厦门大学林子雨编著

《大数据技术原理与应用》

第 14 章 基于 Hadoop 的数据仓库 Hive

（版本号：2016 年 4 月 6 日版本）

（备注：2015 年 8 月 1 日第一版教材中没有本章，本章为 2016 年新增内容，将被放入第二版教材中）

主讲教师：林子雨
厦门大学数据库实验室
二零一六年四月

中国高校大数据课程
公 共 服 务 平 台

中国高校大数据课程公共服务平台，由中国高校首个“数字教师”的提出者和建设者——林子雨老师发起，由厦门大学数据库实验室全力打造，由厦门大学云计算与大数据研究中心、海峡云计算与大数据应用研究中心携手共建。这是国内第一个服务于高校大数据课程建设的公共服务平台，旨在促进国内高校大数据课程体系建设，提高大数据课程教学水平，降低大数据课程学习门槛，提升学生课程学习效果。

平台为教师开展大数据教学和学生学习大数据课程，提供全方位、一站式**免费**服务，包括**讲义 PPT**、教学大纲、备课指南、**学习指南**、上机习题、**授课视频**、技术资料等。

百度搜索“厦门大学数据库实验室”，访问平台主页，或直接访问平台地址：
http://dblab.xmu.edu.cn/post/bigdata-teaching-platform/

扫一扫访问平台主页

《大数据技术原理与应用——概念、存储、处理、分析与应用》，由厦门大学计算机科学系教师林子雨博士编著，是中国高校第一本系统介绍大数据知识的专业教材。本书定位为大数据技术入门教材，为读者搭建起通向“大数据知识空间”的桥梁和纽带，以“构建知识体系、阐明基本原理、引导初级实践、了解相关应用”为原则，为读者在大数据领域“深耕细作”奠定基础、指明方向。

全书共有 13 章，系统地论述了大数据的基本概念、大数据处理架构 Hadoop、分布式文件系统 HDFS、分布式数据 库 HBase、NoSQL 数据库、云数据库、分布式并行编程模型 MapReduce、流计算、图计算、数据可视化以及大数据在互联网、生物医学和物流等各个领域的应用。在 Hadoop、HDFS、HBase 和 MapReduce 等重要章节，安排了入门级的实践操作，让读者更好地学习和掌握大数据关键技术。

本书可以作为高等院校计算机专业、信息管理等相关专业的大数据课程教材，也可供相关技术人员参考、学习、培训之用。

欢迎访问《大数据技术原理与应用——概念、存储、处理、分析与应用》教材官方网站：

http://dblab.xmu.edu.cn/post/bigdata

扫一扫访问教材官网

目录

第 14 章 基于 Hadoop 的数据仓库 Hive

Hive 是一个基于 Hadoop 的数据仓库工具，可以用于对存储在 Hadoop 文件中的数据集进行数据整理、特殊查询和分析处理。Hive 的学习门槛比较低，因为它提供了类似于关系数据库 SQL 语言的查询语言——HiveQL，可以通过 HiveQL 语句快速实现简单的 MapReduce 统计，Hive 自身可以将 HiveQL 语句快速转换成 MapReduce 任务进行运行，而不必开发专门的 MapReduce 应用程序，因而十分适合数据仓库的统计分析。

本章首先介绍了数据仓库的概念、Hive 的基本特征、与其他组件之间的关系、与传统数据库的区别以及它在企业中的具体应用；接着详细介绍了 Hive 的系统架构，包括基本组成模块、工作原理和几种外部访问方式，描述了 Hive 的具体应用及 Hive HA 原理；同时，介绍了新一代开源大数据分析引擎 Impala，它提供了与 Hive 类似的功能，但是，速度要比 Hive 快许多；最后，以单词统计为例，介绍了如何使用 Hive 进行简单编程，并说明了 Hive 编程相对于 MapReduce 编程的优势。

14.1 概述

14.1.1 数据仓库概念

数据仓库的一个比较公认的定义是由 W. H. Inmon 给出的，即“数据仓库（Data Warehouse）是一个面向主题的（Subject Oriented）、集成的（Integrated）、相对稳定的（Non-Volatile）、反映历史变化（Time Variant）的数据集合，用于支持管理决策”。

随着信息技术的普及和企业信息化建设步伐的加快，企业逐步认识到建立企业范围内的统一数据存储的重要性，越来越多的企业已经建立或正在着手建立企业数据仓库。企业数据仓库有效集成了来自不同部门、不同地理位置、具有不同格式的数据，为企业管理决策者提供了企业范围内的单一数据视图，从而为综合分析和科学决策奠定了坚实的基础。常见的传统数据仓库工具供应商或产品主要包括 Oracle、Business Objects、IBM、Sybase、Informix、NCR、Microsoft、SAS 等。

数据仓库的体系结构（如图 14-1 所示）通常包含四个层次：数据源、数据存储和管理、数据服务、数据应用，具体如下：

- 数据源：是数据仓库的数据来源，包括了外部数据、现有业务系统和文档资料等；
- 数据集成：完成数据的抽取、清洗、转换和加载任务，数据源中的数据采用 ETL 工具

以固定的周期加载到数据仓库中；

- 数据存储和管理：这一层次主要涉及对数据的存储和管理，包括数据仓库、数据集市、数据仓库检测、运行与维护工具和元数据管理等；
- 数据服务：为前端工具和应用提供数据服务，可以直接从数据仓库中获取数据供前端应用使用，也可以通过 OLAP 服务器为前端应用提供更加复杂的数据服务。OLAP 服务器提供了不同聚集粒度的多维数据集合，使得应用不需要直接访问数据仓库中的底层细节数据，大大减少了数据计算量，提高了查询响应速度。OLAP 服务器还支持针对多维数据集的上钻、下探、切片、切块和旋转等操作，增强了多维数据分析能力；
- 数据应用：这一层次直接面向最终用户，包括数据查询工具、自由报表工具、数据分析工具、数据挖掘工具和各类应用系统。

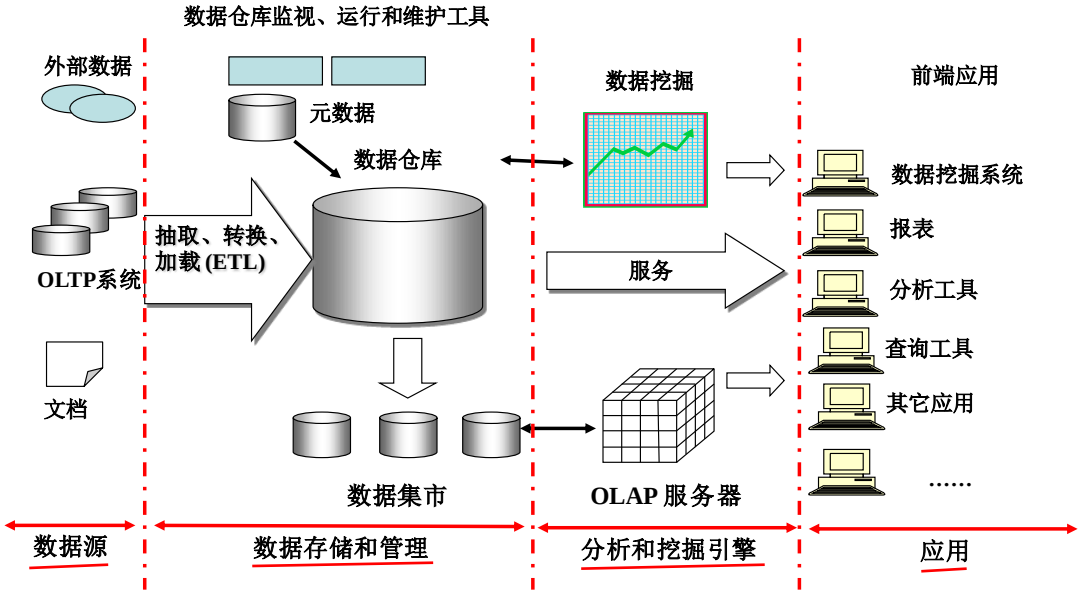


图 14-1 数据仓库的体系结构

14.1.2 传统数据仓库面临的挑战

随着大数据时代的全面到来，传统数据仓库面临的挑战主要包括以下几个方面：

（1）无法满足快速增长的海量数据存储需求。目前企业数据增长速度非常快，动辄几十 TB 的数据，已经大大超出了 Oracle/DB2 等传统数据仓库的处理能力，因为传统数据仓库大都基于关系数据库，关系数据库横向扩展性较差，纵向可扩展性有限。

（2）无法有效处理不同类型的数据。传统数据仓库通常只能存储处理结构化数据，但是，随着企业业务的发展，企业中部署的系统越来越多，数据源的数据格式越来越丰富，很显然，传统数据仓库是无法处理如此众多的数据类型的。

（3）计算和处理能力不足。传统数据仓库由于建立在关系数据库基础之上，因此，会

存在一个很大的痛点，即计算和处理能力不足，当数据量到达 TB 量级后基本无法获得好的性能。

14.1.3 Hive 简介

Hive 是一个构建于 Hadoop 顶层的数据仓库工具，由 Facebook 公司开发，并在 2008 年 8 月开源。Hive 在某种程度上可以看作是用户编程接口，其本身并不存储和处理数据，而是依赖 HDFS 来存储数据，依赖 MapReduce 来处理数据。Hive 定义了简单的类似 SQL 的查询语言——HiveQL，它与大部分 SQL 语法兼容，但是，并不完全支持 SQL 标准，比如，HiveSQL 不支持更新操作，也不支持索引和事务，它的子查询和连接操作也存在很多局限。

HiveQL 语句可以快速实现简单的 MapReduce 任务，这样用户通过编写的 HiveQL 语句就可以运行 MapReduce 任务，不必编写复杂的 MapReduce 应用程序。对于 Java 开发工程师而言，就不必花费大量精力在记忆常见的数据运算与底层的 MapReduce Java API 的对应关系上；对于 DBA 来说，可以很容易把原来构建在关系数据库上的数据仓库应用程序移植到 Hadoop 平台上。所以说，Hive 是一个可以有效、合理、直观地组织和使用数据的分析工具。

现在，Hive 作为 Hadoop 平台上的数据仓库工具，其应用已经十分广泛，主要是因为它具有的特点非常适合数据仓库应用程序。首先，Hive 把 HiveQL 语句转换成 MapReduce 任务后，采用批处理的方式对海量数据进行处理。数据仓库存储的是静态数据，构建于数据仓库上的应用程序只进行相关的静态数据分析，不需要快速响应给出结果，而且数据本身也不会频繁变化，因而很适合采用 MapReduce 进行批处理。其次，Hive 本身还提供了一系列对数据进行提取转化加载的工具，可以存储、查询和分析存储在 Hadoop 中的大规模数据。这些工具能够很好地满足数据仓库各种应用场景，包括维护海量数据、对数据进行挖掘、形成意见和报告等。

14.1.4 Hive 与 Hadoop 生态系统中其他组件的关系

图 14-2 描述了 Hadoop 生态系统中 Hive 与其他组件之间的关系。HDFS 作为高可靠的底层存储，用来存储海量数据；MapReduce 对这些海量数据进行批处理，实现高性能计算；Hive 架构在 MapReduce、HDFS 之上，其自身并不存储和处理数据，需要分别借助于 HDFS 和 MapReduce 实现数据的存储和处理，用 HiveQL 语句编写的处理逻辑，最终都要转化为 MapReduce 任务来运行；Pig 可以作为 Hive 的替代工具，是一种数据流语言和运行环境，适合用于在 Hadoop 平台上查询半结构化数据集，常用于 ETL 过程的一部分，即将外部数据装载到 Hadoop 集群中，然后转换为用户需要的数据格式；HBase 是一个面向列的、分布式的、可伸缩的数据库，它可以提供数据的实时访问功能，而 Hive 只能处理静态数据，主要是 BI 报表数据，就设计初衷而言，在 Hadoop 上设计 Hive，是为了减少复杂 MapReduce 应

用程序的编写工作，在 Hadoop 上设计 HBase 则是为了实现对数据的实时访问，所以，HBase 与 Hive 的功能是互补的，它实现了 Hive 不能提供的功能。

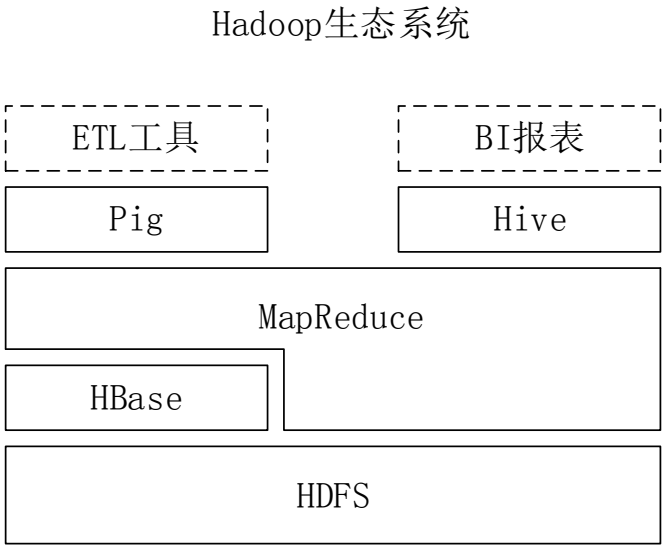


图 14-2 Hadoop 生态系统中 Hive 与其他部分的关系

14.1.5 Hive 与传统数据库的对比分析

Hive 在很多方面和传统的关系数据库类似，但是，它的底层依赖的是 HDFS 和 MapReduce，所以，在很多方面又有别于传统数据库。表 14-1 从数据插入、数据更新、索引、分区、执行延迟、扩展性等方面，对 Hive 和传统数据库进行了对比分析。

表 14-1 Hive 与传统数据库的对比

对比内容	Hive	传统数据库
数据插入	支持批量导入	支持单条和批量导入
数据更新	不支持	支持
索引	支持	支持
分区	支持	支持
执行延迟	高	低
扩展性	好	有限

在传统数据库中，同时支持导入单条数据和批量数据，而 Hive 中仅支持批量导入数据，因为 Hive 主要用来支持大规模数据集上的数据仓库应用程序的运行，常见操作是全表扫描，所以，单条插入功能对 Hive 并不实用。更新和索引是传统数据库中很重要的特性，Hive 不支持数据更新，但是，在 Hive 0.7 版本以后已经可以支持索引了。Hive 是一个数据仓库工具，而数据仓库中存放的是静态数据，所以，Hive 不支持对数据进行更新。Hive 不像传统的关系型数据库那样有键的概念，它只能提供有限的索引功能，使用户可以在某些列上创建索引，从而加速一些查询操作，Hive 中给一个表创建的索引数据，会被保存在另外的表中。

传统的数据库提供分区功能来改善大型表以及具有各种访问模式的表的可伸缩性、可管理性，以及提高数据库效率。Hive 也支持分区功能，Hive 表是分区的形式进行组织的，根据“分区列”的值对表进行粗略的划分，从而加快数据的查询速度。因为 Hive 构建在 HDFS 与 MapReduce 之上，所以，相对于传统数据库而言，Hive 的延迟会比较高，传统数据库中的 SQL 语句的延迟一般少于一秒，而 HiveQL 语句的延迟会达到分钟级。传统关系数据库很难实现横向扩展，纵向扩展的空间也很有限。相反，Hive 的开发和运行环境是基于 Hadoop 集群的，所以具有较好的横向可扩展性。

14.1.6 Hive 在企业中的部署和应用

1. Hive 在企业大数据分析平台中的应用

Hadoop 除了广泛应用到云计算平台上实现海量数据计算外，还在很早之前就被应用到了企业大数据分析平台的设计与实现。当前企业中部署的大数据分析平台，除了依赖于 Hadoop 的基本组件 HDFS 和 MapReduce 外，还结合使用了 Hive、Pig、HBase 与 Mahout，从而满足不同业务场景的需求，图 14-3 描述了企业实际应用中一种常见的大数据分析平台部署框架。

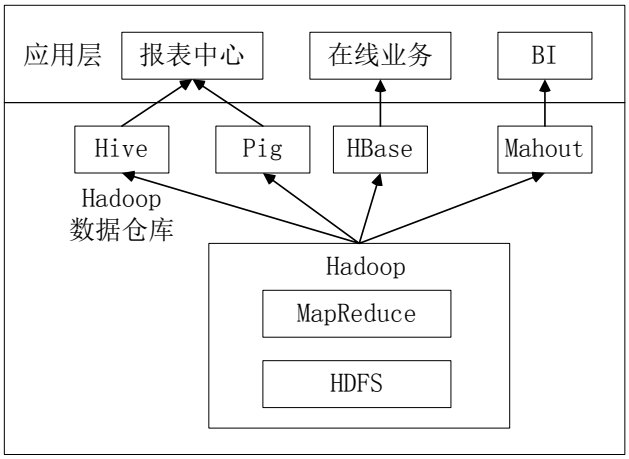


图 14-3 企业中一种常见的大数据分析平台部署框架

在这种部署架构中，Hive 和 Pig 主要应用于报表中心，其中，Hive 用于报表分析，Pig 用于报表中数据的转换工作。因为，HDFS 不支持随机读写操作，而 HBase 正是为此开发的，可以较好地支持实时访问数据，所以，HBase 主要用于在线业务。Mahout 提供了一些可扩展的机器学习领域的经典算法的实现，旨在帮助开发人员更加方便快捷地创建商务智能应用程序，所以，Mahout 常用于 BI（商务智能）。

2.Hive 在 Facebook 公司中的应用

Facebook 公司开发了数据仓库工具 Hive，并在企业内部进行了大量部署。随着 Facebook 网站使用量的增加，网站上需要处理和存储的日志和维度数据激增。继续在 Oracle 系统上

实现数据仓库，其性能和可扩展性已经不能满足需求，于是，Facebook 开始使用 Hadoop。图 14-4 展示了 Facebook 的数据架构的基本组件以及这些组件间的数据流。

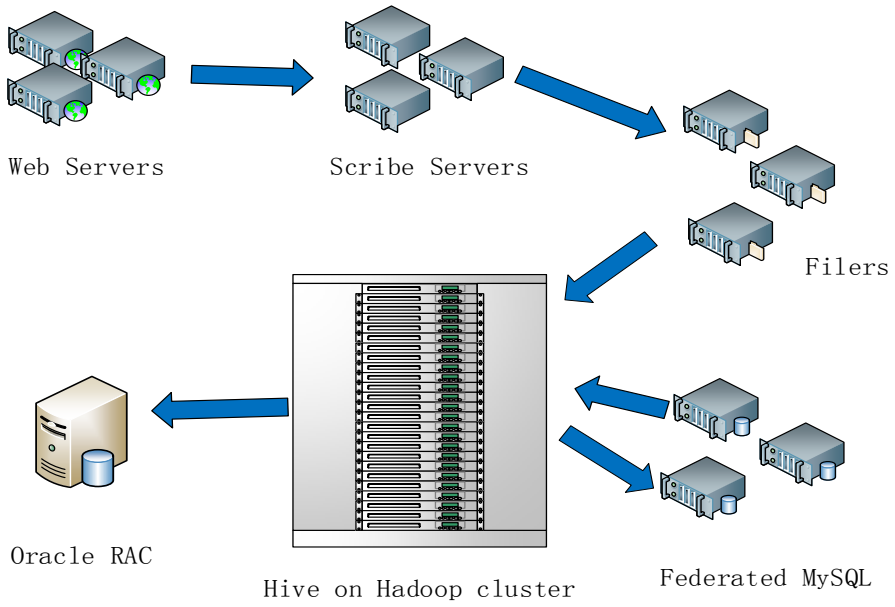


图 14-4 Facebook 的数据仓库架构

如图 14-4 所示，数据处理过程如下：首先，由 Web 服务器及内部服务（如搜索后台）产生日志数据，然后，Scribe 服务器把几百个甚至上千个日志数据集存放在几个甚至几十个网络文件服务器（Filers）上。网络文件服务器上的大部分日志文件被复制存放在 HDFS 系统中。每天，维度数据也从内部的 MySQL 数据库上复制到这个 HDFS 系统中。然后，Hive 为 HDFS 收集的所有数据创建一个数据仓库，用户可以通过编写 HiveQL 语言创建各种概要信息和报表以及历史数据分析，同时，内部的 MySQL 数据库也可以从中获取处理后的数据，并把需要实时联机访问的数据存放在 Oracle RAC 上，这里的 RAC（Real Application Clusters）是 Oracle 的一项核心技术，可以在低成本服务器上构建高可用性数据库系统。

14.2 Hive 系统架构

图 14-5 显示了 Hive 的主要组成模块、Hive 如何与 Hadoop 交互工作以及从外部访问 Hive 的几种典型方式。

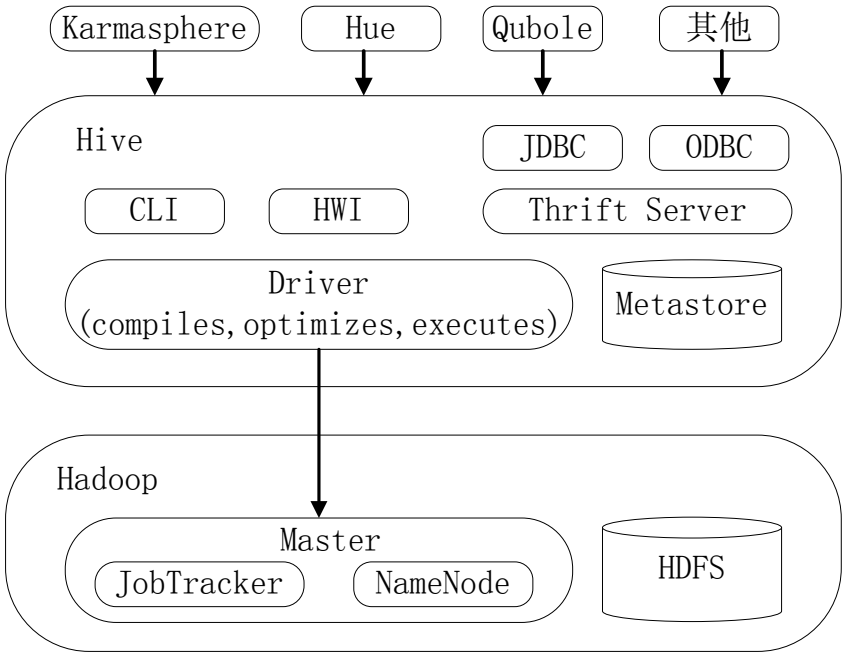


图 14-5 Hive 系统架构

Hive 主要由以下三个模块组成：用户接口模块、驱动模块以及元数据存储模块。用户接口模块包括 CLI、HWI、JDBC、ODBC、Thrift Server 等，用来实现外部应用对 Hive 的访问。CLI 是 Hive 自带的一个命令行界面，HWI 是 Hive 的一个简单网页界面，JDBC、ODBC 以及 Thrift Server 可以向用户提供进行编程访问的接口，其中，Thrift Server 是基于 Thrift 软件框架开发的，它提供 Hive 的 RPC 通信接口。驱动模块（Driver）包括编译器、优化器、执行器等，负责把 HiveSQL 语句转换成一系列 MapReduce 作业（下一节介绍转换过程的基本原理），所有命令和查询都会进入到驱动模块，通过该模块对输入进行解析编译，对计算过程进行优化，然后按照指定的步骤执行。元数据存储模块（Metastore）是一个独立的关系型数据库，通常是与 MySQL 数据库连接后创建的一个 MySQL 实例，也可以是 Hive 自带的 derby 数据库实例。元数据存储模块中主要保存表模式和其他系统元数据，如表的名称、表的列及其属性、表的分区及其属性、表的属性、表中数据所在位置信息等。

除了用 Hive 自带的 CLI 和 HWI 工具来访问 Hive 外，对于那些更喜欢用图形界面的用户，可以采用图 14-5 中列举的几种典型外部访问工具：Karmasphere、Hue、Qubole 等。

Karmasphere 是由 Karmasphere 公司发布的一个商业产品。Karmasphere 可以直接访问 Hadoop 里面结构化和非结构化的数据，还可以运用 SQL 及其他语言进行即席查询和进一步的分析。Karmasphere 还为开发人员提供了一种图形化环境，可以在里面开发自定义算法，为应用程序和可重复的生产流程创建实用的数据集。

Hue 是由 Cloudera 公司提供的一个开源项目，它是运营和开发 Hadoop 应用的图形化用户界面。Hue 程序被整合到一个类似桌面的环境，以 Web 程序的形式发布，对于单独的用户来说不需要额外的安装。

Qubole 公司提供了“Hive 即服务”的方式。Qubole 服务托管在亚马逊的 AWS 云平台，这样用户在分析存储在亚马逊 S3 中的数据集时，就无需自己进行 Hadoop 系统管理，Qubole 提供的 Hadoop 服务能够根据用户的工作负载动态调整服务器资源配置，实现随需计算，对于用户来说，这大大简化了大数据应用的复杂性，同时也大大降低了成本。

14.3 Hive 工作原理

Hive 可以快速实现简单的 MapReduce 统计，主要是通过自身组件把 HiveQL 语句转换成 MapReduce 任务来实现的。下面首先介绍在没有使用 Hive 时，几个简单 SQL 语句是如何转化为 MapReduce 任务来执行的；然后，再详细介绍在 Hive 中 SQL 语句（即 HiveQL）是如何转化为 MapReduce 任务来执行的。

14.3.1 SQL 语句转换成 MapReduce 作业的基本原理

（1）用 MapReduce 实现连接操作

假设参与连接（join）的两个表分别为用户表 User 和订单表 Order，User 表有两个属性，即 uid 和 name，Order 表也有两个属性，即 uid 和 orderid，它们的连接键为公共属性 uid。这里对两个表执行连接操作，得到用户的订单号与用户名的对应关系，具体的 SQL 语句命令如下：

```
select name, orderid from user u join order o on u.uid=o.uid;
```

图 14-6 描述了连接操作转化为 MapReduce 任务的具体执行过程。首先，在 Map 阶段，User 表以 uid 为键（key），以 name 和表的标记位（这里 User 的标记位记为 1）为值（value）进行 Map 操作，把表中记录转化成生成一系列键值对的形式。同样地， Order 表以 uid 为键，以 orderid 和表的标记位（这里表 Order 的标记位记为 2）为值进行 Map 操作，把表中记录转化成生成一系列键值对的形式。比如，User 表中记录(1,Lily)转化为键值对(1,<1,Lily>)，其中，括号中的第一个“1”是 uid 的值，第二个“1”是表 User 的标记位，用来标识这个键值对来自 User 表；再比如，Order 表中记录(1,101)转化为键值对(1,<2,101>)，其中，“2”是表 Order 的标记位，用来标识这个键值对来自 Oder 表。接着，在 Shuffle 阶段，把 User 表和 Order 表生成的键值对按键值进行哈希，然后传送给对应的 Reduce 机器执行，比如键值对(1,<1,Lily>)、(1,<2,101>)和 (1,<2,102>)传送到同一台 Reduce 机器上，键值对(2,<1,Tom>)和(2,<2,103>)传送到另一台 Reduce 机器上。当 Reduce 机器接收这些键值对时，还需要按表的标记位对这些键值对进行排序，以优化连接操作。最后，在 Reduce 阶段，对同一台 Reduce 机器上的键值对，根据“值”（value）中的表标记位，对来自 User 和 Order 这两个表的数据进行笛卡尔积连接操作，以生成最终的连接结果。比如，键值对(1,<1,Lily>)与键值对(1,<2,101>)和 (1,<2,102>)的连接结果分别为(Lily ,101>)和 (Lily, 102)，键值对(2,<1,Tom>)和键值对(2,<2,103>)的连接结果为(Tom, 103)。

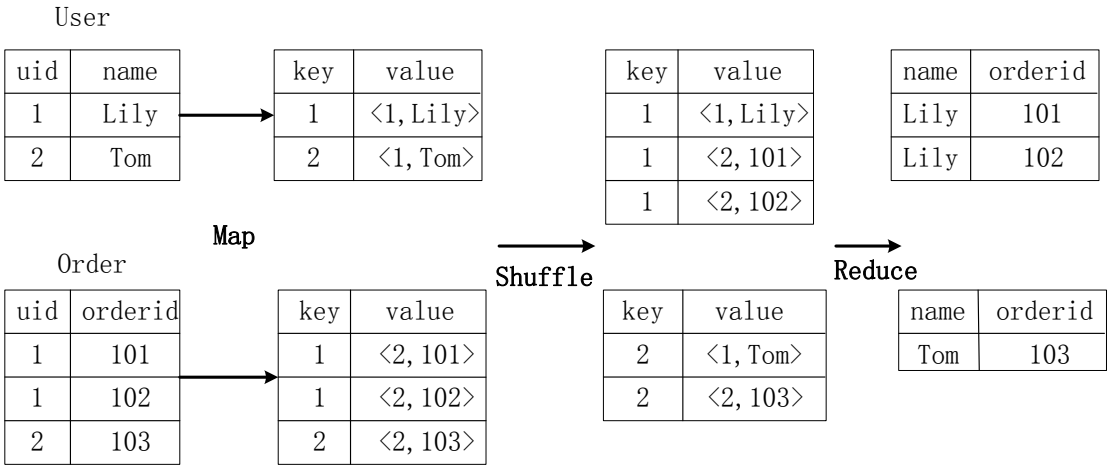


图 14-6 用 MapReduce 实现连接操作的基本原理

（2）用 MapReduce 实现分组操作

假设分数表 Score 具有两个属性，即 rank（排名）和 level（级别），这里存在一个分组（Group By）操作，其功能是把表 Score 的不同片段按照 rank 和 level 的组合值进行合并，计算不同 rank 和 level 的组合值分别有几条记录。具体的 SQL 语句命令如下：

```
select rank, level ,count(*) as value from score group by rank, level;
```

图 14-7 描述了分组操作转化为 MapReduce 任务的具体执行过程。首先，在 Map 阶段，对表 Score 进行 Map 操作，生成一系列键值对，对于每个键值对，其键为“<rank,level>”，值为“拥有该<rank,value>组合值的记录的条数”。比如，Score 表的第一片段中有两条记录(A,1)，所以，记录(A,1)转化为键值对(<A,1>,2)，Score 表的第二片段中只有一条记录(A,1)，所以，记录(A,1)转化为键值对(<A,1>,1)。接着，在 Shuffle 阶段，对 Score 表生成的键值对，按照“键”的值进行哈希，然后根据哈希结果传送给对应的 Reduce 机器去执行，比如键值对(<A,1>,2)和 (<A,1>,1)传送到同一台 Reduce 机器上，键值对(<B,2>,1)传送到另一台 Reduce 机器上。然后，Reduce 机器对接收到的这些键值对，按“键”的值进行排序。最后，在 Reduce 阶段，对于 Reduce 机器上的这些键值对，把具有相同键的所有键值对的“值”进行累加，生成分组的最终结果，比如，在同一台 Reduce 机器上的键值对(<A,1>,2)和 (<A,1>,1>)Reduce 后的输出结果为(A,1,3)，(<B,2>,1)的 Reduce 后的输出结果为(B,2,1)。

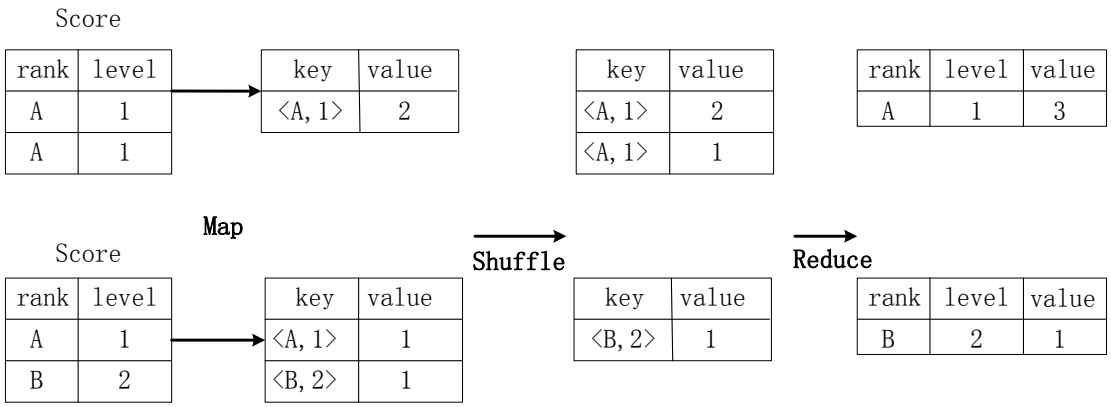


图 14-7 用 MapReduce 实现分组操作的实现原理

14.3.2 Hive 中 SQL 查询转换成 MapReduce 作业的过程

当用户向 Hive 输入一段命令或查询（即 HiveQL 语句）时，Hive 需要与 Hadoop 交互工作来完成该操作。该命令或查询首先进入到驱动模块，由驱动模块中的编译器进行解析编译，并由优化器对该操作进行优化计算，然后交给执行器去执行。执行器通常的任务是启动一个或多个 MapReduce 任务，有时也不需要启动 MapReduce 任务，比如，执行包含*的操作时（如 select * from 表），就是全表扫描，选择所有的属性和所有的元组，不存在投影和选择操作，因此，不需要执行 Map 和 Reduce 操作。图 14-8 描述了用户提交一段 SQL 查询后，Hive 把 SQL 语句转化成 MapReduce 任务进行执行的详细过程。

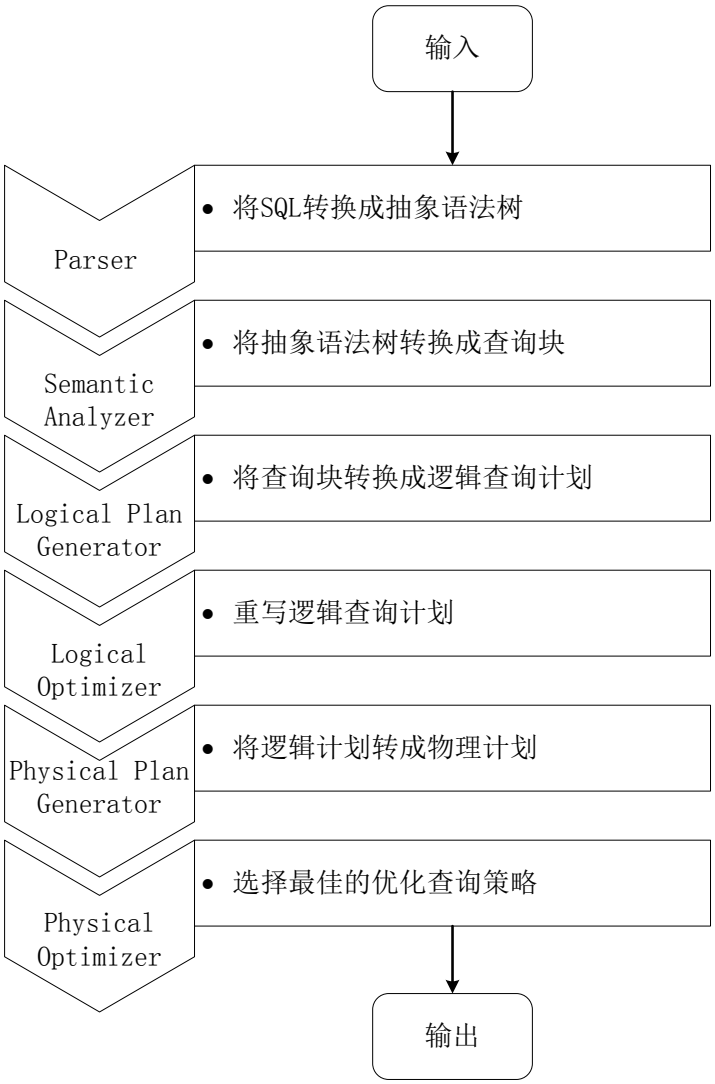


图 14-8 Hive 中 SQL 查询的 MapReduce 作业转化过程

如图 14-7 所示，在 Hive 中，用户通过命令行 CLI 或其他 Hive 访问工具，向 Hive 输入一段命令或查询以后，SQL 查询被 Hive 自动转化为 MapReduce 作业，具体步骤如下：

第 1 步：由 Hive 驱动模块中的编译器——Antlr 语言识别工具，对用户输入的 SQL 语言进行词法和语法解析，将 SQL 语句转化为抽象语法树（AST Tree）的形式；

第 2 步：对该抽象语法树进行遍历，进一步转化成 QueryBlock 查询单元。因为抽象语法树的结构仍很复杂，不方便直接翻译为 MapReduce 算法程序，所以，Hive 把抽象语法树进一步转化为 QueryBlock，其中，QueryBlock 是一条最基本的 SQL 语法组成单元，包括输入源、计算过程和输出三个部分；

第 3 步：再对 QueryBlock 进行遍历，生成 OperatorTree（操作树）。其中，OperatorTree 由很多逻辑操作符组成，如 TableScanOperator、SelectOperator、FilterOperator、JoinOperator、GroupByOperator 和 ReduceSinkOperator 等。这些逻辑操作符可以在 Map 阶段和 Reduce 阶段完成某一特定操作；

第 4 步：通过 Hive 驱动模块中的逻辑优化器对 OperatorTree 进行优化，变换 OperatorTree 的形式，合并多余的操作符，从而减少 MapReduce 任务数量以及 Shuffle 阶段的数据量；

第 5 步：对优化后的 OperatorTree 进行遍历，根据 OperatorTree 中的逻辑操作符生成需要执行的 MapReduce 任务；

第 6 步：启动 Hive 驱动模块中的物理优化器，对生成的 MapReduce 任务进行优化，生成最终的 MapReduce 任务执行计划；

第 7 步：最后由 Hive 驱动模块中的执行器，对最终的 MapReduce 任务进行执行输出。

需要说明的是，Hive 驱动模块中的执行器执行最终的 MapReduce 任务时，Hive 本身是不会生成 MapReduce 算法程序的，它需要通过一个表示“Job 执行计划”的 XML 文件，来驱动执行内置的、原生的 Mapper 和 Reducer 模块。Hive 通过和 JobTracker 通信来初始化 MapReduce 任务，而不需要直接部署在 JobTracker 所在的管理节点上执行。通常在大型集群上，会有专门的网关机来部署 Hive 工具。这些网关机的作用主要是远程操作和管理节点上的 JobTracker 通信来执行任务。Hive 要处理的数据文件通常存储在 HDFS 上，HDFS 由名称节点（NameNode）来管理。

14.4 Hive HA 基本原理

Hive 的功能十分强大，可以支持采用 SQL 方式查询 Hadoop 平台上的数据，但是，在实际应用中，Hive 也暴露出不稳定的问题，在极少数情况下，甚至会出现端口不响应或者进程丢失的问题。Hive HA（High Availability）的出现，就是为了解决这类问题。

如图 14-9 所示，在 Hive HA 中，在 Hadoop 集群上构建的数据仓库是由多个 Hive 实例进行管理的，这些 Hive 实例被纳入到一个资源池中，并由 HAProxy 提供一个统一的对外接口。客户端的查询请求首先访问 HAProxy，由 HAProxy 对访问请求进行转发。HAProxy 收到请求后，会轮询资源池里可用的 Hive 实例，执行逻辑可用性测试，如果某个 Hive 实例逻辑可用，就会把客户端的访问请求转发到该 Hive 实例上，如果该 Hive 实例逻辑不可用，就把它放入黑名单，并继续从资源池中取出下一个 Hive 实例进行逻辑可用性测试。对于黑名单中的 Hive 实例，HiveHA 会每隔一段时间进行统一处理，首先尝试重启该 Hive 实例，如果重启成功，就再次把它放入到资源池中。由于采用 HAProxy 提供统一的对外访问接口，因此，对于程序开发人员来说，可以把它认为是一台超强“Hive"。

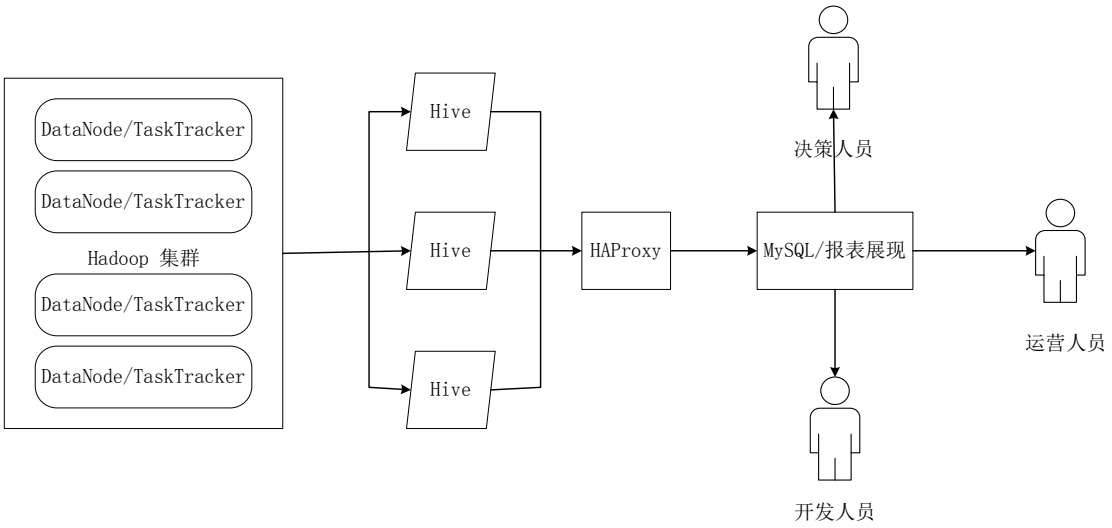


图 14-9 Hive HA 基本原理

14.5 Impala

Hive 作为现有比较流行的数据仓库分析工具之一，得到了广泛的应用，但是由于 Hive 采用 MapReduce 来完成批量数据处理，因此，实时性不好，查询延迟较高。Impala 作为新一代开源大数据分析引擎，支持实时计算，它提供了与 Hive 类似的功能，并在性能上比 Hive 高出 3~30 倍。Impala 发展势头迅猛，甚至有可能会超过 Hive 的使用率而成为 Hadoop 上最流行的实时计算平台。

14.5.1 Impala 简介

Impala 是由 Cloudera 公司开发的新型查询系统，它提供 SQL 语义，能查询存储在 Hadoop 的 HDFS 和 HBase 上的 PB 级别海量数据。Hive 虽然也提供了 SQL 语义，但是 Hive 底层执行任务最终仍然需要借助于 MapReduce，而 MapReduce 是一个面向批处理的非实时计算框架，不能满足查询的实时交互性。Impala 最初是参照 Dremel 系统进行设计的，Dremel 系统是由 Google 公司开发的交互式数据分析系统，可以在 2-3 秒内分析 PB 级别的海量数据。所以，Impala 也可以实现大数据的快速查询。

需要指出的是，虽然 Impala 的实时查询性能要比 Hive 好很多，但是，Impala 的目的并不在于替换现有的包括 Hive 在内的 MapReduce 工具，而是提供一个统一的平台用于实时查询。事实上，Impala 的运行依然需要依赖于 Hive 的元数据。总体而言，Impala 与其它组件之间的关系如图 14-10 所示。

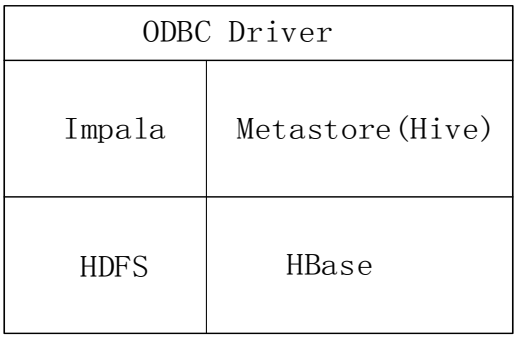


图 14-10 Impala 与其他组件的关系

与 Hive 类似，Impala 也可以直接与 HDFS 和 HBase 进行交互。Hive 底层执行使用的是 MapReduce，所以主要用于处理长时间运行的批处理任务，例如批量提取、转化、加载类型的任务。而 Impala 则采用了与商用并行关系数据库类似的分布式查询引擎，可以直接从 HDFS 或者 HBase 中用 SQL 语句查询数据，而不需要把 SQL 语句转化成 MapReduce 任务来执行，从而大大降低了延迟，可以很好地满足实时查询的要求。另外，Impala 和 Hive 采用相同的 SQL 语法、ODBC 驱动程序和用户接口。

14.5.2 Impala 系统架构

Impala 的系统架构如图 14-11 所示，图中的虚线模块是属于 Impala 的组件，从图中可以看出，Impala 和 Hive、HDFS、HBase 等工具是统一部署在一个 Hadoop 平台上的。Impala 主要由 Impalad、State Store 和 CLI 三部分组成，具体如下：

● Impalad：Impalad 是 Impala 的一个进程，负责协调客户端提交的查询的执行，给其他 Impalad 分配任务以及收集其他 Impalad 的执行结果进行汇总。另外，Impalad 也会执行其他 Impalad 给其分配的任务，主要就是对本地 HDFS 和 HBase 里的部分数据进行操作。Impalad 进程主要包含 Query Planner、Query Coordinator 和 Query Exec Engine 三个模块，与 HDFS 的数据节点（HDFS DN）运行在同一节点上，并且完全分布运行在 MPP（大规模并行处理系统）架构上。

● State Store：负责收集分布在集群中各个 Impalad 进程的资源信息，从而用于查询的调度。State Store 会创建一个 statestored 进程，来跟踪集群中的 Impalad 的健康状态及位置信息。statestored 进程通过创建多个线程来处理 Impalad 的注册订阅以及与各个 Impalad 保持心跳连接，另外，各 Impalad 都会缓存一份 State Store 中的信息。当 State Store 离线后，Impalad 一旦发现 State Store 处于离线状态时，就会进入恢复模式，并进行反复注册。当 State Store 重新加入集群后，自动恢复正常，更新缓存数据。

● CLI：CLI 给用户提供了执行查询的命令行工具，同时，Impala 还提供了 Hue、JDBC 及 ODBC 使用接口。

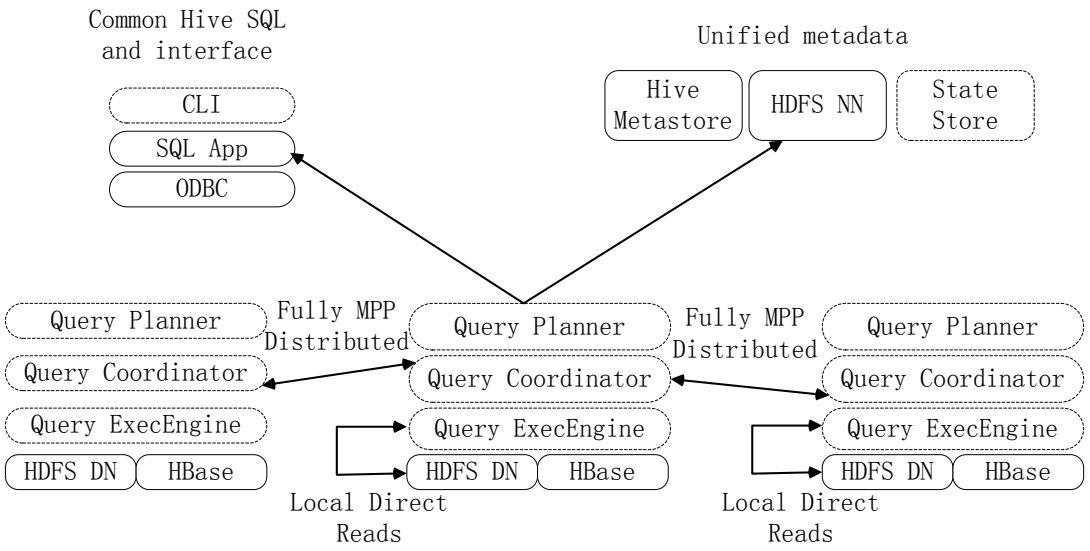


图 14-11 Impala 系统架构

Impala 中的元数据直接存储在 Hive 中。Impala 采用与 Hive 相同的元数据、SQL 语法、ODBC 驱动程序和用户接口，从而使得在一个 Hadoop 平台上，可以统一部署 Hive 和 Impala 等分析工具，同时支持批处理和实时查询。

14.5.3 Impala 查询执行过程

如图 14-12 所示，Impala 查询的执行过程具体如下：

第 0 步：注册和订阅。当用户提交查询前，Impala 先创建一个 Impalad 进程来具体负责协调客户端提交的查询，该进程会向 State Store 提交注册订阅信息，State Store 会创建一个 statestored 进程，statestored 进程通过创建多个线程来处理 Impalad 的注册订阅信息。

第 1 步：提交查询。用户通过 CLI 客户端提交一个查询到 Impalad 进程，Impalad 的 Query Planner 对 SQL 语句进行解析，生成解析树；然后，Planner 把这个查询的解析树变成若干 PlanFragment，发送到 Query Coordinator，其中，PlanFragment 由 PlanNode 组成的，能被分发到单独的节点上执行，每个 PlanNode 表示一个关系操作和对其执行优化需要的信息。

第 2 步：获取元数据与数据地址。Query Coordinator 从 MySQL 元数据库中获取元数据（即查询需要用到哪些数据），从 HDFS 的名称节点中获取数据地址（即数据被保存在哪个数据节点上），从而得到存储这个查询相关数据的所有数据节点。

第 3 步：分发查询任务。Query Coordinator 初始化相应 Impalad 上的任务，即把查询任务分配给所有存储这个查询相关数据的数据节点。

第 4 步：汇聚结果。Query Executor 通过流式交换中间输出，并由 Query Coordinator 汇聚来自各个 Impalad 的结果。

第 5 步：返回结果。Query Coordinator 把汇总后的结果返回给 CLI 客户端。

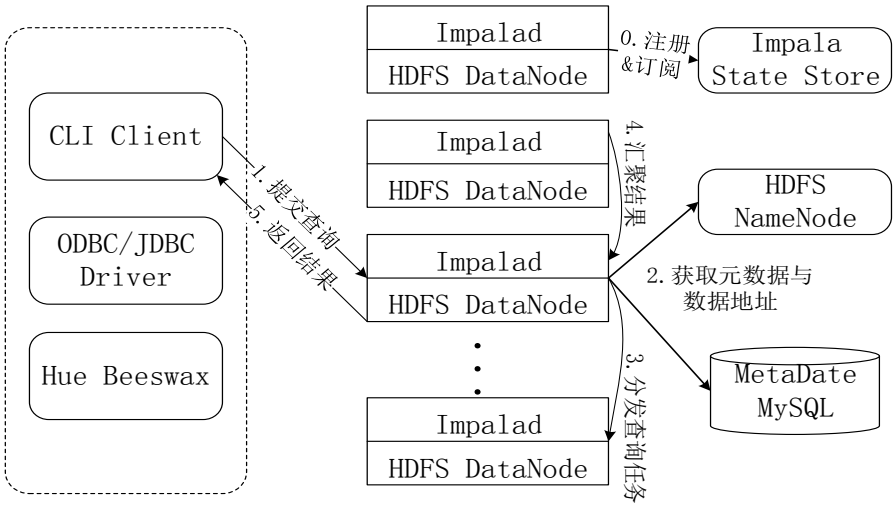


图 14-12 Impala 查询执行过程

14.5.4 Impala 与 Hive 的比较

Impala 作为新一代开源大数据分析引擎，与现在比较流行的 Hive 相比，既有相同点，又有不同点，它们的区别与联系可以通过图 14-13 进行展现。

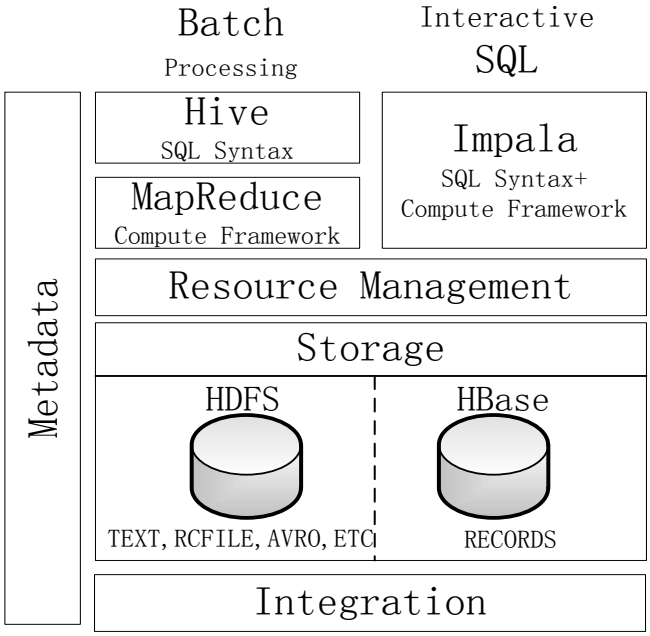


图 14-13 Impala 与 Hive 的对比

Hive 与 Impala 的不同点总结如下：第一，Hive 比较适合进行长时间的批处理查询分析，而 Impala 适合进行实时交互式 SQL 查询。第二，Hive 依赖于 MapReduce 计算框架，执行计划组合成管道型的 MapReduce 任务模式进行执行，而 Impala 则把执行计划表现为一棵完

整的执行计划树，可以更自然地分发执行计划到各个 Impalad 执行查询。第三，Hive 在执行过程中，如果内存放不下所有数据，则会使用外存，以保证查询能顺序执行完成，而 Impala 在遇到内存放不下数据时，不会利用外存，所以，Impala 目前处理查询时会受到一定的限制。

Hive 与 Impala 的相同点总结如下：第一，Hive 与 Impala 使用相同的存储数据池，都支持把数据存储于 HDFS 和 HBase 中，其中，HDFS 支持存储 TEXT、RCFILE、PARQUET、AVRO、ETC 等格式的数据，HBase 存储表中记录。第二，Hive 与 Impala 使用相同的元数据。第三，Hive 与 Impala 中对 SQL 的解释处理比较相似，都是通过词法分析生成执行计划。

总的来说，Impala 的目的不在于替换现有的 MapReduce 工具，把 Hive 与 Impala 配合使用效果最佳，可以先使用 Hive 进行数据转换处理，之后再使用 Impala 在 Hive 处理后的结果数据集上进行快速的数据分析。

14.6 Hive 编程实践

本节首先介绍 Hive 的数据类型，然后介绍 Hive 的基本操作，最后，给出一个 WordCount 应用实例，并简单分析 Hive 与 MapReduce 在执行 WordCount 时的区别。

14.6.1 Hive 的数据类型

Hive 支持关系数据库中的大多数基本数据类型，同时 Hive 还支持关系数据库中不常出现的的 3 种集合数据类型。表 14-2 中列举了 Hive 所支持的基本数据类型，包括多种不同长度的整型和浮点型数据类型、布尔类型以及无长度限制的字符串类型。另外，新版本（Hive v0.8.0 以上）中还支持时间戳数据类型和二进制数组数据类型。表 14-3 列举了 Hive 中的列所支持的 3 种集合数据类型：struct、map、array。这里需要注意的是，表 14-3 的示例实际上调用的是内置函数。

表 14-2 Hive 的基本数据类型

类型	描述	示例
TINYINT	1 个字节（8 位）有符号整数	1
SMALLINT	2 个字节（16 位）有符号整数	1
INT	4 个字节（32 位）有符号整数	1
BIGINT	8 个字节（64 位）有符号整数	1
FLOAT	4 个字节（32 位）单精度浮点数	1.0
DOUBLE	8 个字节（64 位）双精度浮点数	1.0
BOOLEAN	布尔类型，true/false	true
STRING	字符串，可以指定字符集	“xmu”
TIMESTAMP	整数、浮点数或者字符串	1327882394（Unix 新纪元秒）

BINARY	字节数组	[0,1,0,1,0,1,0,1]

表 14-3 Hive 的集合数据类型

类型	描述	示例
ARRAY	一组有序字段，字段的类型必须相同	Array(1,2)
MAP	一组无序的键/值对，键的类型必须是原子的，值可以是任何数据类型，同一个映射的键和值的类型必须相同	Map(‘a’,1,’b’,2)
STRUCT	一组命名的字段，字段类型可以不同	Struct(‘a’,1,1,0)

14.6.2 Hive 基本操作

HiveQL 是 Hive 的查询语言，和 SQL 语言比较类似，对 Hive 的操作都是通过编写 HiveQL 语句来实现的，接下来介绍一下 Hive 中常用的几个基本操作。

1.create: 创建数据库、表、视图

（1）创建数据库

① 创建数据库 hive

```
hive> create database hive;
```

② 创建数据库 hive，因为 hive 已经存在，所以会抛出异常，加上 if not exists 关键字，则不会抛出异常

```
hive> create database if not exists hive;
```

（2）创建表

① 在 hive 数据库中，创建表 usr，含三个属性 id，name，age

```
hive> use hive;
hive>create table if not exists usr(id bigint,name string,age int);
```

② 在 hive 数据库中，创建表 usr，含三个属性 id，name，age，存储路径为“/usr/local/hive/warehouse/hive/usr”

```
hive>create table if not exists hive.usr(id bigint,name string,age int)
    >location ‘/usr/local/hive/warehouse/hive/usr’;
```

③ 在 hive 数据库中，创建外部表 usr，含三个属性 id，name，age，可以读取路径“/usr/local/data”下以“,”分隔的数据。

```
hive>create external table if not exists hive.usr(id bigint,name string,age int)
```

>row format delimited fields terminated by ','

Location '/usr/local/data';

④ 在 hive 数据库中，创建分区表 usr，含三个属性 id，name，age，还存在分区字段 sex。

hive>create table hive.usr(id bigint,name string,age int) partition by(sex boolean);

⑤ 在 hive 数据库中，创建分区表 usr1，它通过复制表 usr 得到。

hive> use hive;

hive>create table if not exists usr1 like usr;

（3）创建视图

① 创建视图 little_usr，只包含 usr 表中 id，age 属性

hive>create view little_usr as select id,age from usr;

2. drop：删除数据库、表、视图

（1）删除数据库

① 删除数据库 hive，如果不存在会出现警告

hive> drop database hive;

② 删除数据库 hive，因为有 if exists 关键字，即使不存在也不会抛出异常

hive>drop database if not exists hive;

③ 删除数据库 hive，加上 cascade 关键字，可以删除当前数据库和该数据库中的表

hive> drop database if not exists hive cascade;

（2）删除表

① 删除表 usr，如果是内部表，元数据和实际数据都会被删除；如果是外部表，只删除元数据，不删除实际数据

hive> drop table if exists usr;

（3）删除视图

① 删除视图 little_usr

hive> drop view if exists little_usr;

3. alter：修改数据库、表、视图

（1）修改数据库

① 为 hive 数据库设置 dbproperties 键值对属性值来描述数据库属性信息

```
hive> alter database hive set dbproperties('edited-by'='lily');
```

（2）修改表

① 重命名表 usr 为 user

```
hive> alter table usr rename to user;
```

② 为表 usr 增加新分区

```
hive> alter table usr add if not exists partition(age=10);
hive> alter table usr add if not exists partition(age=20);
```

③ 删除表 usr 中分区

```
hive> alter table usr drop if exists partition(age=10);
```

④ 把表 usr 中列名 name 修改为 username，并把该列置于 age 列后

```
hive>alter table usr change name username string after age;
```

⑤ 在对表 usr 分区字段之前，增加一个新列 sex

```
hive>alter table usr add columns(sex boolean);
```

⑥ 删除表 usr 中所有字段并重新指定新字段 newid，newname，newage

```
hive>alter table usr replace columns(newid bigint,newname string,newage int);
```

⑦ 为 usr 表设置 tblproperties 键值对属性值来描述表的属性信息

```
hive> alter table usr set tabproperties('notes'='the columns in usr may be null except id');
```

（3）修改视图

① 修改 little_usr 视图元数据中的 tblproperties 属性信息

```
hive> alter view little_usr set tabproperties('create_at'='refer to timestamp');
```

4. show：查看数据库、表、视图

（1）查看数据库

① 查看 Hive 中包含的所有数据库

```
hive> show databases;
```

② 查看 Hive 中以 h 开头的所有数据库

```
hive>show databases like 'h.*';
```

（2）查看表和视图

① 查看数据库 hive 中所有表和视图

hive> use hive;

hive> show tables;

② 查看数据库 hive 中以 u 开头的所有表和视图

hive> show tables in hive like ‘u.*’;

5. describe：描述数据库、表、视图

（1）描述数据库

① 查看数据库 hive 的基本信息，包括数据库中文件位置信息等

hive> describe database hive;

② 查看数据库 hive 的详细信息，包括数据库的基本信息及属性信息等

hive>describe database extended hive;

（2）描述表和视图

① 查看表 usr 和视图 little_usr 的基本信息，包括列信息等

hive> describe hive.usr;

hive> describe hive.little_usr;

② 查看表 usr 和视图 little_usr 的详细信息，包括列信息、位置信息、属性信息等

hive> describe extended hive.usr;

hive> describe extended hive.little_usr;

③ 查看表 usr 中列 id 的信息

hive> describe extended hive.usr.id;

6. load：向表中装载数据

① 把目录’/usr/local/data’下的数据文件中的数据装载进 usr 表并覆盖原有数据

hive> load data local inpath ‘/usr/local/data’ overwrite into table usr;

② 把目录’/usr/local/data’下的数据文件中的数据装载进 usr 表不覆盖原有数据

hive> load data local inpath ‘/usr/local/data’ into table usr;

③ 把分布式文件系统目录’hdfs://master_server/usr/local/data’下的数据文件数据装载进 usr 表并覆盖原有数据

hive> load data inpath ‘hdfs://master_server/usr/local/data’ overwrite into table usr;

7. select：查询表中数据

该命令和 SQL 语句完全相同这里不再赘述。

8. insert：向表中插入数据或从表中导出数据

① 向表 usr1 中插入来自 usr 表的数据并覆盖原有数据

hive> insert overwrite table usr1

> select * from usr where age=10;

② 向表 usr1 中插入来自 usr 表的数据并追加在原有数据后

hive> insert into table usr1

> select * from usr where age=10;

14.6.3 Hive 应用实例：WordCount

现在我们通过一个实例——词频统计，来深入学习一下 Hive 的具体使用。首先，需要创建一个需要分析的输入数据文件，然后编写 HiveQL 语句实现 WordCount 算法，在 Linux 下实现步骤如下：

（1）创建 input 目录，其中 input 为输入目录。命令如下：

$ cd /usr/local/hadoop

$ mkdir input

（2）在 input 文件夹中创建两个测试文件 file1.txt 和 file2.txt，命令如下：

$ cd /usr/local/hadoop/input

$ echo "hello world" > file1.txt

$ echo "hello hadoop" > file2.txt

（3）进入 hive 命令行界面，编写 HiveQL 语句实现 WordCount 算法，命令如下：

$ hive

hive> create table docs(line string);

hive> load data inpath 'input' overwrite into table docs;

```
hive>create table word_count as
    select word, count(1) as count from
    (select explode(split(line,' '))as word from docs) w
    group by word
    order by word;
```

执行完成后，用 select 语句查看运行结果，如图 5-1 所示：

```
OK
Time taken: 2.662 seconds
hive> select * from word_count;
OK
hadoop  1
hello   2
world   1
Time taken: 0.043 seconds, Fetched: 3 row(s)
```

图 5-1 WordCount 算法统计结果查询

14.6.4 Hive 编程的优势

词频统计算法是最能体现 MapReduce 思想的算法之一，因此，这里以 WordCount 实例为例，简单比较一下其在 MapReduce 中的编程实现和在 Hive 中编程实现的不同点。首先，采用 Hive 实现 WordCount 算法需要编写较少的代码量。在 MapReduce 中，WordCount 类由 63 行 Java 代码编写而成（该代码可以通过下载 Hadoop 源码后，在以下目录：%HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar 包中找到），而在 Hive 中只需要编写 7 行代码。其次，在 MapReduce 的实现中，需要进行编译生成 jar 文件来执行算法，而在 Hive 中则不需要，虽然 HiveQL 语句的最终实现需要转换为 MapReduce 任务来执行，但是这些都是由 Hive 框架自动完成的，用户不需要了解具体实现细节。

由上可知，采用 Hive 实现最大的优势是，对于非程序员，不用学习编写复杂的 Java MapReduce 代码了，只需要用户学习使用简单的 HiveQL 就可以了，而这对于有 SQL 基础的用户而言是非常容易的。

本章小结

本章详细介绍了 Hive 的基本知识。Hive 是一个构建于 Hadoop 顶层的数据仓库工具，主要用于对存储在 Hadoop 文件中的数据集进行数据整理、特殊查询和分析处理。Hive 在某种程度上可以看作是用户编程接口，本身不存储和处理数据，依赖 HDFS 存储数据，依赖 MapReduce 处理数据。

Hive 支持使用自身提供的命令行 CLI、简单网页 HWI 访问方式以及通过 Karmasphere、Hue、Qubole 等工具的外部访问。

Hive 在数据仓库中的具体应用中，主要用于报表中心的报表分析统计上。在 Hadoop 集群上构建的数据仓库由多个 Hive 进行管理，具体实现采用 Hive HA 原理的方式，实现一台超强“Hive"。

Impala 作为新一代开源大数据分析引擎，支持实时计算，在性能上比 Hive 高出 3~30 倍，甚至可能会超过 Hive 的使用率而成为 Hadoop 上最流行的实时计算平台。

本章最后以单词统计为例，详细介绍了如何使用 Hive 进行简单编程。

习题

1.试述在 Hadoop 生态系统中 Hive 与其他组件之间的相互关系。

2.请比较 Hive 与传统数据库之间的异同点。

3.请简述 Hive 的几种访问方式。

4.请分别对 Hive 的几个主要组成模块进行简要介绍。

5.请简述向 Hive 中输入一条查询的具体执行过程。

6.请简述 Hive HA 原理。

7.请简述 Impalad 进程的主要作用。

8.请比较 Hive 与 Impala 的异同点。

9.请简述 State Store 的作用。

10.请简述 Impala 执行一条查询的具体过程。

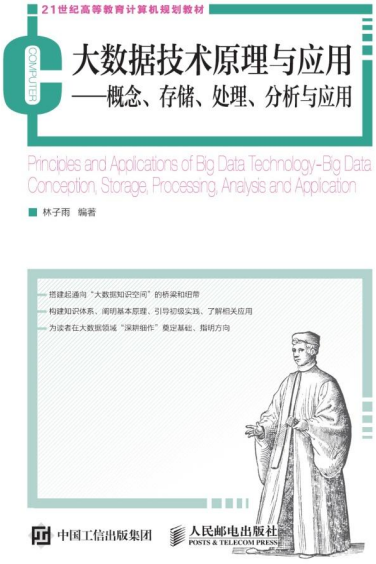
11.请列举 Hive 所支持的 3 种集合数据类型。

12.请列举几个 Hive 的常用操作及基本语法。

附录 1:任课教师介绍

林子雨(1978－),男,博士,厦门大学计算机科学系助理教授,主要研究领域为数据库,实时主动数据仓库,数据挖掘.
主讲课程：《大数据技术基础》
办公地点：厦门大学海韵园科研 2 号楼
E-mail: ziyulin@xmu.edu.cn
个人主页：http://www.cs.xmu.edu.cn/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn

附录 2：课程教材介绍



《大数据技术原理与应用——概念、存储、处理、分析与应用》，由厦门大学计算机科学系教师林子雨博士编著，是中国高校第一本系统介绍大数据知识的专业教材。本书定位为大数据技术入门教材，为读者搭建起通向“大数据知识空间”的桥梁和纽带，以“构建知识体系、阐明基本原理、引导初级实践、了解相关应用”为原则，为读者在大数据领域“深耕细作”奠定基础、指明方向。

全书共有 13 章，系统地论述了大数据的基本概念、大数据处理架构 Hadoop、分布式文件系统 HDFS、分布式数据 库 HBase、NoSQL 数据库、云数据库、分布式并行编程模型 MapReduce、流计算、图计算、数据可视化以及大数据在互联网、生物医学和物流等各个领域的应用。在 Hadoop、HDFS、HBase 和 MapReduce 等重要章节，安排了入门级的实践操作，让读者更好地学习和掌握大数据关键技术。

本书可以作为高等院校计算机专业、信息管理等相关专业的大数据课程教材，也可供相关技术人员参考、学习、培训之用。

欢迎访问《大数据技术原理与应用——概念、存储、处理、分析与应用》教材官方网站：http://dblab.xmu.edu.cn/post/bigdata

扫一扫访问教材官网

附录 3：中国高校大数据课程公共服务平台介绍

中国高校大数据课程公共服务平台，由中国高校首个“数字教师”的提出者和建设者——林子雨老师发起，由厦门大学数据库实验室全力打造，由厦门大学云计算与大数据研究中心、海峡云计算与大数据应用研究中心携手共建。这是国内第一个服务于高校大数据课程建设的公共服务平台，旨在促进国内高校大数据课程体系建设，提高大数据课程教学水平，降低大数据课程学习门槛，提升学生课程学习效果。平台服务对象涵盖高校、教师和学生。平台为高校开设大数据课程提供全流程辅助，为教师开展教学工作提供一站式服务，为学生学习大数据课程提供全方位辅导。平台重点打造“9 个 1 工程”，即 1 本教材（含官网）、1 个教师服务站、1 个学生服务站、1 个公益项目、1 堂巡讲公开课、1 个示范班级、1 门在线课程、1 个交流群（QQ 群、微信群）和 1 个保障团队。

平台主页：http://dblab.xmu.edu.cn/post/bigdata-teaching-platform/

扫一扫访问平台主页