

架构大数据: 挑战、现状与展望

王 珊^{1), 2)} 王会举^{1), 2)} 覃雄派^{1), 2)} 周 烜^{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)

摘 要 大数据分析相比于传统的数据仓库应用, 具有数据量大、查询分析复杂等特点. 为了设计适合大数据分析的数据仓库架构, 文中列举了大数据分析平台需要具备的几个重要特性, 对当前的主流实现平台——并行数据库、MapReduce 及基于两者的混合架构进行了分析归纳, 指出了各自的优势及不足, 同时也对各个方向的研究现状及作者在大数据分析方面的努力进行了介绍, 对未来研究做了展望.

关键词 大数据; 大规模可扩展; MapReduce; 并行数据库; 深度分析

中图法分类号 TP311 DOI 号: 10. 3724/SP. J. 1016. 2011. 01741

Architecting Big Data: Challenges, Studies and Forecasts

WANG Shan^{1), 2)} WANG Hui-Ju^{1), 2)} QIN Xiong-Pai^{1), 2)} ZHOU Xuan^{1), 2)}

¹⁾(Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University of China) of Ministry of Education, Beijing 100872)
²⁾(School of Information, Renmin University of China, Beijing 100872)

Abstract Compared with traditional data warehouse applications, big data analytics are huge and complex. To design a favorable architecture for big data analytics, this paper lists some key features for big data analytics, summarizes current main implementation platforms(parallel databases, MapReduce, and hybrid architectures based on them), and points their pros and cons. Some current researches are also investigated, our work are introduced and some challenging research problems in the future are discussed.

Keywords big data; large scale; MapReduce; parallel database; deep analytics

1 引 言

最近几年, 数据仓库又成为数据管理研究的热点领域, 主要原因是当前数据仓库系统面临的需求在数据源、需提供的数据服务和所处的硬件环境等方面发生了根本性的变化(详见 1.1 节), 这些变化是我们必须面对的.

本文在大数据的时代背景下, 对现有数据仓库

系统实现方案(主要是并行数据库和 MapReduce)进行重新审视, 期望能为设计满足时代需求的数据仓库系统提供理论参考. 限于篇幅, 本文主要关注不同数据仓库实现方案的主体架构及其缺陷在最近几年的改进情况. 依据研究立足点的不同, 本文将该领域的研究归为三大类: 并行数据库、MapReduce、并行数据库和 MapReduce 技术的混合架构. 其中第三类研究又细分为: 并行数据库主导型、MapReduce 主导型、并行数据库和 MapReduce 集成型三种. 本

收稿日期: 2011-08-12; 最终修改稿收到日期: 2011-09-15. 本课题得到国家重大科技专项核高基项目(2010ZX01042-001-002)、国家自然科学基金(61070054, 61170013)、中国人民大学科学研究基金(中央高校基本科研业务费专项资金, 10XN1018)、中国人民大学研究生基金(11XNH120)资助. 王 珊, 女, 1944 年生, 教授, 博士生导师, 中国计算机学会(CCF)高级会员, 主要研究领域为高性能数据库、知识工程、数据仓库. E-mail: swang@ruc.edu.cn. 王会举, 男, 1979 年生, 博士研究生, 主要研究方向为大规模集群数据库、内存数据库. E-mail: wanghuiju@ruc.edu.cn. 覃雄派, 男, 1971 年生, 博士, 讲师, 中国计算机学会(CCF)会员, 主要研究方向为数据库查询优化、内存数据库、并行数据库. 周 烜, 男, 1979 年生, 博士, 副教授, 主要研究方向为信息检索、高性能数据库.

文第 1 节分析大数据时代, 数据仓库所面临的问题及挑战; 第 2 节列出大数据时代的数据仓库平台需要具备的几个重要特性; 第 3 节到第 5 节就这几个特性对各类平台进行归纳分析; 第 6 节对最新研究做一跟踪归纳; 第 7 节介绍中国人民大学在大数据分析方面的研究工作; 第 8 节对未来研究做出展望; 第 9 节总结全文.

1.1 三个变化

(1) 数据量. 由 TB 级升至 PB 级, 并仍在持续爆炸式增长. 根据 WinterCorp 的调查显示, 最大的数据仓库中的数据量, 每两年增加 3 倍^[1] (年均增长率为 173%), 其增长速度远超摩尔定律增长速度. 照此增长速度计算, 2015 年最大数据仓库中的数据量将逼近 100PB.

(2) 分析需求. 由常规分析转向深度分析(Deep Analytics). 数据分析日益成为企业利润必不可少的支撑点. 根据 TDWI 对大数据分析的报告^[2] (如图 1), 企业已经不满足于对现有数据的分析和监测, 而是更期望能对未来趋势有更多的分析和预测, 以增强企业竞争力. 这些分析操作包括诸如移动平均线分析、数据关联关系分析、回归分析、市场篮分析等复杂统计分析, 我们称之为深度分析. 值得补充的是, 本文中的大数据分析不仅仅指基于大数据上的深度分析, 也包括常规分析.

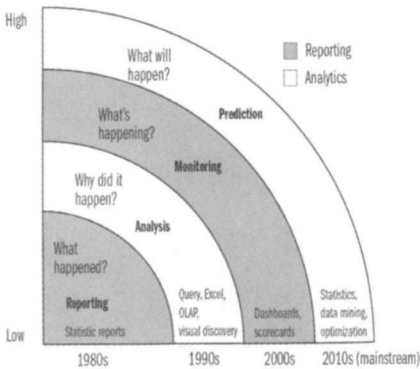


图 1 分析的趋势

(3) 硬件平台. 由高端服务器转向由中低端硬件构成的大规模机群平台. 由于数据量的迅速增加, 并行数据库的规模不得不随之增大, 从而导致其成本的急剧上升. 出于成本的考虑, 越来越多的企业将应用由高端服务器转向了由中低端硬件构成的大规模机群平台.

1.2 两个问题

图 2 是一个典型的数据仓库架构^[3]. 从图中我们可以看出, 传统的数据仓库将整个实现划分为 4

个层次, 数据源中的数据首先通过 ETL 工具被抽取到数据仓库中进行集中存储和管理, 再按照星型模型或雪花模型组织数据, 然后 OLAP 工具从数据仓库中读取数据, 生成数据立方体(MOLAP)或者直接访问数据仓库进行数据分析(ROLAP). 在大数据时代, 此种计算模式存在两个问题:

问题 1. 数据移动代价过高. 在数据源层和分析层之间引入一个存储管理层, 可以提升数据质量并针对查询进行优化, 但也付出了较大的数据迁移代价和执行时的连接代价: 数据首先通过复杂且耗时的 ETL 过程存储到数据仓库中, 在 OLAP 服务器中转化为星型模型或者雪花模型; 执行分析时, 又通过连接方式将数据从数据库中取出. 这些代价在 TB 级时也许可以接受, 但面对大数据, 其执行时间至少会增长几个数量级. 更为重要的是, 对于大量的即席分析, 这种数据移动的计算模式是不可取的.

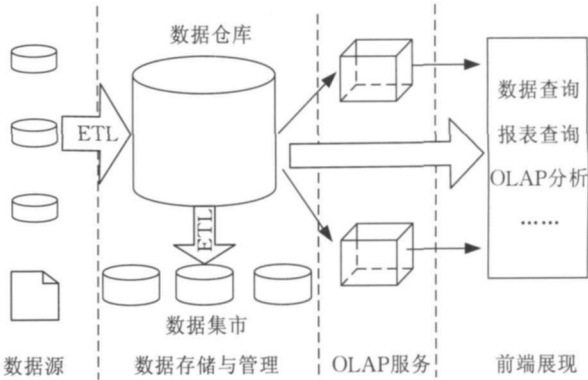


图 2 一个典型的数据仓库架构

问题 2. 不能快速适应变化. 传统的数据仓库假设主题是较少变化的, 其应对变化的方式是对数据源到前端展现的整个流程中的每个部分进行修改, 然后再重新加载数据, 甚至重新计算数据, 导致其适应变化的周期较长. 这种模式比较适合对数据质量和查询性能要求较高、而不太计较预处理代价的场合. 但在大数据时代, 分析处在变化的业务环境中, 这种模式将难以适应新的需求.

1.3 一个鸿沟

在大数据时代, 巨量数据与系统的数据处理能力之间将会产生一个鸿沟: 一边是至少 PB 级的数据量, 另一边是面向传统数据分析能力设计的数据仓库和各种 BI 工具. 如果这些系统或工具发展缓慢, 该鸿沟将会随着数据量的持续爆炸式增长而逐步拉大.

虽然, 传统数据仓库可以采用舍弃不重要数据或者建立数据集市的方式来缓解此问题, 但毕竟只

是权益之策,并非系统级解决方案.而且,舍弃的数据在未来可能会重新使用,以发掘更大的价值.

2 期望特性

本节我们列出对大数据进行分析时,数据仓库系统需具备的几个重要特性(表 1 所示).

表 1 大数据分析平台需具备的特性

特性	简要说明
高度可扩展性	横向大规模可扩展,大规模并行处理
高性能	快速响应复杂查询与分析
高度容错性	查询失败时,只需重做部分工作
支持异构环境	对硬件平台一致性要求不高,适应能力强
较低的分析延迟	业务需求变化时,能快速反应
易用且开放接口	既能方便查询,又能处理复杂分析
较低成本	较高的性价比
向下兼容性	支持传统的商务智能工具

高度可扩展性. 一个明显的事实是,数据库不能依靠一台或少数几台机器的升级(scale-up 纵向扩展)满足数据量的爆炸式增长,而是希望能方便地做到横向可扩展(scale-out)来实现此目标.

普遍认为 shared-nothing 无共享结构(每个节点拥有私有内存和磁盘,并且通过高速网络同其它节点互连)具备较好的扩展性^[4].分析型操作往往涉及大规模的并行扫描、多维聚集及星型连接操作,这些操作也比较适合在无共享结构的网络环境运行.Teradata 即采用此结构,Oracle 在其新产品 Exadata 中也采用了此结构.

高性能. 数据量的增长并没有降低对数据库性能的要求,反而有所提高.软件系统性能的提升可以降低企业对硬件的投入成本、节省计算资源,提高系统吞吐量.巨量数据的效率优化,并行是必由之路.1PB 数据在 50MB/s 速度下串行扫描一次,需要 230 天;而在 6000 块磁盘上,并行扫描 1PB 数据只需要 1 个小时.

高度容错. 大数据的容错性要求在查询执行过程中,一个参与节点失效时,不需要重做整个查询.而机群节点数的增加会带来节点失效概率的增加.在大规模机群环境下,节点的失效将不再是稀有事件(Google 报告,平均每个 MapReduce 数据处理任务就有 1.2 个工作节点失效^[5]).因此在大规模机群环境下,系统不能依赖于硬件来保证容错性,要更多地考虑软件级容错.

支持异构环境. 建设同构系统的大规模机群难度较大,原因在于计算机硬件更新较快,一次性购置

大量同构的计算机是不可取的,而且也会在未来添置异构计算资源.此外,不少企业已经积累了一些闲置的计算机资源,此种情况下,对异构环境的支持可以有效地利用这些闲置计算资源,降低硬件成本的投入.还需特别关注的是,在异构环境下,不同节点的性能是不一样的,可能出现“木桶效应”,即最慢节点的性能决定整体处理性能.因此,异构的机群需要特别关注负载均衡、任务调度等方面的设计.

较低的分析延迟. 分析延迟指的是分析前的数据准备时间.在大数据时代,分析所处的业务环境是变化的,因此也要求系统能动态地适应业务分析需求.在分析需求发生变化时,减少数据准备时间,系统能尽可能快地做出反应,快速地进行数据分析.

易用且开放的接口. SQL 的优点是简单易用,但其主要用于数据的检索查询,对于大数据上的深度分析来讲,是不够的.原因在于:(1)其提供的服务方式依赖于数据移动来实现:将数据从数据库中取出,然后传递给应用程序,该实现方式在大数据时代代价过高;(2)复杂的分析功能,如 R 或 Matlab 中的分析功能,SQL 是难以胜任的.因此,除对 SQL 的支持外,系统还应能提供开放的接口,让用户自己开发需要的功能.设计该接口时,除了关注其易用性和开放性,还需要特别注意两点隐藏的要求:(1)基于接口开发的用户自定义函数,能自动在机群上并行执行;(2)分析在数据库内进行,即分析尽可能靠近数据.

较低的成本. 在满足需求的前提下,某技术成本越低,其生命力就越强.需要指出的是成本是一个综合指标,不仅仅是硬件或软件的代价,还应包括日常运维成本(网络费用、电费、建筑等)和管理人员成本等.据报告,数据中心的主要成本不是硬件的购置成本,而是日常运维成本.因此,在设计系统时需要更多地关注此项内容.

向下兼容性. 数据仓库发展的 30 年,产生了大量面向客户业务的数据处理工具(如 Informatica、DataStage 等)、分析软件(如 SPSS、R、Matlab 等)和前端展现工具(如水晶报表)等.这些软件是一笔宝贵的财富,已被分析人员所熟悉,是大数据时代中小规模数据分析的必要补充.因此,新的数据仓库需考虑同传统商务智能工具的兼容性.由于这些系统往往提供标准驱动程序,如 ODBC、JDBC 等,这项需求的实际要求是对 SQL 的支持.

总之,以较低的成本投入、高效地进行数据分析,是大数据分析的基本目标.

3 并行数据库

并行数据库起源于 20 世纪 80 年代, 当前主流的并行数据库都同早期的 Gamma^[6] 和 Grace^[7] 等并行数据库类似. 这些数据库都支持标准 SQL, 并且实现了数据库界过去 30 年提出的许多先进技术. 其主要采用 shared-nothing 结构, 将关系表在节点间横向划分, 并且利用优化器来对执行过程进行调度和管理. 其目标是高性能和高可用性.

并行数据库的最大优势在于性能. 这主要得益于数据库界近几十年的研究成果——许多先进的技术手段及算法, 如索引、数据压缩、物化视图、结果缓冲、I/O 共享、优化的数据连接等. 但是在大数据时代, 如前言所述, 数据移动的实现方式将影响其性能.

并行数据库通过 SQL 向外提供数据访问服务, SQL 因其简单易用的特点而被广泛使用. 因此, 大多 BI 工具都支持基于标准 SQL 的数据交互方式, 使得关系数据库能较好地兼容当前多数 BI 工具. 某些数据库, 如 IBM DB2 还针对一些 BI 工具进行了优化. 但在大数据分析面前, SQL 接口面临巨大挑战. SQL 的优势源于其对底层数据访问的封装, 但封装在一定程度上影响了其开放性. 而且并行数据库提供的用户自定义函数大都是基于单数据库实例设计的, 从而不能在机群上并行执行, 也即意味着传统的实现方式不适合大数据的处理及分析. 而且, 在并行数据库中实现用户自定义函数往往需要经过复杂的系统交互, 甚至要熟悉数据库的内部结构及系统调用等, 从而难以使用.

并行数据库在扩展性、容错性、成本、对异构环境的支持等几项上有所欠缺. 这几项实际是相互影响的, 我们以其最大问题——扩展性为主线展开讨论. 并行数据库大多支持有限扩展, 一般可扩展至数百节点的规模, 尚未有数千节点规模的应用案例. 并行数据库扩展性有限主要因为如下几点: (1) 并行数据库软件级容错能力较差. 并行数据库基于高端硬件设计, 并且假设查询失败属于稀有事件. 因此当查询失败时, 一般采取重做查询的方式. 而在大规模机群环境下, 查询失败将会变为一个普通事件. 极端情况下, 并行数据有可能出现不停重做查询的局面; (2) 并行数据库对异构硬件的支持非常有限, 且对于处理较慢的节点反应敏感, 容易出现“木桶效应”. 如第 2 节中所论述的, 完全基于同构硬件搭建大规模

机群在现实中是较难实现的. 因而, 对异构硬件的支持能力影响了其扩展性; (3) 并行数据库若做到大规模可扩展, 其代价将会较高(需基于高端硬件来保证可靠性, 需购买昂贵的软件系统), 从而限制了其扩展性; (4) 根据 CAP 理论^{①[8]}, 在分布式系统中, 数据一致性 (Consistency)、可用性 (Availability)、子网可分解性 (Network Partitioning) 不可同时兼得, 选择其中任两项, 便会损害另一项. 并行数据库追求的是数据一致性和系统的可用性, 从而影响了它的扩展能力.

此外, 如 1.2 节所讨论的, 基于并行数据库实现的传统数据仓库借助于外围工具 (ETL 工具、OLAP 产品、BI 报表工具、统计分析软件等) 来完成数据的预处理和分析展现任务, 导致其数据处理及分析过程涉及大量的数据迁移和计算, 分析延迟往往较高.

4 MapReduce

MapReduce^[5] 是 2004 年由 Google 提出的面向大数据集处理的编程模型, 起初主要用作互联网数据的处理, 例如文档抓取、倒排索引的建立等. 但由于其简单而强大的数据处理接口和对大规模并行执行、容错及负载均衡等实现细节的隐藏, 该技术一经推出便迅速在机器学习、数据挖掘、数据分析等领域得到广泛应用^[9].

MapReduce 将数据处理任务抽象为一系列的 Map (映射)–Reduce (化简) 操作对. Map 主要完成数据的过滤操作, Reduce 主要完成数据的聚集操作. 输入输出数据均以 `<key, value>` 格式存储. 用户在使用该编程模型时, 只需按照自己熟悉的语言实现 Map 函数和 Reduce 函数即可, MapReduce 框架会自动对任务进行划分以做到并行执行.

下面本文将以基于 MapReduce 的开源实现 Hadoop^[10] 为主, 对其主要特性进行介绍.

MapReduce 是面向由数千台中低端计算机组成的大规模机群而设计的, 其扩展能力得益于其 shared-nothing 结构、各个节点间的松耦合性和较强的软件级容错能力: 节点可以被任意地从机群中移除, 而几乎不影响现有任务的执行. 该技术被称为 RAIN (Redundant/Reliable Array of Independent (and Inexpensive) Nodes). MapReduce 卓越的扩展能力已在工业界 (Google、Facebook、Baidu、Taobao

① 该理论目前尚存争议.

等)得到了充分验证。MapReduce 对硬件的要求较低,可以基于异构的廉价硬件来搭建机群,且免费开源,因此其构建成本低于并行数据库。但基于 MapReduce 的应用软件相对较少,许多数据分析功能需要用户自行开发,从而会导致使用成本的增加。

作为开源系统,MapReduce 具有完全的开放性:其 `<key,value>` 存储模型具有较强的表现力,可以存储任意格式的数据;Map 和 Reduce 两个基本的函数接口也给用户提供了足够的发挥空间,可以实现各种复杂的数据处理功能。但这种开放性也带来一个问题,就是将本来应由数据库管理系统完成的工作,诸如文件存储格式的设计、模式信息的记录、数据处理算法的实现等,转移给了程序员,从而导致程序员负担过重。程序员水平对系统处理性能起决定性作用。在某些情况下,写 MapReduce 程序的时间远大于写 SQL 语句的时间,部分复杂的 BI 报表分析,可能仅程序的编写和调试就要耗费几天的时间。

基于 MapReduce 平台的分析,无需复杂的数据预处理和写入数据库的过程,而是可以直接基于平面文件进行分析,并且其采用的计算模式是移动计算而非移动数据,因此可以将分析延迟最小化。

在同等硬件条件下,MapReduce 性能远低于并行数据库^[11],这是由其最初的设计定位决定的。MapReduce 的设计初衷是面向非结构化数据的处理。这些数据具有数据量大,处理复杂等特点,而且往往是一次性处理。为了获得较好的扩展能力和容错能力,MapReduce 采取了基于扫描的处理模式和对中间结果步步物化的执行策略,从而导致较高的 I/O 代价。为了减少数据预处理时间,MapReduce 没有使用模式、索引、物化视图等技术手段。其数据预处理仅是一次数据加载操作,但由此导致了一个问题——较高的元组解析代价^[12]。在 MapReduce

环境下,每个查询都是直接从文件系统中读入原始数据文件,而非传统的从数据库中读入经处理过的文件,因此其元组解析代价远高于关系数据库。对数据分析领域来说,连接是关键操作(如传统的星型查询和雪花查询均是依赖于连接来处理查询),但 MapReduce 处理连接的性能尤其不尽如人意。原因在于 MapReduce 最初是针对单数据集设计的处理模型,而连接操作往往涉及多个数据集。在利用 MapReduce 实现连接时,最直接的方式是每个任务执行一个属性上的连接操作,然后将多个 MapReduce 任务通过物化的中间结果串接起来。这种实现方式往往涉及中间结果的读写,从而导致大量的 I/O 操作和网络传输。

MapReduce 目前基本不兼容现有的 BI 工具。原因在于其初衷并不是要成为数据库系统,因此它并未提供 SQL 接口。但已有研究致力于 SQL 语句与 MapReduce 任务的转换工作(例如 Hive),进而有可能实现 MapReduce 与现存 BI 工具的兼容。

5 并行数据库和 MapReduce 的混合架构

基于以上分析,我们可以清楚地看出,基于并行数据库和 MapReduce 实现的数据仓库系统都不是大数据分析的理想方案。针对两者哪个更适合时代需求的问题,业界近年展开了激烈争论。当前基本达成如下共识:并行数据库和 MapReduce 是互补关系,应该相互学习^[13-14]。基于该观点,大量研究着手将两者结合起来,期望设计出兼具两者优点的数据分析平台。这种架构又可以分为三类:并行数据库主导型、MapReduce 主导型、MapReduce 和并行数据库集成型(表 2 对 3 种架构进行了对比分析)。

表 2 混合架构型解决方案对比分析

解决方案	着眼点	代表系统	缺陷
并行数据库主导型	利用 MapReduce 技术来增强其开放性,以实现处理能力的可扩展	Greenplum Aster Data	规模扩展性未改变
MapReduce 主导型	学习关系数据库的 SQL 接口及模式支持等,改善其易用性	Hive Pig Latin	性能问题未改变
并行数据库和 MapReduce 集成型	集成两者,使两者各自做各自擅长的工作	HadoopDB	只有少数查询可以下推至数据库层执行,各自的某些优点在集成后也丧失了
		Vertica	性能和扩展性仍不能兼得
		Teradata	规模扩展性未变

5.1 并行数据库主导型

该种方式关注于如何利用 MapReduce 来增强并行数据库的数据处理能力。代表性系统是 Greenplum

(已被 EMC 收购)和 Aster Data(已被 Teradata 收购)。Aster Data 将 SQL 和 MapReduce 进行结合,针对大数据分析提出了 SQL/MapReduce 框架^[15]。

该框架允许用户使用 C++、Java、Python 等语言编写 MapReduce 函数, 编写的函数可以作为一个子查询在 SQL 中使用, 从而同时获得 SQL 的易用性和 MapReduce 的开放性. 不仅如此, Aster Data 基于 MapReduce 实现了 30 多个统计软件包, 从而将数据分析推向数据库内进行(数据库内分析), 大大提升了数据分析的性能.

Greenplum 也在其数据库中引入了 MapReduce 处理功能^[16]. 其执行引擎可以同时处理 SQL 查询和 MapReduce 任务. 这种方式在代码级整合了 SQL 和 MapReduce: SQL 可以直接使用 MapReduce 任务的输出, 同时 MapReduce 任务也可以使用 SQL 的查询结果作为输入.

总的来说, 这些系统都集中于利用 MapReduce 来改进并行数据库的数据处理功能, 其根本性问题——可扩展能力和容错能力并未改变.

5.2 MapReduce 主导型

该方向的研究主要集中于利用关系数据库的 SQL 接口和对模式的支持等技术来改善 MapReduce 的易用性, 代表系统是 Hive^[17]、Pig Latin^[18] 等.

Hive 是 Facebook 提出的基于 Hadoop 的大型数据仓库, 其目标是简化 Hadoop 上的数据聚集、ad-hoc 查询及大数据集的分析等操作, 以减轻程序员的负担. 它借鉴关系数据库的模式管理、SQL 接口等技术, 把结构化的数据文件映射为数据库表, 提供类似于 SQL 的描述性语言 HiveQL 供程序员使用, 可自动将 HiveQL 语句解析成一优化的 MapReduce 任务执行序列. 此外, 它也支持用户自定义的 MapReduce 函数.

Pig Latin 是 Yahoo! 提出的类似于 Hive 的大数据集分析平台. 两者的区别主要在于语言接口. Hive 提供了类似 SQL 的接口, Pig Latin 提供的是一种基于操作符的数据流式的接口. 图 3 是 Pig Latin 在处理查询时的一个操作实例. 该查询的目的是找出“年龄在 18~25 周岁之间的用户(Users)最频繁访问的 5 个页面(Pages)”. 从图 3 可以看出, Pig 提供的操作接口类似于关系数据库的操作符(对应图中右侧部分中的每一行命令), 用户查询的脚本类似于逻辑查询计划(对应图中左侧部分). 因此, 也可以说 Pig 利用操作符来对 Hadoop 进行封装, Hive 利用 SQL 进行封装.

5.3 MapReduce 和并行数据库集成型

该方向的代表性研究是耶鲁大学提出的 HadoopDB^[19] (已于 2011 年商业化为 Hadapt^[20]).

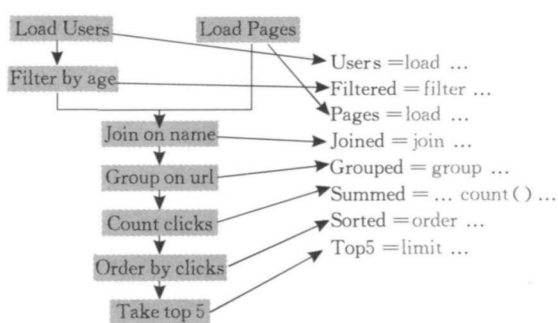


图 3 Pig Latin 的一个查询示例(右边为实际脚本)

Stonebraker 等人设计的 Vertica^[21] 数据库和 NCR 公司的 Teradata^[22] 数据库.

HadoopDB 的核心思想是利用 Hadoop 作为调度层和网络沟通层, 关系数据库作为执行引擎, 尽可能地将查询压入数据库层处理. 目标是想借助 Hadoop 框架来获得较好的容错性和对异构环境的支持; 通过将查询尽可能推入数据库中执行来获得关系数据库的性能优势. HadoopDB 的思想是深远的, 但目前尚无应用案例, 原因在于: (1) 其数据预处理代价过高: 数据需要进行两次分解和一次数据库加载操作后才能使用; (2) 将查询推向数据库层只是少数情况, 大多数情况下, 查询仍由 Hive 完成. 因为数据仓库查询往往涉及多表连接, 由于连接的复杂性, 难以做到在保持连接数据局部性的前提下将参与连接的多张表按照某种模式划分; (3) 维护代价过高. 不仅要维护 Hadoop 系统, 还要维护每个数据库节点; (4) 目前尚不支持数据的动态划分, 需要手工方式将数据一次性划分好. 总的来说, HadoopDB 在某些情况下, 可以同时实现关系数据库的高性能特性和 MapReduce 的扩展性、容错性, 但同时也丧失了关系数据库和 MapReduce 的某些优点, 比如 MapReduce 较低的预处理代价和维护代价、关系数据库的动态数据重分布等.

Vertica 采用的是共存策略: 根据 Hadoop 和 Vertica 各自的处理优势, 对数据处理任务进行划分. 比如 Hadoop 负责非结构化数据的处理, Vertica 负责结构化数据的处理; Hadoop 负责耗时的批量复杂处理, Vertica 负责高性能的交互式查询等, 从而将两者结合起来. Vertica 实际采用的是两套系统, 同时支持在 MapReduce 任务中直接访问 Vertica 数据库中的数据. 由于结构化数据仍在 Vertica 中处理, 在处理结构化大数据上的查询分析时, 仍面临扩展性问题; 如果将查询推向 Hadoop 进行, 又将面临性能问题. 因此, Vertica 的扩展性问题和 Hadoop 的性能问题在该系统中共存.

与前两者相比, Teradata 的集成相对简单. Teradata 采用了存储层的整合: MapReduce 任务可以从 Teradata 数据库中读取数据, Teradata 数据库也可以从 Hadoop 分布式文件系统上读取数据. 同样, Teradata 和 Hadoop 各自的根本性问题都未解决.

6 研究现状

对并行数据库来讲, 其最大问题在于有限的扩展能力和待改进的软件级容错能力; MapReduce 的最大问题在于性能, 尤其是连接操作的性能; 混合式架构的关键是^①, 如何能尽可能多地把工作推向合适的执行引擎(并行数据库或 MapReduce). 本节对近年来在这些问题上的研究做一分析和归纳.

6.1 并行数据库扩展性和容错性研究

华盛顿大学在文献[23]中提出了可以生成具备容错能力的并行执行计划优化器. 该优化器可以依靠输入的并行执行计划、各个操作符的容错策略及查询失败的期望值等, 输出一个具备容错能力的并行执行计划. 在该计划中, 每个操作符都可以采取不同的容错策略, 在失败时仅重新执行其子操作符(在某节点上运行的操作符)的任务来避免整个查询的重新执行.

MIT 于 2010 年设计的 Osprey 系统^[24]基于维表在各个节点全复制、事实表横向切分并冗余备份的数据分布策略, 将一星型查询划分为众多独立子查询. 每个子查询在执行失败时都可以在其备份节点上重新执行, 而不用重做整个查询, 使得数据仓库查询获得类似 MapReduce 的容错能力.

数据仓库扩展性方面的研究较少, 中国人民大学的 LinearDB 原型属于这方面的研究, 详细参见 7.1 节.

6.2 MapReduce 性能优化研究

MapReduce 的性能优化研究集中于对关系数据库的先进技术和特性的移植上.

Facebook 和俄亥俄州立大学合作, 将关系数据库的混合式存储模型应用于 Hadoop 平台, 提出了 RCFile 存储格式^[25]. 与之不同, 文献[26]将列存储技术引入 Hadoop 平台. Hadoop+ +^[27]系统运用了传统数据库的索引技术, 并通过分区数据并置 (Co-Partition) 的方式来提升性能. 文献[28-29]基于 MapReduce 实现了以流水线方式在各个操作符间传递数据, 从而缩短了任务执行时间; 在线聚集 (online aggregation) 的操作模式使得用户可以在查

询执行过程中看到部分较早返回的结果. 两者的不同之处在于前者仍基于 sort-merge 方式来实现流水线, 只是将排序等操作推向了 reducer, 部分情况下仍会出现流水线停顿的情况; 而后者利用 hash 方式来分布数据, 能实现更好的并行流水线操作. 文献[30]提出了 MRShare 架构, 对批量查询进行转换, 将可共享扫描、共享 Map 输出结果等的一组任务合并为一个, 以提升性能. 新加坡国立大学对影响 Hadoop 性能的因素做了深入分析^[12], 并提出了 5 项有效的优化技术, 使得 Hadoop 的性能提升了近 3 倍, 逼近关系数据库的性能.

近年的研究热点是基于 MapReduce 的连接操作的性能优化. 文献[31]对 MapReduce 平台的两表连接算法做了总结, 提出了 Map 端连接、Reduce 端连接及广播式连接等算法. 文献[32]对 MapReduce 框架进行了扩展, 在 Reduce 步骤后添加了一 Merge 步骤来完成连接操作, 提出的 Map-Reduce-Merge 框架可以同时处理两个异构数据源的数据. 对于多表连接, 当前主流的研究集中于仅通过一个任务来完成连接操作. 文献[33-34]提出了一对多复制的方法, 在 Map 阶段结束后, 为保证连接操作的局部性, 元组会被复制到多个节点. 但在节点数和数据量增大的情况下, 会带来 I/O 量及网络传输量的巨大增长. Llama^[35]通过预排序和按连接属性划分数据的方式来降低星型连接的代价, 但要付出可观的预处理代价和空间代价. 不同于以上等值连接优化, 文献[36]提出了针对任意连接条件的优化模型. 以上连接方式都是先执行连接, 然后在连接后的数据上执行聚集操作. 而中国人民大学的 Dumbo^[37]系统却采用了另一种更适应于 MapReduce 平台的思路: 先执行过滤聚集操作, 再基于聚集的数据执行连接. 详细参考 7.2 节.

6.3 HadoopDB 的改进

HadoopDB 于 2011 年针对其架构提出了两种连接优化技术和两种聚集优化技术^[38].

两种连接优化的核心思想都是尽可能地将数据的处理推入数据库层执行. 第 1 种优化方式是根据表与表之间的连接关系, 通过数据预分解, 使参与连接的数据尽可能分布在同一数据库内(参照分解法), 从而实现将连接操作下压进数据库内执行. 该算法的缺点是应用场景有限, 只适用于链式连接. 第

① 其最大问题既包括扩展性也包括性能, 这两项分别取决于并行数据库和 MapReduce(木桶原理), 其改进取决于这两种系统问题的改进.

2 种连接方式是针对广播式连接而设计的. 在执行连接前, 先在数据库内为每张参与连接的维表建立一张临时表, 使得连接操作尽可能在数据库内执行. 该算法的缺点是较多的网络传输和磁盘 I/O 操作.

两种聚集优化技术分别是连接后聚集和连接前聚集. 前者是执行完 Reduce 端连接后, 直接对符合条件的记录执行聚集操作; 后者是将所有数据先在数据库层执行聚集操作, 然后基于聚集数据执行连接操作, 并将不符合条件的聚集数据做减法操作. 该方式适用的条件有限, 主要用于参与连接和聚集的列的基数相乘后小于表记录数的情况.

总的来看, HadoopDB 的优化技术大都局限性较强, 对于复杂的连接操作(如环形连接等)仍不能下推至数据库层执行, 并未从根本上解决其性能问题.

7 MapReduce 和关系数据库技术的融合

综上所述, 当前研究大都集中于功能或特性的移植, 即从一个平台学习新的技术, 到另一平台重新实现和集成, 未涉及执行核心, 因此也没有从根本上解决大数据分析问题. 鉴于此, 中国人民大学高性能数据库实验室的研究小组采取了另一种思路: 从数据的组织和查询的执行两个核心层次入手, 融合关系数据库和 MapReduce 两种技术, 设计高性能的可扩展的抽象数据仓库查询处理框架. 该框架在支持高度可扩展的同时, 又具有关系数据库的性能. 我们团队尝试过两个研究方向: (1) 借鉴 MapReduce 的思想, 使 OLAP 查询的处理能像 MapReduce 一样高度可扩展(LinearDB 原型); (2) 利用关系数据库的技术, 使 MapReduce 在处理 OLAP 查询时, 逼近关系数据库的性能(Dumbo 原型).

7.1 LinearDB

LinearDB^[39]原型系统没有直接采用基于连接的星型模型(雪花模型), 而是对其进行了改造, 设计了扩展性更好的、基于扫描的无连接雪花模型 JFSS (Join-Free Snowflake Schema). 该模型的设计借鉴了泛关系模型的思想, 采用层次编码技术^[40]将维表层次信息压缩进事实表, 使得事实表可以独立执行维表上的谓词判断、聚集等操作, 从而使连接的数据在大规模机群上实现局部性, 消除了连接操作. 图 4 是一个星型模型和无连接雪花模型的对应示意图.

在执行层次上, LinearDB 吸取了 MapReduce 处理模式的设计思想, 将数据仓库查询的处理抽象

为 Transform、Reduce、Merge 3 个操作(TRM 执行模型): (1) Transform. 主节点对查询进行预处理, 将查询中作用于维表的操作(主要是谓词判断, group-by 聚集操作等)转换为事实表上的操作; (2) Reduce. 每个数据节点并行地扫描、聚集本地数据, 然后将处理结果返回给主节点; (3) Merge. 主节点对各个数据节点返回的结果进行合并, 并执行后续的过滤、排序等操作. 基于 TRM 执行模型, 查询可以划分为众多独立的子任务在大规模机群上并行执行. 执行过程中, 任何失败子任务都可以在其备份节点重新执行, 从而获得较好的容错能力. LinearDB 的执行代价主要取决于对事实表的 Reduce(主要是扫描)操作, 因此, LinearDB 可以获得近乎线性的大规模可扩展能力. 实验表明, 其性能比 HadoopDB 至少高出一个数量级^②.

LinearDB 的扩展能力、容错能力和高性能在于其巧妙地结合了关系数据库技术(层次编码技术、泛关系模式)和 MapReduce 处理模式的设计思想, 由此, 可以看出, 结合方式的不同可以导致系统能力的巨大差异.

7.2 Dumbo

Dumbo^[37]的核心思想是根据 MapReduce 的“过滤->聚集”的处理模式, 对 OLAP 查询的处理进行改造, 使其适应于 MapReduce 框架.

Dumbo 采用了类似于 LinearDB 的数据组织模式——利用层次编码技术将维表信息压缩进事实表, 区别在于 Dumbo 采用了更加有效的编码方式, 并针对 Hadoop 分布式文件系统的特点对数据的存储进行了优化.

在执行层次上, Dumbo 对 MapReduce 框架进行了扩展, 设计了新的 OLAP 查询处理框架——TMRP(Transform->Map->Reduce->Postprocess)处理框架(如图 5 所示). 在该框架中, 主节点首先对查询进行转换, 生成一个 MapReduce 任务来执行查询. 该任务在 Map 阶段以流水线方式扫描、聚集本地数据, 并只将本地的聚集数据传至 Reduce 阶段, 来进行数据的合并及聚集、排序等操作. 在 Postprocess 阶段, 主节点在数据节点上传的聚集数据之上执行连接操作. 实验表明, Dumbo 性能远超 Hadoop 和 HadoopDB.

由此我们可以看出, 复杂的 OLAP 查询在

① 又名 LaScOLAP.

② 此数据是基于我们自己实现的 (key, value) 列存模型之上测试得出的结果, 与文献[39]中所列性能有所不同.

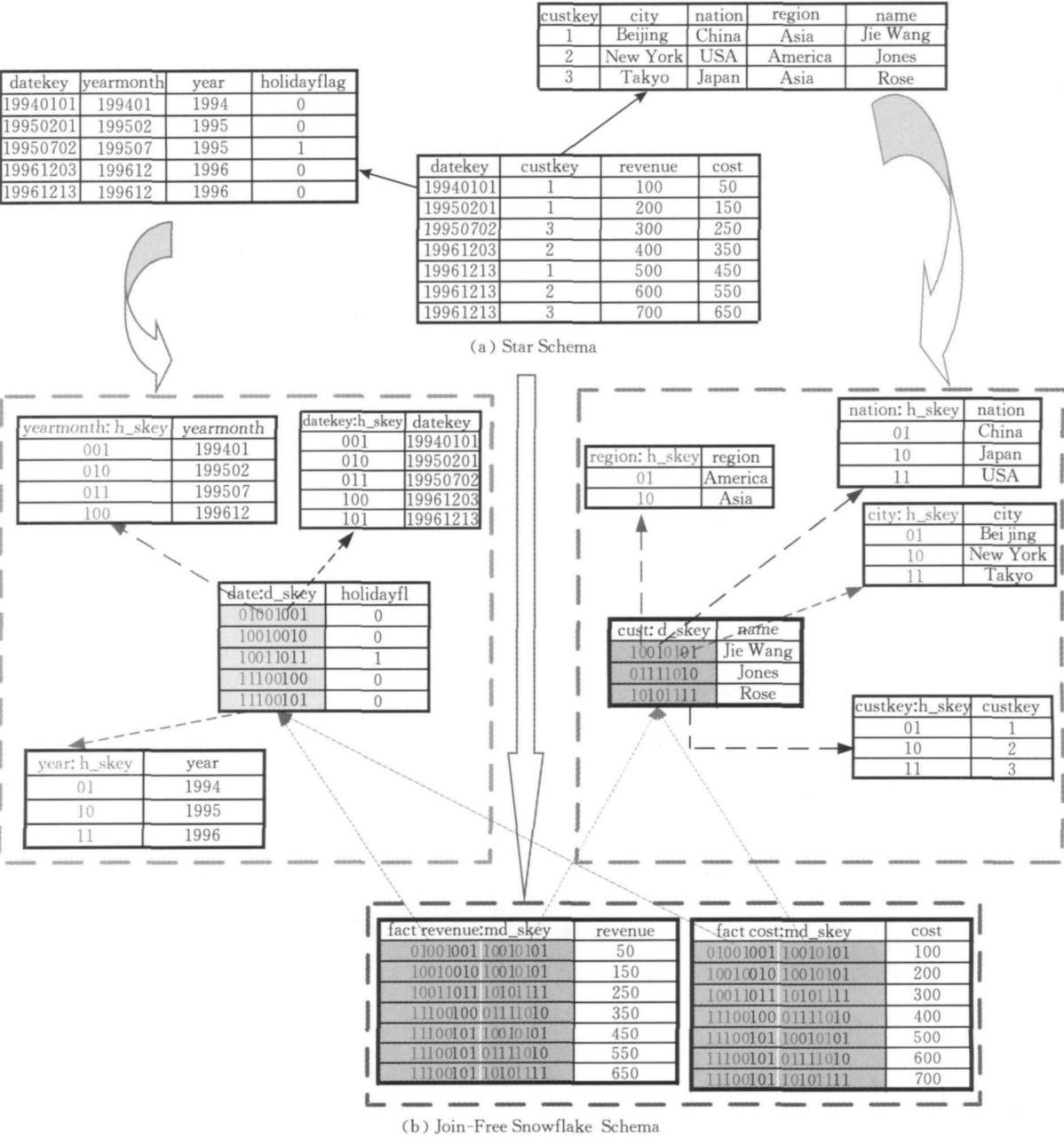


图 4 对比：一个典型星型模型与其对应的无连接雪花模型

MapReduce 框架下也可以获得接近甚至超越关系数据库的性能，其关键在于如何有效地结合关系数据库和 MapReduce 两种技术。仅仅停留于表层的移植和集成是难以从根本上解决大数据分析问题的。我们在文献[41]的研究中也展示了如何基于这种新的数据组织方式来实现复杂分析操作——百分位数的高效计算问题。

LinearDB 和 Dumbo 虽然基本可以达到预期的设计目标，但两者都需要对数据进行预处理，其预处理代价是普通加载时间的 7 倍左右。因此其应对变化的能力还较弱，这是我们未来的工作内容之一。

8 研究展望

当前 3 个方向的研究都不能完美地解决大数据分析问题，也就意味着每个方向都有极具挑战性的工作等待着我们。

对并行数据库来说，其扩展性近年虽有较大改善（如 Greenplum 和 Aster Data 都是面向 PB 级数据规模设计开发的），但距离大数据分析需求仍有较大差距。因此，如何改善并行数据库的扩展能力是一项非常有挑战的工作，该项研究将同时涉及数据一致性协议、容错性、性能等数据库领域的

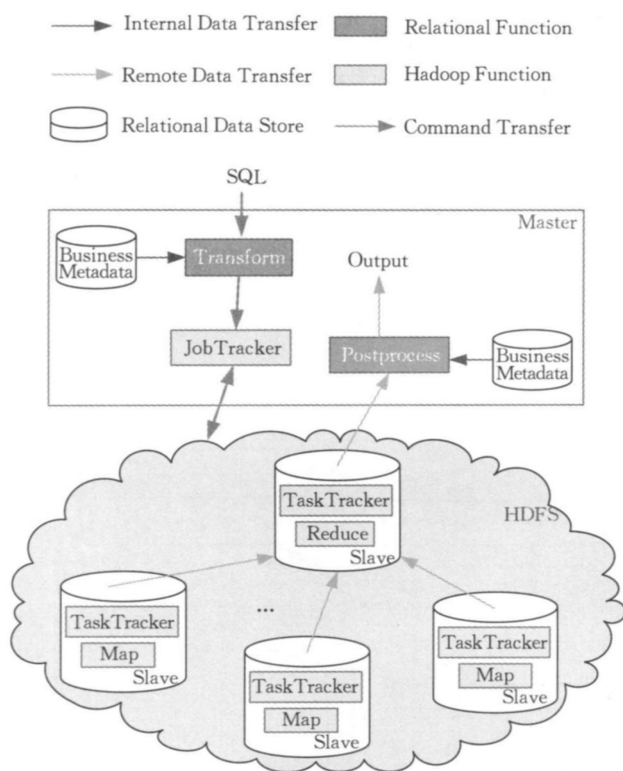


图 5 Dumbo 架构(深灰色部分是新增模块, 剩余部分是 Hadoop 自带模块)

诸多方面。

混合式架构方案可以复用已有成果, 开发量较小, 但只是简单的功能集成似乎并不能有效解决大数据的分析问题, 因此该方向还需要更加深入的研究工作, 比如从数据模型及查询处理模式上进行研究, 使两者能较自然地结合起来, 这将是一项非常有意义的工作。中国人民大学的 Dumbo^[37] 系统即是在深层结合方向上努力的一个例子。

相比于前两者, MapReduce 的性能优化进展迅速, 其性能正逐步逼近关系数据库。该方向的研究又分为两个方向: 理论界侧重于利用关系数据库技术及理论改善 MapReduce 的性能; 工业界侧重于基于 MapReduce 平台开发高效的应用软件。针对数据仓库领域, 我们认为如下几个研究方向比较重要, 且目前研究还较少涉及:

(1) 多维数据的预计算。MapReduce 更多针对的是一次性分析操作。大数据上的分析操作虽然难以预测, 但传统的分析, 如基于报表和多维数据的分析仍占多数。因此, MapReduce 平台也可以利用预计算等手段加快数据分析的速度。基于存储空间的考虑(可以想象, 在爆炸数据之上计算数据立方体需要付出昂贵的存储空间代价), MOLAP 是不可取

的, 混合式 OLAP(HOLAP) 应该是 MapReduce 平台的优选 OLAP 实现方案。具体研究如: ① 基于 MapReduce 框架的高效 Cube 计算算法; ② 物化视图的选择问题, 即物化哪些数据; ③ 不同分析操作的物化手段(比如预测分析操作的物化)及如何基于物化的数据进行复杂分析操作(如数据访问路径的选择问题)。

(2) 各种分析操作的并行化实现。大数据分析需要高效的复杂统计分析功能的支持。IBM 将开源统计分析软件 R 集成进 Hadoop 平台^[42], 增强了 Hadoop 的统计分析功能。但更具挑战性的问题是, 如何基于 MapReduce 框架设计可并行化的、高效的分析算法。尤其需要强调的是, 鉴于移动数据的巨大代价, 这些算法应基于移动计算的方式来实现。

(3) 查询共享。MapReduce 采用步步物化的处理方式, 导致其 I/O 代价及网络传输代价较高。一种有效的降低该代价的方式是在多个查询间共享物化的中间结果, 甚至原始数据, 以分摊代价并避免重复计算。因此如何在多查询间共享中间结果将是一项非常有实际应用价值的研究。

(4) 用户接口。如何较好地实现数据分析的展示和操作, 尤其是复杂分析操作的直观展示。

(5) Hadoop 可靠性研究。当前 Hadoop 采用主从结构, 由此决定了主节点一旦失效, 将会出现整个系统失效的局面。因此, 如何在不影响 Hadoop 现有实现的前提下, 提高主节点的可靠性, 将是一项切实的研究。

(6) 数据压缩。MapReduce 的执行模型决定了其性能取决于 I/O 和网络传输代价。文献[11]在比较并行数据库和 MapReduce 基于压缩数据的性能时, 发现压缩技术并没有改善 Hadoop 的性能^①。但实际情况是, 压缩不仅可以节省空间, 节省 I/O 及网络带宽, 还可以利用当前 CPU 的多核并行计算能力, 平衡 I/O 和 CPU 的处理能力, 从而提高性能。比如并行数据库利用数据压缩后, 性能往往可以大幅提升。此后, 文献[25-26]的研究成功地利用压缩技术提升了 Hadoop 的性能。但这些研究都基于各自的存储模型, 而非 Hadoop 的默认存储模式(行存模型)。因此, MapReduce 上的压缩是一个尚待研究的重要问题。

(7) 多维索引研究。如何基于 MapReduce 框架实现多维索引, 加快多维数据的检索速度。

① 原因未知。

当然, 仍有许多其它研究工作, 比如基于 Hadoop 的实时数据分析、弹性研究、数据一致性研究等, 都是非常有挑战和意义的研究, 限于篇幅我们不再赘述。

9 总 结

本文对大数据分析的主流实现平台(并行数据库、MapReduce 及两者的混合架构)进行了评价、归纳与对比分析, 介绍了中国人民大学在大数据分析方面的研究, 并对当前的研究进行了归纳。从文中可以看出, 每种分析平台都不是完美的, 在大数据面前, 都有很长的路要走。大数据分析迫使我们反思传统的数据仓库架构, 虚心地研究 MapReduce 等新生平台, 以站在更高的层次来思考问题, 从而找到适应时代需求的数据仓库架构。

参 考 文 献

- [1] WinterCorp: 2005 TopTen Program Summary. http://www.wintercorp.com/WhitePapers/WC_TopTenWP.pdf
- [2] TDWI Checklist Report: Big Data Analytics. <http://tdwi.org/research/2010/08/Big-Data-Analytics.aspx>
- [3] Chaudhuri S, Dayal U. An overview of data warehousing and OLAP technology. SIGMOD Rec, 1997, 26(1): 65-74
- [4] Madden S, DeWitt D J, Stonebraker M. Database parallelism choices greatly impact scalability. DatabaseColumn Blog. <http://www.databasecolumn.com/2007/10/database-parallelism-choices.html>
- [5] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters//Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI'04). San Francisco, California, USA, 2004: 137-150
- [6] DeWitt D J, Gerber R H, Graefe G, Heytens M L, Kumar K B, Muralikrishna M. GAMMA — A high performance dataflow database machine//Proceedings of the 12th International Conference on Very Large Data Bases (VLDB'86). Kyoto, Japan, 1986: 228-237
- [7] Fushimi S, Kitsuregawa M, Tanaka H. An overview of the system software of a parallel relational database machine//Proceedings of the 12th International Conference on Very Large Data Bases(VLDB'86). Kyoto, Japan, 1986: 209-219
- [8] Brewer E A. Towards robust distributed systems//Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC'00). Portland, Oregon, USA, 2000: 7
- [9] <http://www.dbms2.com/2008/08/26/known-applications-of-mapreduce/>
- [10] <http://hadoop.apache.org>
- [11] Pavlo A, Paulson E, Rasin A, Abadi D J, DeWitt D J, Madden S, Stonebraker M. A comparison of approaches to large-scale data analysis//Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'09). Providence, Rhode Island, USA, 2009: 165-178
- [12] Jiang D, Ooi B C, Shi L, Wu S. The performance of MapReduce: An in-depth study. PVLDB, 2010, 3(1): 472-483
- [13] Stonebraker M, Abadi D J, DeWitt D J, Madden S, Paulson E, Pavlo A, Rasin A. MapReduce and parallel DBMSs: Friends or foes? Communications of the ACM, 2010, 53(1): 64-71
- [14] Dean J, Ghemawat S. MapReduce: A flexible data processing tool. Communications of the ACM, 2010, 53(1): 72-77
- [15] <http://www.asterdata.com/product/mapreduce.php>
- [16] <http://www.greenplum.com/technology/mapreduce/>
- [17] <http://hive.apache.org/>
- [18] Olston C, Reed B, Srivastava U, Kumar R, Tomkins Andrew. Pig latin: A not-so-foreign language for data processing//Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'08). Vancouver, BC, Canada, 2008: 1099-1110
- [19] Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel J Abadi, Alexander Rasin, Avi Silberschatz. HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads//Proceedings of the 35th International Conference on Very Large Data Bases (VLDB'09). Lyon, France, 2009: 733-743
- [20] Hadapt Inc. <http://www.hadapt.com>
- [21] <http://www.vertica.com/the-analytics-platform/native-bit-and-hadoop-mapreduce-integration/>
- [22] Xu Y, Kostamaa P. Integrating hadoop and parallel DBMS//Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'10). Indianapolis, Indiana, USA, 2010: 969-974
- [23] Upadhyaya P, Kwon Y C, Balazinska M. A latency and fault-tolerance optimizer for online parallel query plans//Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'11). Athens, Greece, 2011: 241-252
- [24] Yang C, Yen C, Tan C, Madden S. Osprey: Implementing MapReduce-style fault tolerance in a shared-nothing distributed database//Proceedings of the 24th International Conference on Data Engineering (ICDE'10). Long Beach, California, USA, 2010: 657-668
- [25] He Yongqiang, Lee Rubao, Huai Yin, Shao Zheng, Jain Namit, Zhang Xiaodong, Xu Zhiwei. RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems//Proceedings of the 24th International Conference on Data Engineering (ICDE'11). Hannover, Germany, 2011: 1199-1208
- [26] Floratou A, Patel J M, Shekita E J, Tata Sandeep. Column-oriented storage techniques for MapReduce. PVLDB, 2011, 4(7): 419-429
- [27] Jens Dittrich, Jorge-Arnulfo Quiroz Ruiz, Alekh Jindal, Yagiz Kargin, Vinay Setty, Jrg Schad. Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). PVLDB, 2010, 3(1): 518-529
- [28] Condie T, Conway N, Alvaro P, Hellerstein J M, Elmeleegy

- K, Sears R. MapReduce online//Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation(NSDI' 10). San Jose, California, 2010: 313-328
- [29] Li Boduo, Mazur Edward, Diao Yanlei, McGregor Andrew, Shenoy Prashant J. A platform for scalable one-pass analytics using MapReduce//Proceedings of the ACM SIGMOD International Conference on Management of Data(SIGMOD' 11). Athens, Greece, 2011: 985-996
- [30] Nykiel T, Potamias M, Mishra C, Kollis G, Koudas N. MRShare: Sharing across multiple queries in MapReduce. PVLDB, 2010, 3(1): 494-505
- [31] Blanas S, Patel Jignesh, Ercegovac V, Rao J, Shekita E J, Tian Y. A comparison of join algorithms for log processing in MapReduce//Proceedings of the ACM SIGMOD International Conference on Management of Data(SIGMOD' 10). Indianapolis, Indiana, USA, 2010: 975-986
- [32] Yang H-C, Dasdan A, Hsiao R-L, Parker D S. Map-reduce-merge: Simplified relational data processing on large clusters//Proceedings of the ACM SIGMOD International Conference on Management of Data(SIGMOD' 07). Beijing, China, 2007: 1029-1040
- [33] Afrati F N, Ullman J D. Optimizing joins in a map-reduce environment//Proceedings of the 13th International Conference on Extending Database Technology. Lausanne, Switzerland, 2010: 99-110
- [34] Jiang D, Tung A K H, Chen G. Map join-reduce: Towards scalable and efficient data analysis on large clusters. TKDE, 2010, 23(9): 1299-1311
- [35] Lin Y, Agrawal D, Chen C, Ooi B C, Wu S. Llama: Leveraging columnar storage for scalable join processing in the MapReduce framework//Proceedings of the ACM SIGMOD International Conference on Management of Data(SIGMOD' 11). Athens, Greece, 2011: 961-972
- [36] Okcan A, Riedewald M. Processing theta joins using MapReduce//Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD' 11). Athens, Greece, 2011: 949-960
- [37] Wang Huiju, Wang Shan, Qin Xiongpai, Li Furong, Zhou Xuan, Qin Zuoyan, Zhu Qing. Efficient star query processing on Hadoop—A hierarchy encoding based approach (Technical report)
- [38] Bajda-Pawlikowski Kamil, Abadi Daniel J, Silberschatz Avi, Paulson Erik. Efficient processing of data warehousing queries in a split execution environment//Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD' 11). Athens, Greece, 2011: 985-996. 2011: 1165-1176
- [39] Wang Huiju, Qin Xiongpai, Zhang Yansong, Wang Shan, Wang Zhanwei. LinearDB: A relational approach to make data warehouse scale like MapReduce//Proceedings of the Database Systems for Advanced Applications 16th International Conference (DASFAA' 11). Hong Kong, China, 2011: 306-320
- [40] Karayannis N, Tsois A, Sellis T K, Pieringer R, Markl V, Ramsak F, Fenk R, Elhardt K, Bayer R. Processing star queries on hierarchically-clustered fact tables//Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 02). Hong Kong, China, 2002: 730-741
- [41] Qin Xiongpai, Wang Huiju, Du Xiaoyong, Wang Shan. Parallel aggregation queries over star schema: A hierarchical encoding scheme and efficient percentile computing as a case//Proceedings of the 9th IEEE International Symposium on Parallel and Distributed Processing with Applications(ISPA' 11). Busan, Korea, 2011: 329-334
- [42] Das S, Sismanis Y, Beyer K S, Gemulla R, Haas P J, McPherson J. Ricardo: Integrating R and Hadoop//Proceedings of the ACM SIGMOD International Conference on Management of Data(SIGMOD' 10). Athens, Greece, Indianapolis, Indiana, USA, 2010: 987-998



WANG Shan, born in 1944, professor, Ph.D. supervisor. Her research interests include high performance database, data warehouse and knowledge engineering.

WANG Hui-Ju, born in 1979, Ph.D. candidate. His research interests include data warehouse, parallel database, high performance database.

QIN Xiong-Pai, born in 1973, Ph.D., lecturer. His research interests include query optimization, main-memory database, parallel database.

ZHOU Xuan, born in 1979, Ph.D., associate professor. His current research interests include IR, and high performance databases.

Background

We have studied large scale data warehouse system since 2006, and focused on scalable main-memory OLAP system, scalable star queries processing for SN parallel database and Hadoop platform, etc. Until now, we have developed five prototypes—ScaMMDB, ScaMMDB II, MOSSDB, LinearDB, and Dumbo. Currently we are working on the optimization of Hadoop. This work is partly supported by the Important National Science & Technology Specific Projects of

China (“HGX” Projects, Grant No 2010ZX01042-001-002), the National Natural Science Foundation of China (Grant No. 61070054, 61170013), the Fundamental Research Funds for the Central Universities (the Research Funds of Renmin University of China, Grant No 10XNH018), and the Graduate Science Foundation of Renmin University of China (Grant No. 10XNH096 and No 11XNH120).