据按行列混合模式存储在 HDFS 上.MapReduce 可以直接访问 HDFS 上的数据进行数据分析,也可以通过 HBase 间接访问 HDFS 上的数据,以提高分析性能.

很自然地,在 HDFS 上如何表达和管理数据?NoSQL[12]数据库便应运而生.一开始确实是“No SQL”的含义,因为对大容量的非结构化数据的处理需求,都不是 SQL 所擅长的.NoSQL 的重点在于如何表达和存储非结构化数据,先后提出了 Key-Value 结构,也就是有一个 Key,对于一个值(可以是很复杂的非结构化数据),XML 描述的文档、图结构等.后来人们发现:无论从应用程序的继承角度还是提高生产率的角度,SQL 都是不可或缺的工具,因此,在上层提供 SQL 引擎,成为大数据管理系统的共识.

1.5 小 结

从上述数据库管理系统发展的简史中,可以感受到以下几点:第一,数据库管理系统的功能是伴随着应用的发展而不断丰富起来的,因此任何时候,应用的需要才是技术发展的动力;第二,将数据管理的一些功能逐步从业务逻辑中分离出来,形成独立的软件系统,这是提高应用生产效率的有效手段和途径;第三,系统分层是一个好主意,上层可以屏蔽下层的一些实现细节,为更上层提供更简洁的服务;第四,语言或者说接口是定义一个系统的最有效的方法.

2 大数据管理系统的现状

自 20 世纪 70 年代起,关系数据库由于具备严格的关系理论辅助数据建模、数据独立性高,查询优化技术实现突破,逐渐成为数据管理中的主流技术.时至今日,关系数据库仍然是数据管理,特别是涉及到人、财、物等需要精细管理应用的主流技术.关系数据库信守的原则是 one-size-fits-all,认为所有有关数据管理的任务都应该交由关系数据库来解决.进入大数据时代,大数据的许多应用,特别是互联网的应用,比如社交网络、知识图谱、搜索引擎、阿里的“双十一”等数据管理问题,使用传统的关系数据库已经无法满足应用处理的要求,人们开始尝试研制适合自己应用场景的大数据系统.谷歌三件套,包括 GFS、MapReduce 计算框架以及 BigTable 的提出,以及以 Hadoop 为核心的开源生态系统的形成,人们意识到 one-size-does-not-fit-all,即无法使用单一的数据管理系统来解决所有大数据应用的问题.在经历相当长的一段时间的探索之后,人们对数据库系统的各个模块,包括存储系统、数据组织模型、查询处理引擎、查询接口等,依托谷歌管理和分析大数据的设计思路进行了解耦,并从模型、可靠性、可伸缩性、性能等方面对各个模块进行了重新的设计.可以发现:现阶段主流的大数据管理系统具有了明显的分层结构,自底向上分别为大数据存储系统、NoSQL 系统、大数据计算系统、大数据查询处理引擎.各类系统独立发展,并根据大数据应用的实际需要,通过采用松耦合的方式进行组装,构建为完整的大数据管理系统,支撑各类大数据查询、分析与类人智能应用.这实际上就是 one-size-fits-a-bunch 的设计理念.正如周傲英教授指出的:“如果说在数据库时期,解决数据管理问题需要‘削足适履’来使用数据库系统,那么到了大数据时代,人们开始根据每个不同的应用度身定制自己的系统,也就是‘量足制鞋’”[13].

2.1 存储系统

在大数据存储方面,以 HDFS 等为代表的开源系统目前已成为大数据存储领域的标准之一.HDFS 面向大文件(GB 级别及以上)的存储管理,因此在设计上,HDFS 的存取单元数据块(block)比一般单机文件系统的数据存取单元要大得多.较低版本的 HDFS 的数据块默认为 128MB,2.0 版本后的数据块默认大小为 256MB.HDFS 可以运行在数万个基于普通 X86 架构的商用机集群上,适合一次性写入、多次读取的应用场景.为了应对节点故障,HDFS 引入多个数据备份和容错机制,保证存储系统的高可用性.一些大数据应用,例如淘宝、Facebook、微信等,需要管理海量的小文件(如图片、用户上传文件等),这类应用如果使用 HDFS 的大数据存储系统进行数据管理会产生大量的块内空间浪费,同时产生大量文件到存储节点映射等元数据,负责管理元数据的 Namenode 会成为文件系统存取的性能瓶颈.淘宝的 TFS[14]就是为了管理海量小文件应运而生的分布式文件系统.其他常见的分布式文件系统还包括 Ceph[15]、Amazon S3[16,17]、FastDFS[18]、GridFS[19]、MogileFS[20],这些系统可作为 HDFS 的重要补充.

HDFS 是基于磁盘的分布式文件系统,数据的访问需要频繁的 I/O 调用,这会对系统性能造成影响.大数据系统强调存储和计算分离,不同的计算系统可以使用同一份存储在底层 HDFS 中的数据,同一计算系统也可使用不同的分布式文件系统.通过引入分布式缓冲区管理系统,可以屏蔽底层的分布式文件系统,将来自不同文件系统的热点数据尽可能地维护在缓冲区中,减少上层计算访问数据带来的 I/O 开销.典型的分布式缓冲区管理系统包括 Alluxio[21,22]、Redis[23]、Memcache[24].例如,Alluxio 是开源的分布式内存文件系统,提供了与基于磁盘文件系统相同的访问接口,并屏蔽了底层不同的文件系统.通过引入分层存储特性,在高速内存与低速磁盘之间部署高性能 SSD 存储设备,构建磁盘、SSD、内存三级数据存储架构,并结合数据的新鲜度和访问热度,优化数

据块在内存、SSD、磁盘上的存储策略,使得频繁使用的数据优先缓存在高性能存储介质中,从而减少磁盘访问带来的开销.

2.2 面向不同数据模型的NoSQL系统

数据模型是数据管理系统的核心.大数据应用可以是结构化的关系数据、图数据,也可以是半结构化的XML 或 JSON 数据,还可以是非结构化的多媒体、网页等数据.遵循 one-size-does-not-fit-all 的理念,NoSQL 数据库基于键值对、文档、图等不同数据模型,为管理不同类型的数据提供了有效的数据存储服务.从历史发展的角度看,Google BigTable[3]是较早提出的键值对模型,该模型使得对多列历史数据的有序存取较为高效.数据进行层次范围划分后分布到多台分片服务器上,采用严格一致性进行数据更新.Amazon Dynamo[25]采用另一种不同的键值对存储方法,将键映射到某个特定的值,采用最终一致性方法进行数据更新.另一些流行的 NoSQL 系统部分借鉴了这几个系统的特征.如 HBase 的设计与 BigTable 非常类似,Voldemort[26]则复制了 Dynamo 的很多特征,Cassandra[27]则在数据模型方面借鉴了 BigTable,而在数据划分和一致性方面借鉴了 Dynamo.由于键值对模型概念简单且具有较高的存取效率和可扩展性,还有很多其他的系统,如 Redis、RAMCloud[28]等,均以键值对为基础进行模型的设计和实现.文档是另一种数据类型,文档数据库主要用于存储、检索和管理面向文档的信息.在 NoSQL 框架内,文档可以看作是键值存储的一个子类.不同之处在于数据处理的方式:在键值存储中,数据对数据库本身是不透明的,而面向文档的系统则依赖于文档中的内部结构.文档数据库与传统的关系数据库也有很大不同.关系数据库将数据存储在表中,单个对象可能分布在多个表中,而文档数据库则将给定对象的所有信息存储在数据库的单个对象中,并且每个数据库对象内部结构可以各不相同.该方式简化了外部对象到数据库对象的映射,一定程度上方便了 Web 应用的开发和部署.图数据管理技术起源于 20 世纪 70 年代,在这一阶段,关系数据库由于具备严格的关系理论辅助数据建模,操作接口简单,数据独立性高,查询优化技术实现突破,逐渐成为数据管理中的主流技术;相反地,图数据管理技术在数据建模、查询表达等方面复杂度高,这一阶段的图数据管理技术发展缓慢.进入 21 世纪,随着语义网技术的发展以及社交网络等真实大图数据的迅猛增长,应用需求的驱动,使得图数据管理的相关研究工作重新成为热点.值得探讨的是:与成熟的关系数据管理技术相比,图数据管理仍然缺乏统一的数据模型和查询语言,目前主流的图数据库包括 Neo4J[29]、OrientDB[30]、微软的 Azure Cosmos DB[31,32]等,图模型的常用数据结构为标签图(例如语义网中 RDF、部分知识图谱)和属性图,图的基本操作包括图匹配、图导航、图与关系的复合操作等,常用的查询语言为 SPARQL[33]、Cypher[34]和 Gremlin[35]等,这些语言的语法都与关系结构化查询语言 SQL 相近.基于其他数据模型的开源系统可参见文献[36].

2.3 计算系统

大数据计算系统可以采用不同的计算模型,常见的计算模型包括批计算、流计算、迭代计算、交互式计算等.每一类计算系统可以抽象出基本访问接口,例如:批计算系统 MapReduce 提供了 Map 和 Reduce 接口,流计算系统 Storm[37]提供了 Spout 和 Bolt 接口以分别接收和处理数据,迭代计算系统 Giraph 提供了面向图中顶点计算的 compute 接口,交互式计算系统 Presto 提供了 SQL 的访问接口.开发者只需实现相应的接口函数,就可以调用平台的分布式、可扩展、容错的计算能力,完成复杂的分析、查询任务.批计算面向批量、静态数据集上的计算,特别适合海量数据的计算,其对查询的响应延时没有过高的要求.典型的批处理系统包括 Hadoop、Spark 等.流计算系统面向的是实时流入的数据,并对每个单独数据项流入系统时作出实时的处理,流计算对查询响应的实时性要求高.典型的流计算系统有 Spark Streaming[38]、Storm、Flink[39]、Yahoo 的 S4[40]、阿里的 JStorm[41]和 Blink[42]、Facebook 的 Puma[43]、IBM 的 StreamBase[44].迭代计算面向的是需要多轮计算的应用场景,其中,上一轮计算的输出可作为下一轮计算的输入,直至满足一定条件时系统终止计算.典型的应用包括具有明显迭代特征的数据挖掘算法,例如 *K*-means、*K*-medoids、Semi-clustering 等,迭代的图计算,例如 PageRank、最短路径等.迭代计算的系统包括 GraphX[45,46]、Giraph[47]、GraphLab[48]、Haloop[49]等.交互式计算类似于传统关系数据库,为了达到实时的交互式响应,很多交互式系统会把数据完全维护在内存中,如 Presto[50],典型的交互式计算

系统包括 Impala[51]、Presto、Drill[52]等.

2.4 查询处理引擎

大数据的查询处理引擎基于大数据计算系统,通过计算系统提供的通用接口,借助分布式查询优化技术,实现数据的高性能查询与分析.大数据的查询处理引擎为用户和开发者提供类 SQL(有些甚至可以兼容 SQL)的查询语言,通过语法解析器,把查询语句转换成对计算层的作业调度,最后由计算层把结果返回给用户.根据调用计算系统的不同,这些引擎可分为:(1) 基于 MapReduce 的查询分析引擎;(2) 基于内存式批计算系统的查询分析引擎;(3) 基于 MPP 的查询分析引擎.早期的大数据查询分析引擎基于 MapReduce,又称为 SQL-on-Hadoop. Hive[53,54]是基于 Hadoop 的一个数据仓库工具,提供类 SQL 的查询语言 HiveQL,通过把用户提交的 HiveQL 语句转化为 MapReduce 作业并提交到 Hadoop 集群中运行.MapReduce 作业串行执行,Hadoop 监控作业执行过程,所有作业完成后返回结果给用户.其中,分布式查询优化体现在从 HiveQL 到转化为 MapReduce 的作业以及 MapRedue 的多作业调度.Hive 适合面向大数据集的批处理作业,例如搜索引擎的日志分析.因为 Hive 基于高延时的 MapReduce 批计算模型,所以不适合那些需要低延迟的应用.Spark SQL[55]是基于内存的批计算系统 Spark 的查询引擎,其原理与 Hive 类似.Presto 是基于 MPP 的查询分析引擎,具有较低的响应延时,可用作交互式查询引擎.与 Hive 把 HiveQL 转化成多个 MapRedue 作业不同,Presto 使用了一个定制的操作符和查询执行引擎来支持 SQL.除了改进的调度算法之外,所有的数据处理都是在内存中进行的,操作之间采用流水线处理方式,避免了不必要的磁盘读写和额外的延迟.

2.5 小 结

进入大数据时代,大数据从业者已不像数据库时代那样追求使用关系数据库管理系统解决所有数据管理的问题,而是探索从数据存储、数据组织、通用计算、查询处理等维度对数据管理系统进行解藕,解藕后的系统各模块彼此相对独立而又各自发展.根据大数据应用的实际需要,各模块可通过采用松耦合的方式进行组装,构建完整的大数据管理系统.

3 大数据管理系统的特征

认识新生事物可行的办法就是“瞎子摸象”,也就是试图从多个不同的视角去刻画,综合起来就可以获得立体丰满的图景了.为此,本节将从数据特征、系统特征以及应用特征这 3 个角度认识大数据管理系统.数据特征描述系统管理对象的特点,系统特征描述系统应具备的功能,而应用特征描述如何在大数据管理系统中进行应用的开发.

3.1 大数据管理系统的数据特征

大数据的数据特征可以用 4V 来刻画,就是大容量(volume)、多类型(variety)、快变化(volocity)和低质量(veracity).这既是对大数据特征的刻画,也是对大数据管理系统提出的新的要求.

(1) 大容量.多大的数据量可以称为“大”?这没有一个确定的标准,是与当时的技术水平和应用水平相关的,因此,大容量的挑战是“与时俱进”的.1975 年,著名的超大规模数据库会议(VLDB)召开第 1 届年会的时候,面临的挑战是管理 100 万条记录的商业数据.这在今天看来是很小的数据集了.在 21 世纪初,所谓数据密集型应用,数据量大约在 1T 左右,而今天所说的大数据,容量基本上在数百 TB 甚至 PB 级别,才会对现有的数据库技术产生真正意义上的挑战.1998 年,图灵奖获得者 Jim Gray 曾提出数据量的增长符合“摩尔定律”,也就是说,每 18 个月,新增的存储量等于有史以来存储量之和.以 BAT 为例,百度的数据量已经超过 1 000P,无疑是互联网大数据的执牛耳者;

(2) 多类型.多类型是大数据显著的特点,关系数据库只能处理关系型数据,这是它的主要限制.现实世界中形形色色的应用并不能保证只有关系型数据,事实上,大数据就是要汇聚多个来源的数据,因此,数据种类既有结构化数据,也有各种非结构化数据.如何在一个系统平台中处理多种类型的数据,是大数据的核心挑战之一;

(3) 变化快.变化快的特征指的是产生、收集、分析大量数据的速度快,变化快的特点要求系统在数据产生的过程中分析数据,而不必将数据预先存入数据库中.关于变化快这一点,我们感受得可能不够深刻.举一个例子,“双十一”是淘宝搞的网上促销活动,据公开数据显示,2017 年,支付峰值在 11 日凌晨 5 分钟 22 秒时为 25.6 万笔/秒,是 2016 年的 2.1 倍,这些交易给底层数据库处理带来的峰值是 4 200 万次/秒.这还不是最高的要求,据说,某些网络监控系统需要实现的数据入库的要求超过 100GB 条记录/秒.这就要求系统具有很高的吞吐量才行;

(4) 低质量.这是指大数据通常都是自动采集的,天然地具有噪音,如何在有噪音的情况下还能被有效地运用?这不是传统的查询操作所能做到的,需要发展更复杂的数据治理、数据分析和机器学习技术.

3.2 大数据管理系统的系统特征

目前,大数据管理技术还在快速进化之中,还远没有成型.管理好 4V 的数据,从大数据管理的功能定位出发,我们可以归纳出如下 5 个大数据管理的系统特征.

- 第一,大数据管理系统是一个开放的系统.大数据环境下,数据类型是开放的,系统面向的不能仅仅是事先定义好的数据类型,也需要支持各种非结构化的数据类型.其次,数据操作是开放的,系统面向的不能仅仅是确定的操作算子,需要支持用户定义的操作的实施.因此,大数据管理系统不能仅仅是单一的处理引擎,需要有一个框架可以容纳和支持多个不同类型数据处理引擎并存.需要说明的是:传统的关系数据库遵循封闭世界假设,即不在数据表中的记录均为假.基于此假设,关系数据库的查询优化技术,包括查询等价变换、查询包含、物化视图更新等操作,也均基于封闭世界假设.而大数据的开放性特征使得传统关系数据库的封闭世界假设不再适用.在此背景下,大数据系统中的查询表达和查询优化将是一个难点,特别是面对用户定义的操作以及跨引擎的查询等如何进行优化,是一个巨大的挑战;
- 第二,大数据管理系统是一个多模型并存的系统.大数据的多类型特征,使得单一的关系模型无法为多样化的大数据进行有效建模.在关系模型中,关系模式要求至少满足第一范式.为了减少数据冗余和数据异常,往往要求关系模式满足尽可能高的范式要求;在管理数据时,需要事先定义关系模式,由于模式的修改代价非常昂贵,往往在进行数据库设计时,要求数据库模式(关系模式等的集合)相对稳定.上述这几个特性在大数据应用背景下均无法得到有效满足.首先,大数据的应用主要面向分析型,数据以追加操作为主,数据冗余和数据异常不是主要矛盾.事实上,在实际的大数据应用中,性能往往是需要优先考虑的因素.为了提高分析性能,在数据建模时,往往通过引入数据冗余,降低关系模式满足的范式要求(甚至都可以不满足第一范式),减少甚至避免昂贵的连接操作.这种做法在宽表、面向社交网络数据管理的文档模型中得到了广泛应用.从这一点上看,大数据应用的数据建模与传统的关系数据库设计理念是相违背的.其次,大数据应用中,特别是面向非结构化数据管理的互联网应用,关系模式往往无法事先就完整地定义下来,并且随着应用的深入开展,关系模式也是不断变化的,这与关系数据库建模中要求数据库模式相对稳定的假设是相违背的.在同一个系统中,支持多数据模型混合并存并提供一种通用的数据建模方法,目前还没有一个成熟的技术方案;
- 第三,大数据管理系统将是一个强调高可用性和分布式可扩展性的系统.大数据应用的多样性特征弱化了传统关系数据库中事务的 ACID 特性,强调数据的高可用和分布式可扩展,容许数据有最终一致性甚至弱一致性.传统的数据库面向的是涉及到人、财、物等需要精细管理的应用,特别是面向转账、记账、订票等核心业务的应用,事务在其中起着决定性的作用.而大数据应用的驱动,包括域名服务(DNS)、搜索、电子邮件等,已经不同于传统的涉及到人、财、物等需要精细管理的应用,它不要求严格的数据一致性,甚至可以容许数据的部分不一致,强调的是服务的高可用性和分布式可扩展性;
- 第四,大数据管理系统将是一个量质融合的系统,不仅需要管理大容量的数据,还要管理带噪音的数据.传统的关系数据库通过数据完整性约束的检查与维护机制,在破坏数据库完整性的操作发生时,通过拒绝等行为保护数据库不受侵害.因此,传统关系数据库可以认为不关心数据质量问题.但是对于大数据管理系统,缺失数据、矛盾数据、不完整数据的存在是常态,因此,对数据的查询要用对结果的排序

来取代,对数据的统计建模要用机器学习模型来取代;

- 第五,大数据管理系统的中心将是知识管理.大数据的价值在于知识,如何支持从大数据中发现知识,是大数据管理系统不可或缺的功能.有两种基本的方法:一种是知识图谱,还有一种是深度学习.这都是目前大数据知识发现中最重要的方法.

大数据管理系统还没有像关系数据库那样成型,我国在大数据与云计算重点研发计划中,已经设立了面向领域的大数据管理系统的两个项目:一个是支持工业大数据领域,一个是支持科学大数据领域.目标是结合领域应用的特点,探索大数据管理系统的基础架构、核心功能和示范应用.

3.3 大数据管理系统的应用特征

大数据管理系统的应用到底长什么样?如何进行开发?目前也许还看不清楚.但是,我们认为以下几点值得关注.

(1) 以对象为中心进行数据组织,实现数据汇聚.不同于传统的信息系统,传统的信息系统是以业务为中心的,数据需求来自于业务系统,数据组织是为了业务处理的高效率;大数据则不同,大数据管理系统是以服务对象为核心,汇聚来自不同的业务系统的数据,包括不同种类、不同频度的数据等,构成大数据以进行处理;

(2) 以第四范式为解决问题的新模式.第四范式的另一种叫法是基于数据的科学发现方法,是相对于其他3种范式而言的,即实验观察、理论推导和计算机模拟.实验观察是最古老的解决问题的方法,例如,通过实验获得经验,总结规律,形成新知识;理论推导是以数学为工具的方法;计算机模拟可以用于研究核爆炸、天气演化等.不同于上述3种模式,大数据方法是通过积累大数据,并开发大数据上的数据挖掘方法来发掘规律;

(3) 以机器学习为主要应用类型.也许,一种称为OLML(online machine learning)的应用将大量出现,即在同一个大数据集上,多个用户都在训练自己的机器学习模型,并进行预测.如果我们将机器模型训练看成是大数据不同数据子集上的计算的话,一个OLML就类似于一次SQL查询处理.

3.4 小 结

作为大数据管理的核心,大数据管理系统是一套面向大容量、多类型、快变化、低质量数据管理的系统软件,该系统以量质融合的知识管理为中心,支持结构化、半结构化、非结构化等多类型数据的组织、存储和管理,具备高可用和分布式可扩展的系统架构特征和不同级别的事务特征,提供类SQL的数据查询语言定义、操纵、控制、可视化数据,支持快速的应用开发和系统运维.AsterixDB[56]是由学术界和工业界联合开发的、探索面向大数据管理的一套开源系统.与NoSQL数据库相比,AsterixDB提供了一套较为完整的类SQL语言来定义和操纵数据.但该系统目前只支持单记录的事务,无法提供不同级别的事务一致性;系统的设计也不是面向知识管理的,因此,该系统距理想中的大数据管理系统还有很长的一段路要走.

4 大数据管理系统未来发展展望

结合大数据管理系统正在经历以软件为中心到以数据为中心的计算平台的变迁以及软件基础设施化的大数据时代特征,本节从数据模型、计算模式、系统架构、新硬件、自适应优化这5个方面展望大数据管理系统的未来发展.

4.1 多数据模型并存

大数据应用的鲜明特征之一就是数据的多样性,既有结构化的关系数据、图数据、轨迹数据,也有非结构化的文本数据、图片数据,甚至是视频数据等.淘宝的“双十一”就是这类典型的大数据应用.大数据管理系统的一个基本要求就是能够支持结构化、半结构化、非结构化等多种数据类型的组织、存储和管理,形成以量质相融合的知识管理为中心、并以此提供面向知识服务的快速应用开发接口.纵观现有的大数据系统,特别是以NoSQL数据库为主的大数据系统,走的仍然是一条一种数据模型解决一类数据的传统道路.虽然也符合

one-size-fits-a- bunch 的设计理念,但应用的要求仍然希望这里的“bunch”尽可能地接近“all”.具体来说,图数据库支撑的是类似于社交网络、知识图谱、语义网等强关联数据的管理;关系数据库支撑的是人、财、物等需要精细数据管理的应用;键值对数据库适合非结构化或宽表这类无需定义数据模式或模式高度变化的数据管理.在新型大数据应用背景下,把多种类型的数据用同一个大数据管理系统组织、存储和管理起来,并提供统一的访问接口,这是大数据管理系统的一条必经之路.多数据模型并存下的数据管理会存在很多的技术挑战,具体包括:

(1) 数据如何建模?关系数据库具有严格的关系数据理论,并从降低数据冗余度和数据异常两个维度辅助数据建模.而在新的数据模型下,甚至是多数据模型下,如何进行数据建模是一个值得探索的课题;

(2) 数据的访问提供统一的用户接口.多模型之间的数据如何交互和协同以及提供与存储层和计算层的统一交互接口;

(3) 对多数据模型混合的数据处理提供执行优化,通过统一的资源管理优化任务调度,通过性能预估优化计算和通信等.

4.2 多计算模式互相融合

未来的大数据管理系统具有多计算模式并存的特点.目前,Hadoop、Spark[57]及 Flink 等主流大数据系统具有不同的计算模式,系统通常会偏重于批任务模式或流任务模式中的一种,这些系统提供的用户接口也不统一.然而在实际应用中,经常存在同时需要批任务、流任务处理的需求,例如淘宝的“双十一”就是批流融合的典型应用.因此,未来的大数据管理系统需要对批、流计算模式进行统一设计,实现统一的能够进行批流融合的计算引擎.同时,需要设计能够屏蔽底层不同计算模式差异的用户接口,方便使用.

机器学习是大数据管理中另一类重要的计算模式.目前,学术界、工业界广泛使用 TensorFlow[58]、Spark MLlib[59]、Caffe[60]等系统处理相应机器学习任务.TensorFlow 能够以数据流图作为表示形式,在参数服务器开发、执行机器学习任务;Spark MLib 基于 MapReduce 模型接口完成对大量数据的训练.这些系统仅关注机器学习中的算法训练,而实际应用中存在多种计算模式混合的情况,且参数模型可达百亿维度,现有系统均无法解决.因此,能够兼容高维机器学习计算模式,也是未来大数据管理系统的重要内容.

大数据管理系统也应兼容交互式计算模式,满足日益增长的对交互式大数据分析应用的需求.现今,Hive 等主流分析工具在易用性方面有较大的提升空间,目前主要由专业人员使用,普通分析人员较难掌握.同时,这些交互式系统在与操作人员交互的过程中还存在操作延迟长等问题,更高效的智能交互计算模式也是未来大数据管理系统需要考虑的方向之一.

总之,大数据存在对批计算、流计算、机器学习、交互式计算等多种计算模式的需求.同时,数据存储量大,无法对任一计算模式均保留一份数据,未来的大数据系统需要在同样存储数据的基础上支持多种计算模式.目前主流的大数据系统均基于开源软件,各层开源软件可相互兼容.未来的大数据管理系统需要兼容这些主流的大数据系统,同时,将存储、通用计算、专用计算分层,明确各层的接口,并在各层设计、实现兼容多种计算模式,降低系统耦合性.

4.3 可伸缩调整

在软件基础设施化的大数据时代特征背景下,未来的大数据管理系统应以云计算为平台,具有更好的分布可扩展、可伸缩调整特点,能够实现跨域的无缝融合.未来的大数据管理系统通过高速网络将不同的硬件资源连接构成一个计算系统整体,互相配合,为终端用户服务.云平台上可以运行多类应用,不同的应用需要不同的服务资源,因此系统配有多种存储与数据组织模块,可满足不同上层任务负载和计算模式的需求.系统面向多类终端用户,用户可以根据需求选择、配置合适的存储架构和数据组织方式,针对特定应用,选择、组装对应的功能模块,并可根据任务负载的强弱实时调整系统的规模和负载的分配策略.同时,针对不同用户的需求,对应用进行深入理解,提取特征进行模型构建,实现弹性可伸缩调整是未来大数据管理系统的核心技术之一.

目前的大数据管理系统通常使用分布式文件系统(例如 HDFS 和 Ceph)或者直通式键值系统管理数据的存

储,并在此基础上对键值、文档、图等进行组织,构成 NoSQL 系统,为用户提供服务.NoSQL 系统提供了更灵活的数据模型,但相对于传统 SQL 技术不具有强一致性,且通常只用于执行简单的分析任务.而未来的大数据管理系统应具有 NewSQL[61]特性,可实现传统 SQL 和 NoSQL 间的平衡,具体包括:(1) 具有传统关系数据模型和传统数据库的事务 ACID 一致性,用户可以使用 SQL 语句对系统进行操作;(2) 具有 NoSQL 可扩展等灵活特性,能够利用高速网络和内存计算,实现对海量数据的存储管理和分析等功能,系统可伸缩调整.

4.4 新硬件驱动

大数据管理系统由硬件和软件两方面构成,软件技术可受益于硬件技术发展,同时也受硬件技术体系结构特征和局限性的约束.通过对不同硬件设计合适的数据结构和算法可提升硬件效率.目前,硬件体系结构正在经历巨大变革,在向专用硬件的方向发展.同时,各类新型存储、高速互联设备的出现也在改变以往大数据管理系统中的设计与底层支持.

近些年,以 GPU 为代表的加速器件得到了迅猛发展[62],也有越来越多的大数据系统使用 GPU、Xeon Phi[63]、FPGA[64]等新硬件加速大数据管理任务.相对于传统管理系统,新硬件驱动的大数据管理系统可提供更快的负载处理速度和更好的实时可视化及处理效果.虽然新硬件驱动为大数据管理系统提供了新思路,但也带来了一系列需要解决的挑战.

(1) 新硬件分配任务.不同种类的加速设备具有完全不同的体系结构特征,它们适合处理的任务特征不同,因此在未来的大数据管理系统中,需要尽可能地使各加速设备处理合适的负载;

(2) 数据传输.由于各设备可能独立接入系统,处理负载时需从主存复制数据到设备.因此,在进行任务分配时,应充分考虑数据传输时间;

(3) 新硬件下的数据结构和算法.传统系统中适合 x86 架构处理器的数据结构可能不适合 GPU、FPGA 等新硬件,需要考虑新硬件的执行特点有针对性地设计新的数据结构和算法.

在存储和数据传输方面,新硬件也可发挥新的作用.以非易失存储器(non-volatile memory)[65]为代表的新介质可进一步加速数据处理过程,如,在故障恢复时减少恢复时间等.在大数据管理系统的存储层级,有可能会有多种存储类型,如何设计合适的数据存储也是新硬件驱动下系统设计重要的考虑因素.在分布式系统中,网络传输可能是性能瓶颈,更快速的数据传输速度和新的网络技术,如 RDMA[66]、Infiniband[67]等,可以缓解以往分布式系统中的数据传输瓶颈,如何利用这些新技术也是未来大数据管理系统设计的重要内容.

4.5 自适应调优

目前,大数据管理系统通常采用分布式文件系统和直通式键值存储等开源存储系统,并在这些开源系统的基础上构建以键值对、文档等为主要数据组织的 NoSQL 系统.虽然目前的系统能够为大数据提供存储服务,且能够进行系统扩展,但系统功能相对单一,面向复杂的计算模型和负载任务通常显得自适应能力不足,缺少必要的可伸缩调整.开发新型的能够自适应多种计算模型和任务的可伸缩调优技术,是未来大数据管理系统的发展方向.未来大数据管理系统的存储需要支持具有不同访问特征的计算模型和任务,如何针对不同模型自适应地调整内部模块、选择合适的存储以及如何对于不同任务按需分配不同的存储资源进行自适应的弹性调优(例如,通过分析系统日志来优化软件系统配置的方法[68]),是未来大数据管理系统在数据存储方面需要重点考虑的内容.

未来的大数据管理系统应能够基于不同的存储介质和存储架构有效地对数据进行组织,并根据上层计算模型的访问模式自适应地选择合适的模块,同时能够做到根据不同任务需求分配资源.具体可包括:(1) 支持多种类型存储,如具有高并发、低延迟的直通式键值存储和分布式存储等;(2) 支持主流数据模型,能够对数据进行高效组织,如对关系模型和图模型的数据提供统一访问接口,同时采用合适的数据划分策略,通过预估减少系统在存储层和计算层间的数据传输量;(3) 支持多种计算模式和混合任务的自适应调优,通过对不同存储类型和数据类型进行组织,对混合计算模型和任务构建性能模型,自动选择合适的存储模块并进行调优;(4) 支持大数据存储的可伸缩调整和容错,能够根据数据和任务类型提升不同类型存储的效率,并能面向不同任务准确地分

配合适的资源.

5 总 结

本文回顾了数据管理的发展历史,指出了大数据的开放性特征使得传统关系数据库中的三大核心技术——关系模型、查询优化、事务处理技术均无法满足大数据的管理要求.大数据时代背景下,数据管理技术的设计理念从原来关系数据库的 one-size-fits-all 转变到 one-size-does-not-fit-all,人们开始尝试研发适合自己应用的大数据系统.现阶段,主流的大数据管理系统生态具有明显的分层结构,各类子系统独立发展,通过采用松耦合的方式进行组装,构建为完整的大数据管理系统.通过对大数据管理系统应具备的数据特征、系统特征和应用特征的分析,指出大数据管理系统技术还在快速进化之中,并给出了大数据管理系统的概念说明.最后,本文从数据模型、计算模式、系统架构、新硬件、自适应优化这 5 个方面对大数据管理系统的未来发展趋势作了预测.

References:

[1] Kshetri N. Big data's role in expanding access to financial services in China. Int'l Journal of Information Management, 2016,36(3): 297−308.

[2] Yazdanifard R, Li MTH. The review of Alibaba's online business marketing strategies which navigate them to present success. Global Journal of Management and Business Research: EMarketing, 2014,14(7):33−39.

[3] Lien CH, Cao Y. Examining WeChat users' motivations, trust, attitudes, and positive word-of-mouth: Evidence from China. Computers in Human Behavior, 2014,41:104−111.

[4] Codd EF. A relational model of data for large shared data banks. Communications of the ACM, 1970,13(6):377−387.

[5] Gray J, Reuter A. Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishers, 1993.

[6] Han J, Pei J, Kamber M. Data Mining: Concepts and Techniques. 3rd ed., Morgan Kaufmann Publishers, 2002.

[7] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. In: Proc. of the 6th Symp. on Operating System Design and Implementation. 2004. 137−150.

[8] 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: Proc. of the 7th Symp. on Operating Systems Design and Implementation. 2006. 205−218.

[9] Ghemawat S, Gobioff H, Leung ST. The Google file system. In: Proc. of the 19th ACM Symp. on Operating Systems Principles. 2003. 29−43.

[10] White T. Hadoop: The Definitive Guide. 4th ed., O'Reilly Media, Inc., 2012.

[11] George L. HBase: The Definitive Guide: Random Access to Your Planet-Size Data. O'Reilly Media, Inc., 2011.

[12] Leavitt N. Will NoSQL databases live up to their promise? IEEE Computer, 2010,43(2):12−14.

[13] Zhou AY. Understanding on the big data: Beyond the data management and analytics. Big Data Research, 2017,3(2):3−18 (in Chinese with English abstract).

[14] Taobao file system. 2018. http://tfs.taobao.org/

[15] Weil SA, Brandt SA, Miller EL, Long DDE, Maltzahn C. Ceph: A scalable, high-performance distributed file system. In: Proc. of the 7th Symp. on Operating Systems Design and Implementation. 2006. 307−320.

[16] Amazon S3. 2018. https://aws.amazon.com/s3/

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

[18] Liu XY, Yu Q, Liao JW. FastDFS: A high performance distributed file system. ICIC Express Letters, Part B, Applications: An Int'l Journal of Research and Surveys, 2014,5(6):1741−1746.

[19] Santos MND, Cerqueira R. GridFS: Targeting data sharing in grid environments. In: Proc. of the 6th IEEE Int'l Symp. on Cluster Computing and the Grid. 2006. 17.

[20] MogileFS. 2018. https://github.com/mogilefs

[21] Li H, Ghodsi A, Zaharia M, *et al*. Tachyon: Reliable, memory speed storage for cluster computing frameworks. In: Proc. of the ACM Symp. on Cloud Computing (SOCC 2014). ACM Press, 2014. 1−15.

[22] Alluxio. 2018. https://www.alluxio.org/

[23] Redis. 2018. https://redis.io/

[24] Memcache. 2018. http://pecl.php.net/package/memcache/

[25] DeCandia 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. In: Proc. of the 21st ACM SIGOPS Symp. on Operating Systems Principles. 2007. 205−220.

[26] Voldemort. 2018. http://www.project-voldemort.com/voldemort/

[27] Apache Cassandra. 2018. https://cassandra.apache.org/

[28] Ousterhout J, Agrawal P, Erickson D, Kozyrakis C, Leverich J, Mazières D, Mitra S, Narayanan A, Parulkar G, Rosenblum M, Rumble SM, Stratmann E, Stutsman R. The case for RAMClouds: Scalable high-performance storage entirely in DRAM. ACM SIGOPS Operating Systems Review, 2010,43(4):92−105.

[29] Neo4j. 2018. https://neo4j.com/

[30] OrientDB. 2018. https://orientdb.com/

[31] Microsoft Azure. 2018. https://azure.microsoft.com/zh-cn/services/cosmos-db/

[32] Paz JRG. Introduction to Azure Cosmos DB. Microsoft Azure Cosmos DB Revealed, Berkeley: Apress, 2018. 1−23.

[33] Arenas M, Pérez J. Querying semantic Web data with SPARQL. In: Proc. of the 30th ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems. 2011. 305−316.

[34] Cypher. 2018. http://www.neo4j.org/learn/cypher/

[35] Gremlin. 2018. https://github.com/tinkerpop/gremlin/wiki

[36] Graph DBMS. 2018. https://db-engines.com/en/ranking/graph+dbms/

[37] Apache storm. 2018. http://storm.apache.org/

[38] Apache spark streaming. 2018. http://spark.apache.org/streaming/

[39] Apache flink. 2018. http://flink.apache.org/

[40] Neumeyer L, Robbins B, Nair A, Kesari A. S4: Distributed stream computing platform. In: Proc. of the 2010 IEEE Int'l Conf. on Data Mining Workshops. 2010. 170−177.

[41] JStorm. 2018. http://www.jstorm.io/

[42] Blink. 2018. https://flink.apache.org/poweredby.html/

[43] Chen GJ, Wiener JL, Iyer S, Jaiswal A, Lei L, Simha N, Wang W, Wilfong K, Williamson T, Yilmaz S. Realtime data processing at Facebook. In: Proc. of the 2016 ACM Int'l Conf. on Management of Data. 2016. 1087−1098.

[44] StreamBase. 2018. https://www.tibco.com/products/tibco-streambase

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

[46] Xin RS, Gonzalez JE, Franklin MJ, Stoica I. GraphX: A resilient distributed graph system on spark. In: Proc. of the 1st Int'l Workshop on Graph Data Management Experiences and Systems. 2013. 2:1−2:6.

[47] Apache giraph. 2018. http://giraph.apache.org/

[48] Low YC, Bickson D, Gonzalez J, Guestrin C, Kyrola A, Hellerstein JM. Distributed GraphLab: A framework for machine learning and data mining in the cloud. Proc. of the VLDB Endowment, 2012,5(8):716−727.

[49] 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.

[50] Presto. 2018. http://prestodb.io/

[51] Bittorf M, Bobrovytsky T, Erickson C, *et al*. Impala: A modern, open-source SQL engine for Hadoop. In: Proc. of the 7th Biennial Conf. on Innovative Data Systems Research. 2015.

[52] Apache drill. 2018. http://drill.apache.org/

[53] Thusoo A, Sarma JS, Jain N, *et al*. Hive: A warehousing solution over a map-reduce framework. Proc. of the VLDB Endowment, 2009,2(2):1626−1629.

[54] Apache hive. 2018. https://hive.apache.org/

[55] Armbrust M, Xin RS, Lian C, *et al*. Spark SQL: Relational data processing in Spark. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data. 2015. 1383−1394.

[56] Asterixdb. 2018. https://asterixdb.apache.org/

[57] Zaharia M, Xin RS, Wendell P, *et*. *al*. Apache Spark: A unified engine for big data processing. Communications of the ACM, 2016, 59(11):56−65.

[58] Abadi M, Barham P, Chen J, *et al*. TensorFlow: A system for large-scale machine learning. In: Proc. of the 12th USENIX Conf. on Operating Systems Design and Implementation. 2016. 265−283.

[59] Meng X, Bradley J, Yavuz B, *et al*. MLlib: Machine learning in apache Spark. The Journal of Machine Learning Research, 2016, 17(1):1235−1241.

[60] Jia Y, Shelhamer E, Donahue J, *et al*. Caffe: Convolutional architecture for fast feature embedding. In: Proc. of the 22nd ACM Int'l Conf. on Multimedia. 2014. 675−678.

[61] Grolinger K, Higashino WA, Tiwari A, *et al*. Data management in cloud environments: NoSQL and NewSQL data stores. Journal of Cloud Computing: Advances, Systems and Applications, 2013,2(1):49:1−49:24.

[62] Rossbach CJ, Currey J, Silberstein M, Ray B, Witchel M. PTask: Operating system abstractions to manage GPUs as compute devices. In: Proc. of the 23rd ACM Symp. on Operating Systems Principles. 2011. 233−248.

[63] Jeffers J, Reinders J. Intel Xeon Phi Coprocessor High Performance Programming. Morgan Kaufmann Publishers, 2013.

[64] Indeck RS, Cytron RK, Franklin MA, *et al*. Associative database scanning and information retrieval using FPGA devices: U.S. Patent 7,139,743, 2006-11-21.

[65] Hurst TN, Perlov C, Wilson C, *et al*. Non-Volatile memory: U.S. Patent 6,646,912, 2003-11-11.

[66] Pandya AA. TCP/IP processor and engine using RDMA: U.S. Patent 7,376,755, 2008-5-20.

[67] Liu J, Wu J, Panda DK. High performance RDMA-based MPI implementation over InfiniBand. Int'l Journal of Parallel Programming, 2004,32(3):167−198.

[68] Wang JM. Key technologies in big data applications development and runtime support platform. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1516−1528 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5231.htm [doi: 10.13328/j.cnki.jos.005231]

附中文参考文献:

[13] 周傲英.感悟大数据——从数据管理和分析说起.大数据,2017,3(2):3−18.

[68] 王建民.领域大数据应用开发与运行平台技术研究.软件学报,2017,28(16):1516−1528. http://www.jos.org.cn/1000-9825/5231.htm [doi: 10.13328/j.cnki.jos.005231]

杜小勇(1963－),男,浙江开化人,博士,教授,博士生导师,CCF 会士,主要研究领域为数据管理技术,语义网技术,智能信息检索技术.

卢卫(1981－),男,博士,副教授,CCF 专业会员,主要研究领域为云计算与大数据管理,空间与文本数据管理,索引技术.

张峰(1988－),男,博士,讲师,CCF 专业会员,主要研究领域为数据库系统,高性能计算.