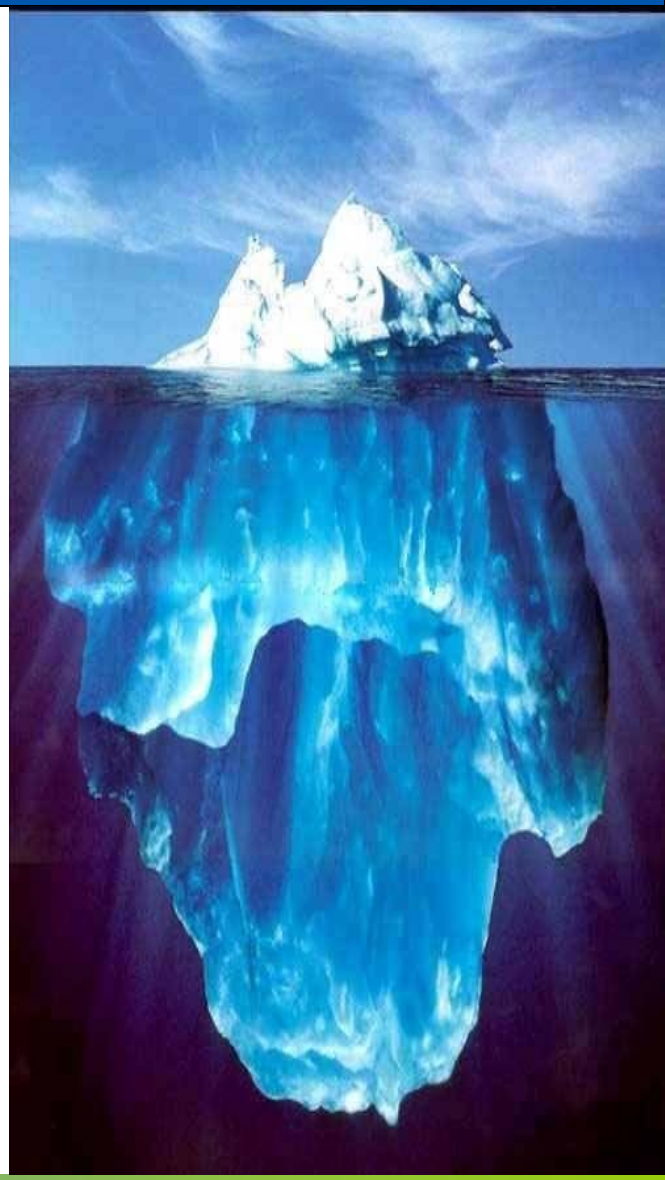
《大数据技术与应用》

第九章 基于Hadoop的数据仓库Hive

提纲

- 9.1 概述
- 9.2 Hive系统架构
- 9.3 Hive工作原理
- 9.4 Hive HA基本原理
- 9.5 Impala
- 9.6 Hive编程实践



9.1 概述

- 9.1.1 数据仓库概念
- 9.1.2 传统数据仓库面临的挑战
- 9.1.3 Hive简介
- 9.1.4 Hive与Hadoop生态系统中其他组件的关系
- 9.1.5 Hive与传统数据库的对比分析
- 9.1.6 Hive在企业中的部署和应用

9.1.1数据仓库概念

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。

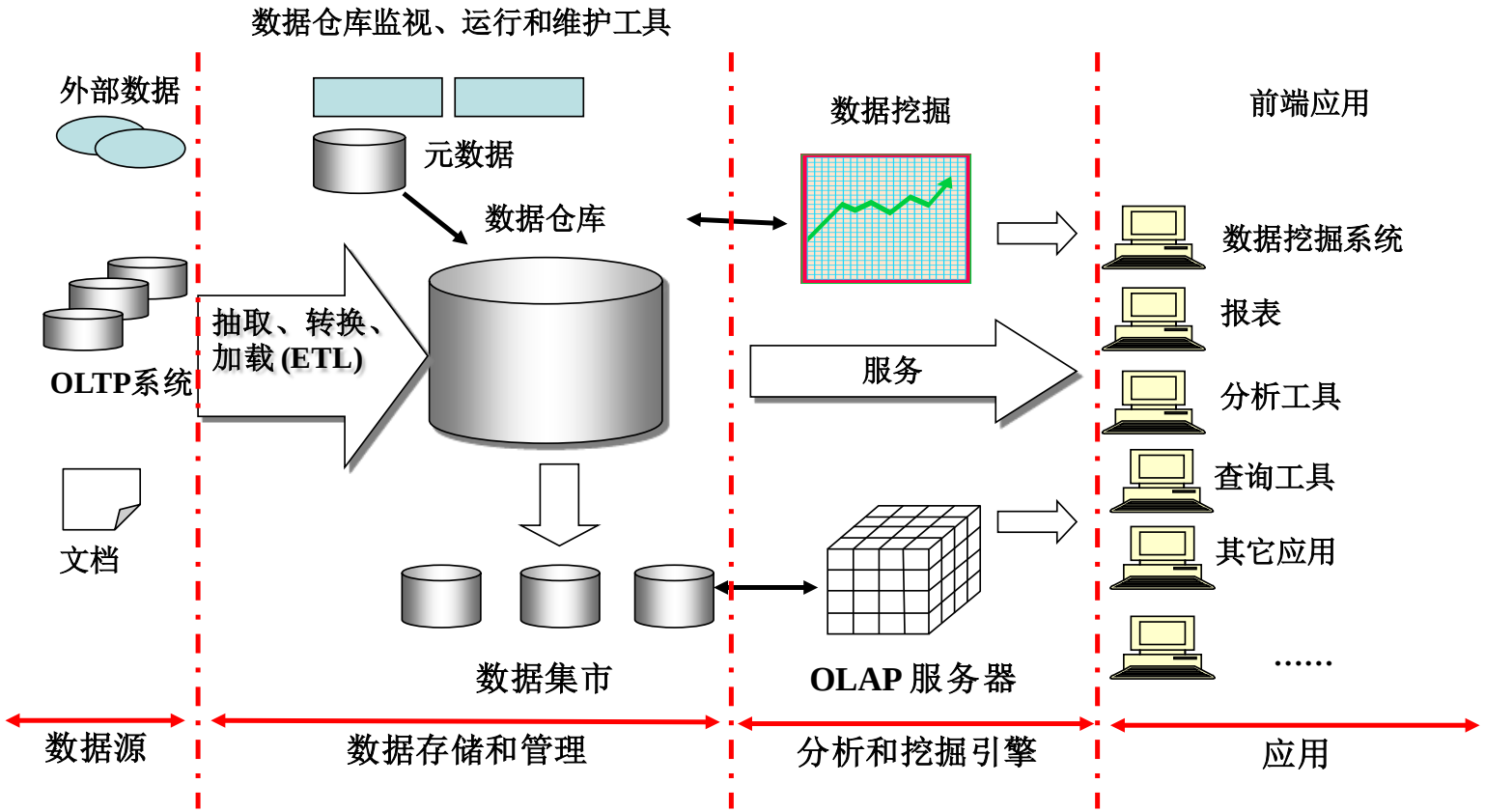


图9-1 数据仓库的体系结构

9.1.2 传统数据仓库面临的挑战

- （1）无法满足快速增长的海量数据存储需求
- （2）无法有效处理不同类型的数据
- （3）计算和处理能力不足

9.1.3 Hive简介

•Hive是一个构建于Hadoop顶层的数据仓库工具
•支持大规模数据存储、分析，具有良好的可扩展性
•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
•依赖分布式文件系统HDFS存储数据
•依赖分布式并行计算模型MapReduce处理数据
•定义了简单的类似SQL 的查询语言——HiveQL
•用户可以通过编写的HiveQL语句运行MapReduce任务
•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
•是一个可以提供有效、合理、直观组织和使用数据的分析工具

9.1.3 Hive简介

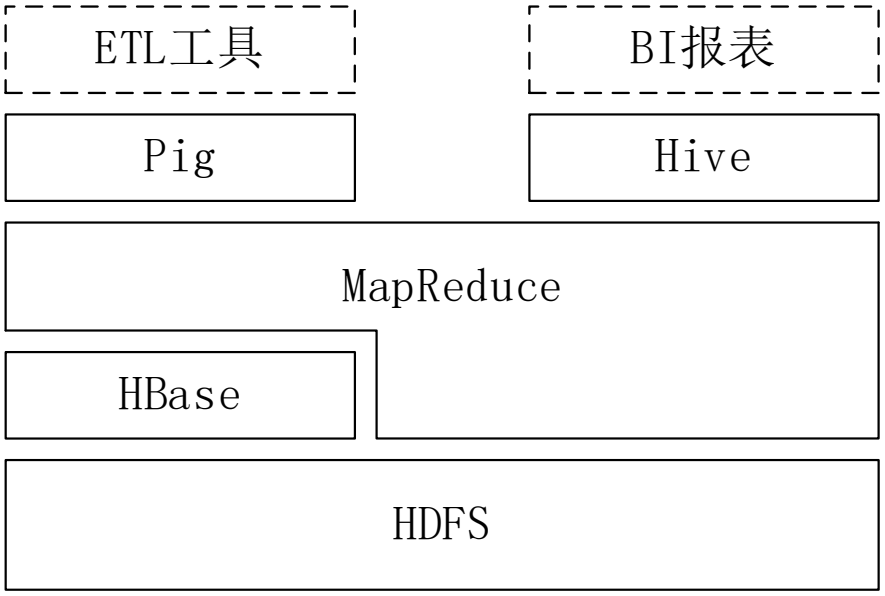
Hive具有的特点非常适用于数据仓库

- **采用批处理方式处理海量数据**
  - Hive需要把HiveQL语句转换成MapReduce任务进行运行
  - 数据仓库存储的是静态数据，对静态数据的分析适合采用批处理方式，不需要快速响应给出结果，而且数据本身也不会频繁变化
- **提供适合数据仓库操作的工具**
  - Hive本身提供了一系列对数据进行提取、转换、加载（ETL）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

9.1.4 Hive与Hadoop生态系统中其他组件的关系

- **Hive依赖于HDFS 存储数据**
- **Hive依赖于MapReduce 处理数据**
- **在某些场景下Pig可以作为Hive的替代工具**
- **HBase 提供数据的实时访问**

Hadoop生态系统



9.1.5 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

对比项目	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

9.1.6 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

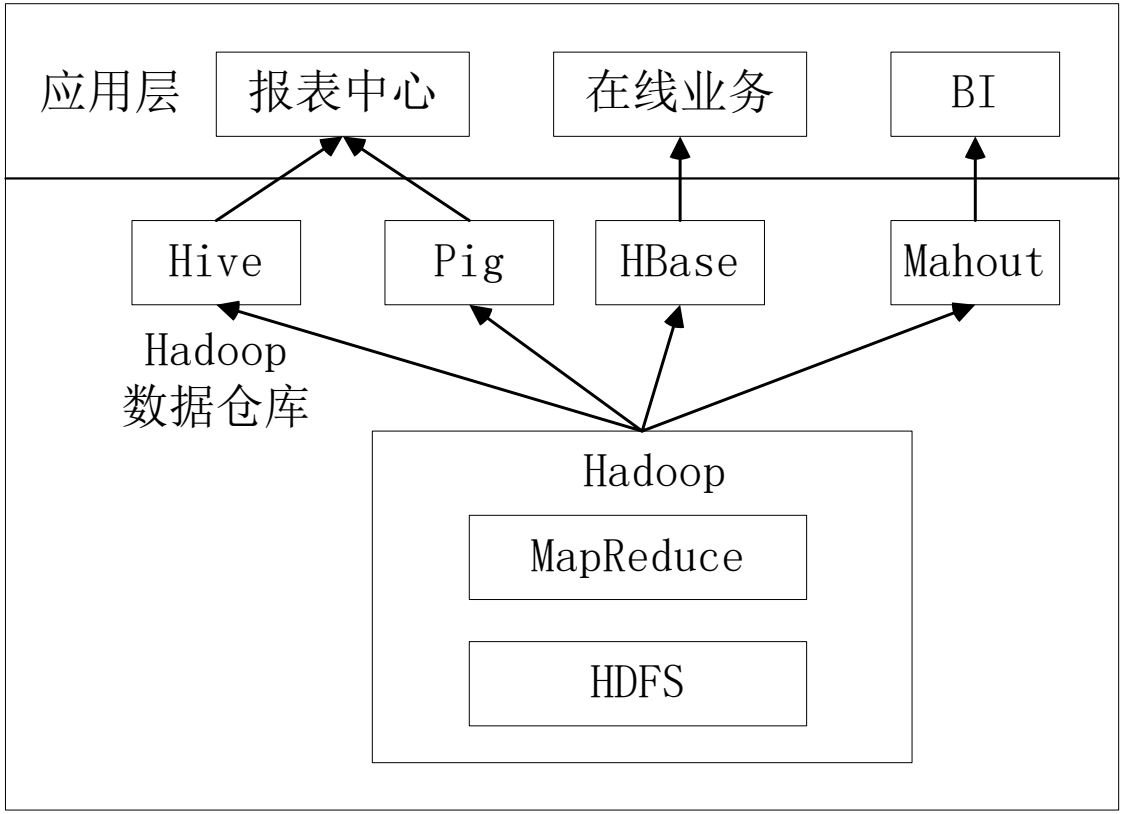


图 企业中一种常见的大数据分析平台部署框架

9.1.6 Hive在企业中的部署和应用

2.Hive在Facebook公司中的应用

- 基于Oracle的数据仓库系统已经无法满足激增的业务需求
- 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

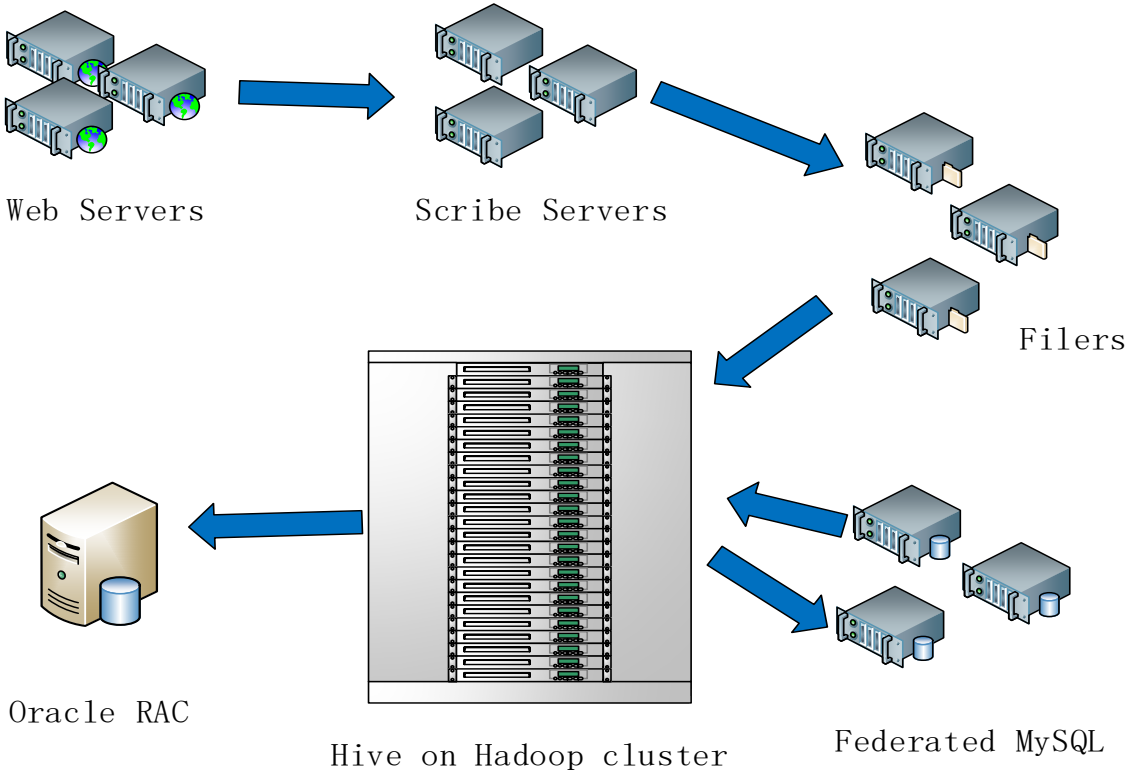


图 Facebook的数据仓库架构

9.2 Hive系统架构

- 用户接口模块包括CLI、HWI、JDBC、ODBC、Thrift Server

- 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

- 元数据存储模块（Metastore）是一个独立的关系型数据库（自带derby数据库，或MySQL数据库）

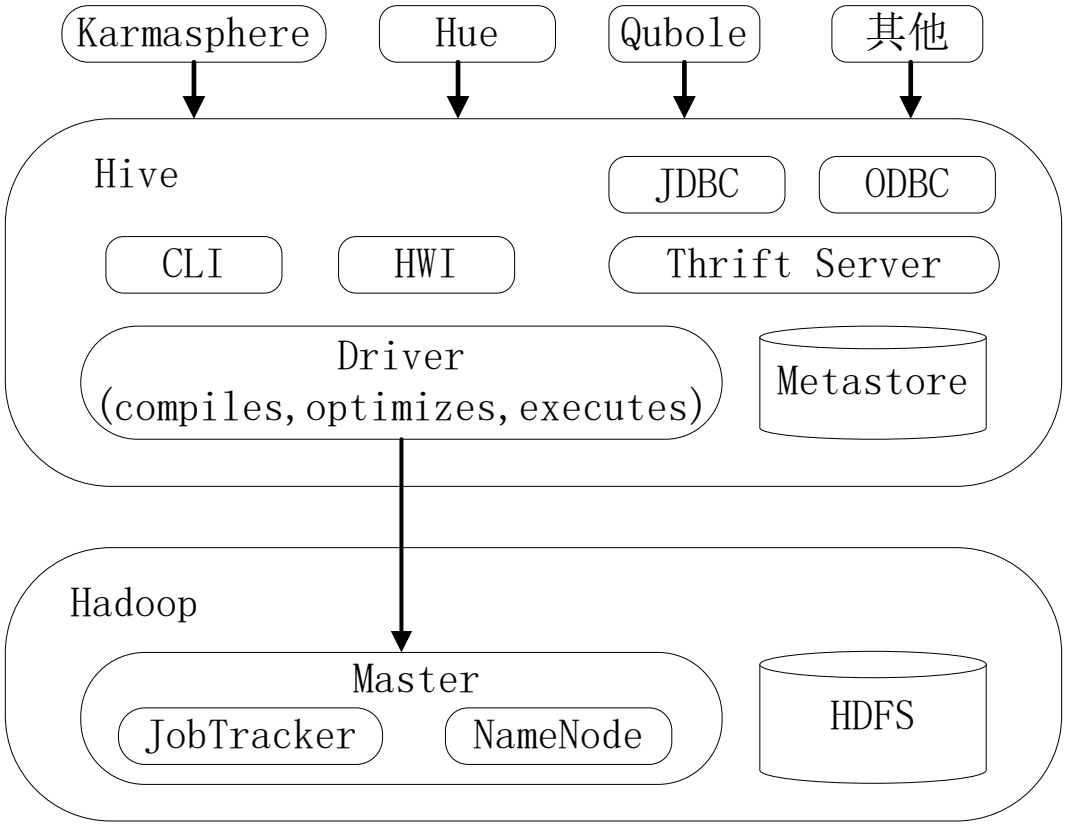


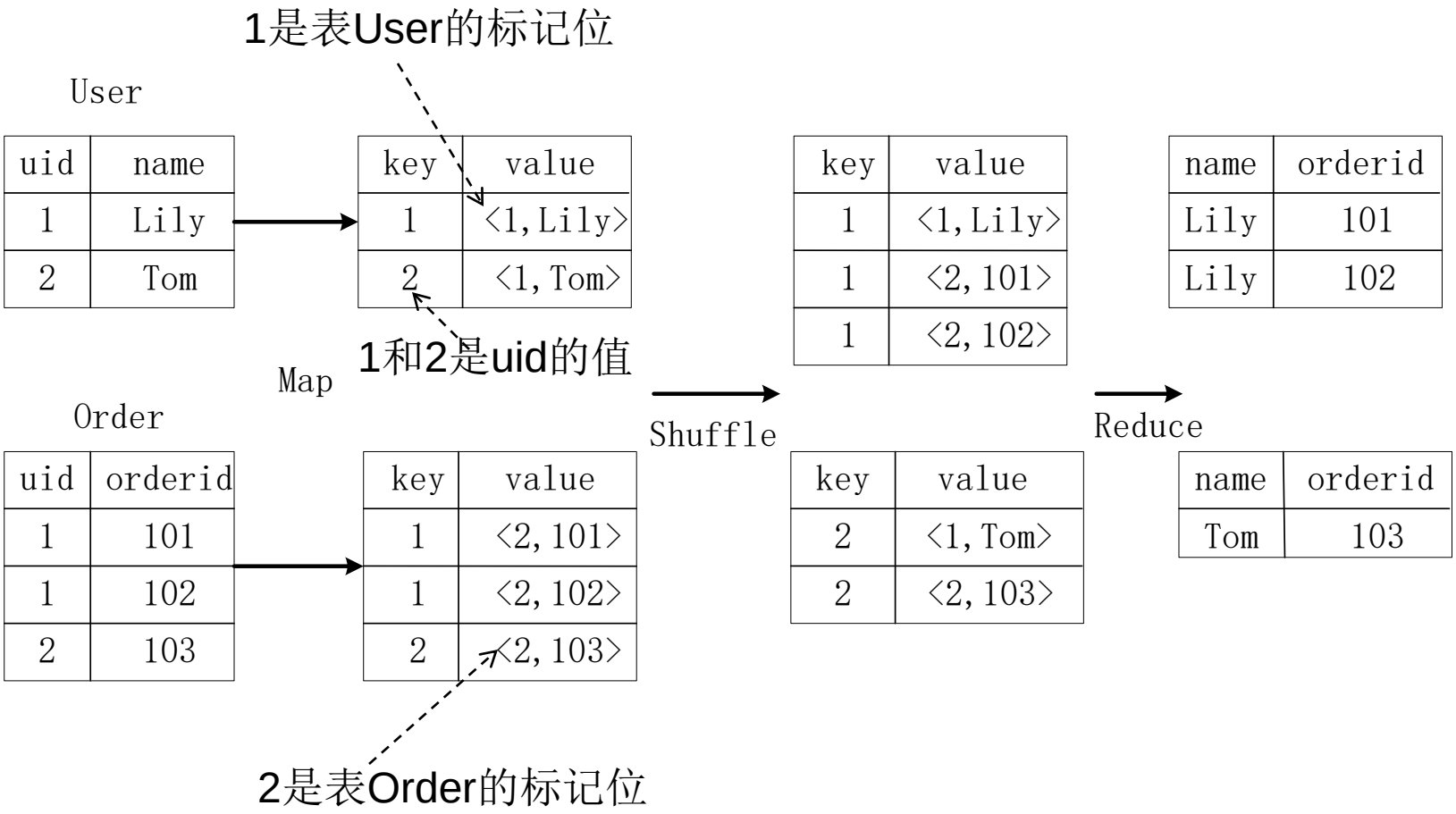
图 Hive系统架构

9.3 Hive工作原理

- 9.3.1 SQL语句转换成MapReduce作业的基本原理
- 9.3.2 Hive中SQL查询转换成MapReduce作业的过程

9.3.1 SQL语句转换成MapReduce的基本原理

1.join的实现原理

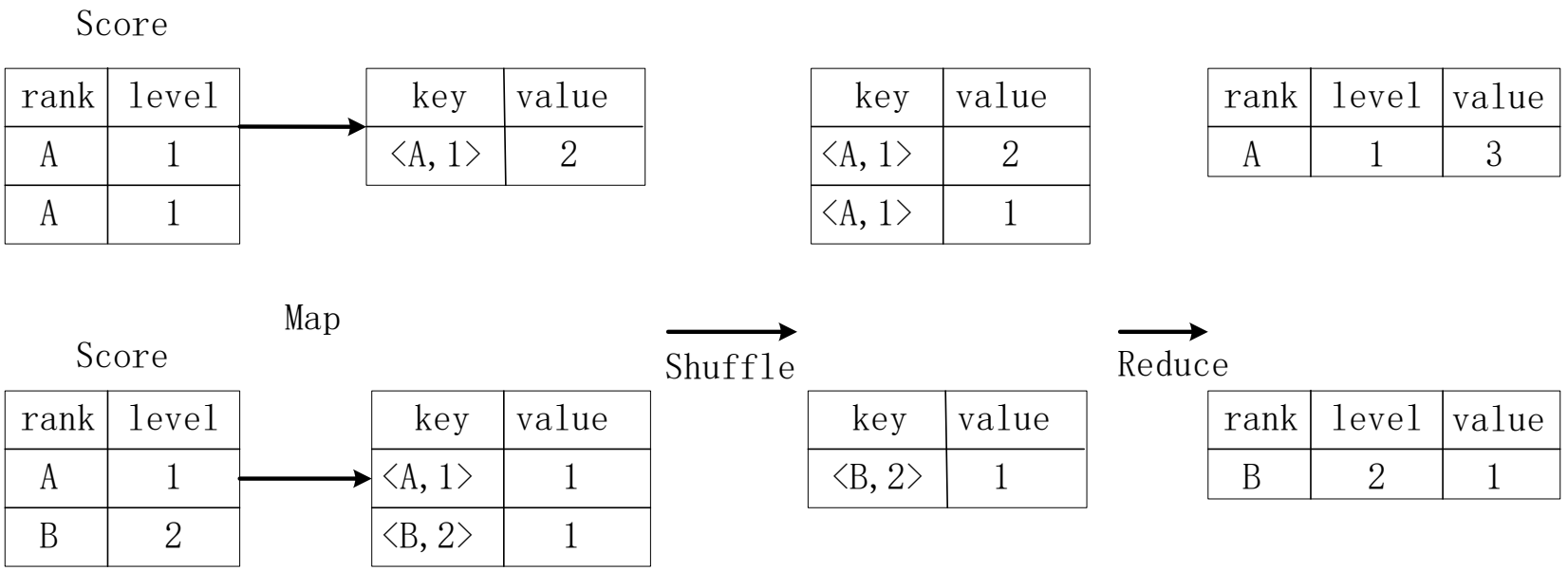


9.3.1 SQL语句转换成MapReduce的基本原理

2. group by的实现原理

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：

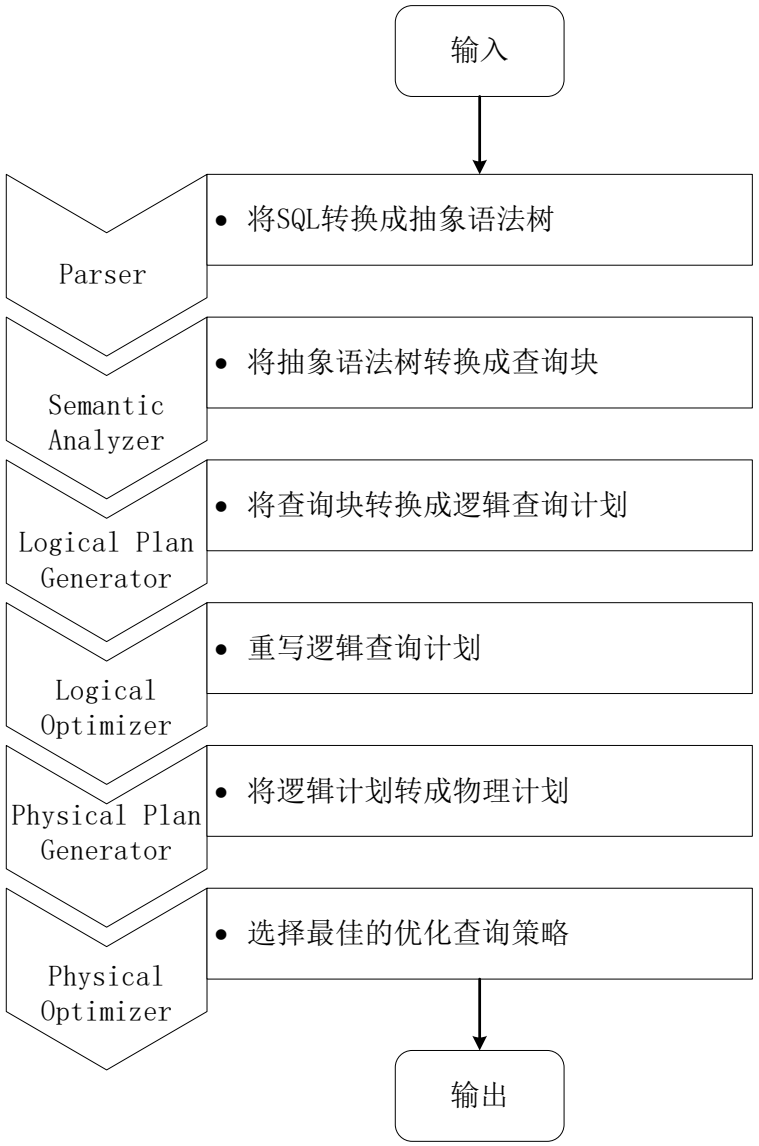
select rank, level ,count(*) as value from score group by rank, level



9.3.2 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时，Hive需要与Hadoop交互工作来完成该操作：
  - 驱动模块接收该命令或查询编译器
  - 对该命令或查询进行解析编译
  - 由优化器对该命令或查询进行优化计算
  - 该命令或查询通过执行器进行执行

9.3.2 Hive中SQL查询转换成MapReduce作业的过程



第1步：由Hive驱动模块中的编译器对用户输入的SQL语言进行词法和语法解析，将SQL语句转化为抽象语法树的形式

第2步：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，因此，把抽象语法书转化为查询块

第3步：把查询块转换成逻辑查询计划，里面包含了许多逻辑操作符

第4步：重写逻辑查询计划，进行优化，合并多余操作，减少MapReduce任务数量

第5步：将逻辑操作符转换成需要执行的具体MapReduce任务

第6步：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：由Hive驱动模块中的执行器，对最终的MapReduce任务进行执行输出

9.3.2 Hive中SQL查询转换成MapReduce作业的过程

几点说明：

- 当启动MapReduce程序时，Hive本身是不会生成MapReduce算法程序的
- 需要通过一个表示“Job执行计划”的XML文件驱动执行内置的、原生的Mapper和Reducer模块
- Hive通过和JobTracker通信来初始化MapReduce任务，不必直接部署在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。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
- 数据文件通常存储在HDFS上，HDFS由名称节点管理

9.4 Hive HA基本原理

问题：在实际应用中，Hive也暴露出不稳定的问题

解决方案：Hive HA（High Availability）

- 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。
- 对于程序开发人员来说，可以把它认为是一台超强“Hive”。

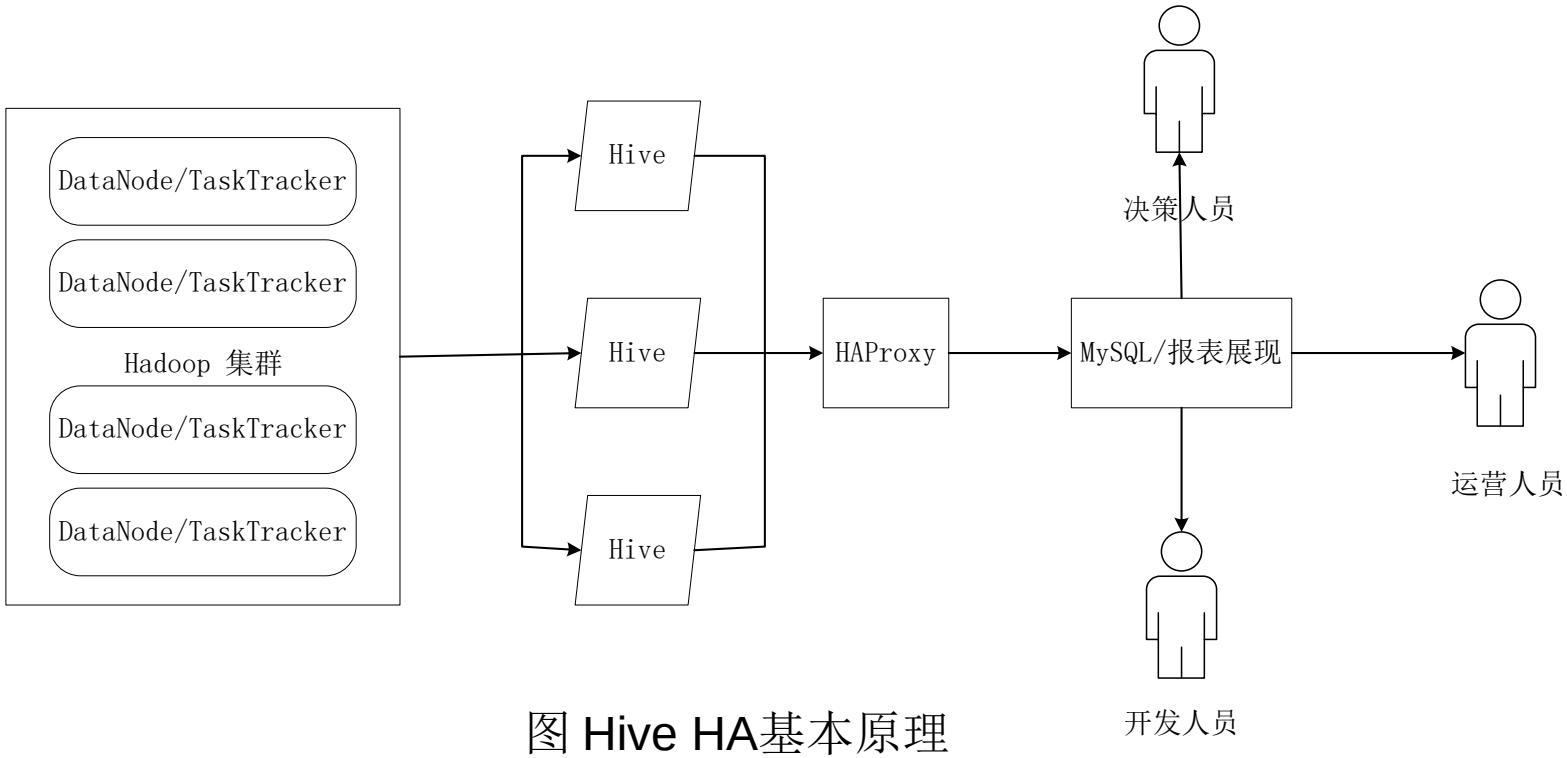


图 Hive HA基本原理

9.5 Impala

9.5.1 Impala简介

9.5.2 Impala系统架构

9.5.3 Impala查询执行过程

9.5.4 Impala与Hive的比较

9.5.1 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级大数据，在性能上比Hive高出3~30倍**
- **Impala的运行需要依赖于Hive的元数据**
- **Impala是参照 Dremel系统进行设计的**
- **Impala采用了与商用并行关系数据库类似的分布式查询引擎，可以直接与HDFS和HBase进行交互查询**
- **Impala和Hive采用相同的SQL语法、ODBC驱动程序和用户接口**

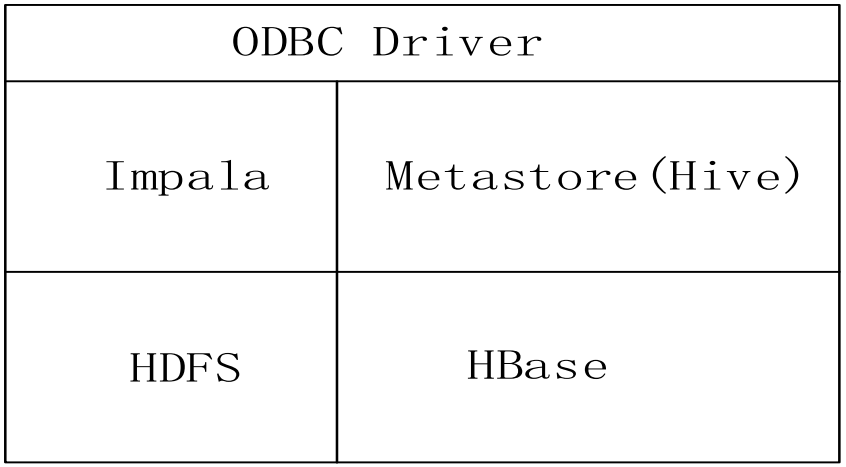


图 Impala与其他组件关系

9.5.2 Impala系统架构

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

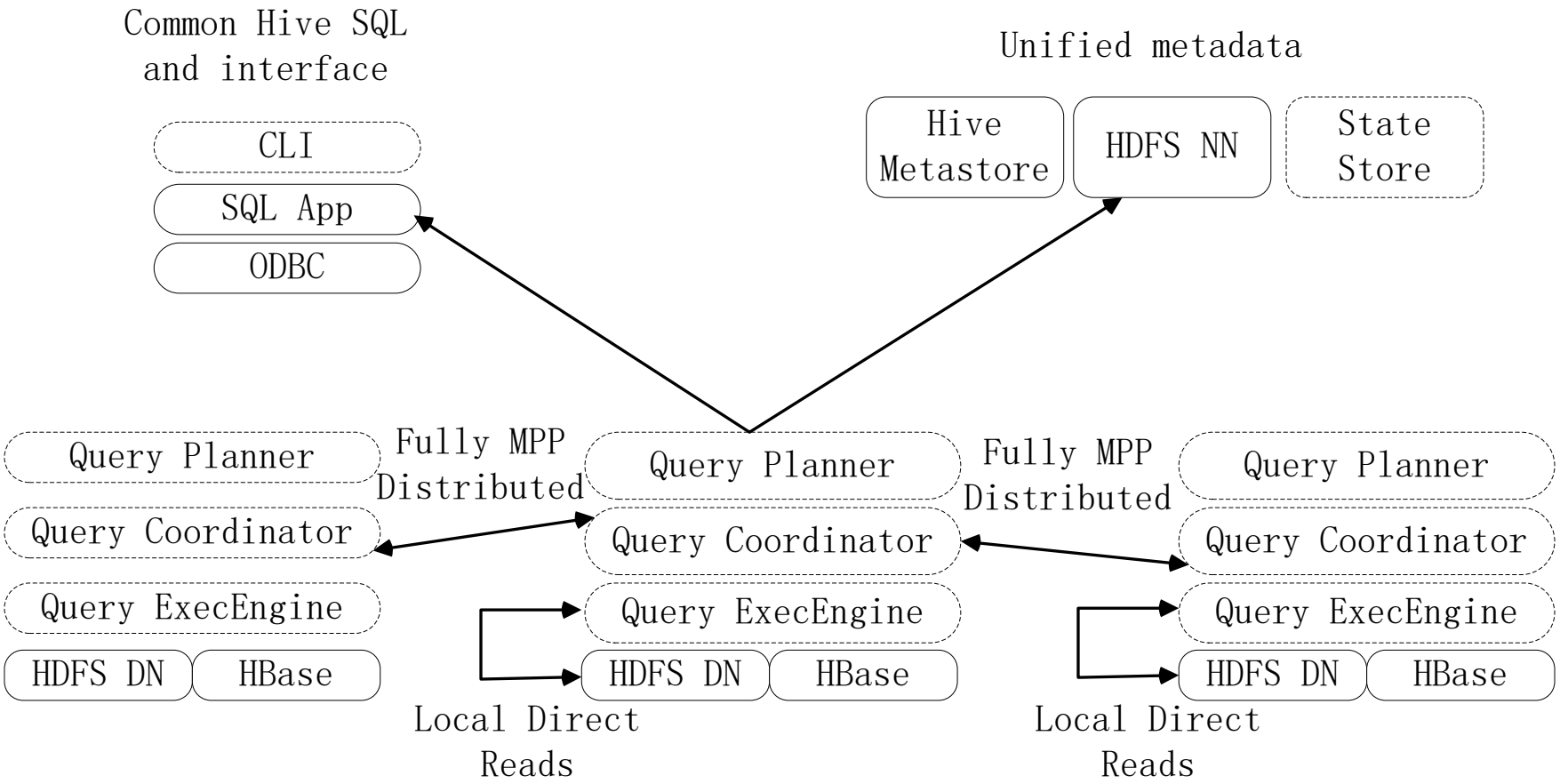


图　Impala系统架构

9.5.2 Impala系统架构

Impala主要由Impalad，State Store和CLI三部分组成

1. Impalad

- 负责协调客户端提交的查询的执行
- 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
- 给其他Impalad分配任务以及收集其他Impalad的执行结果进行汇总
- 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

•2. State Store：会创建一个statestored进程；负责收集分布在集群中各个Impalad进程的资源信息，用于查询调度。

3. CLI：给用户提供查询使用的命令行工具；还提供了Hue、JDBC及ODBC的接口

说明：**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**

9.5.3 Impala查询执行过程

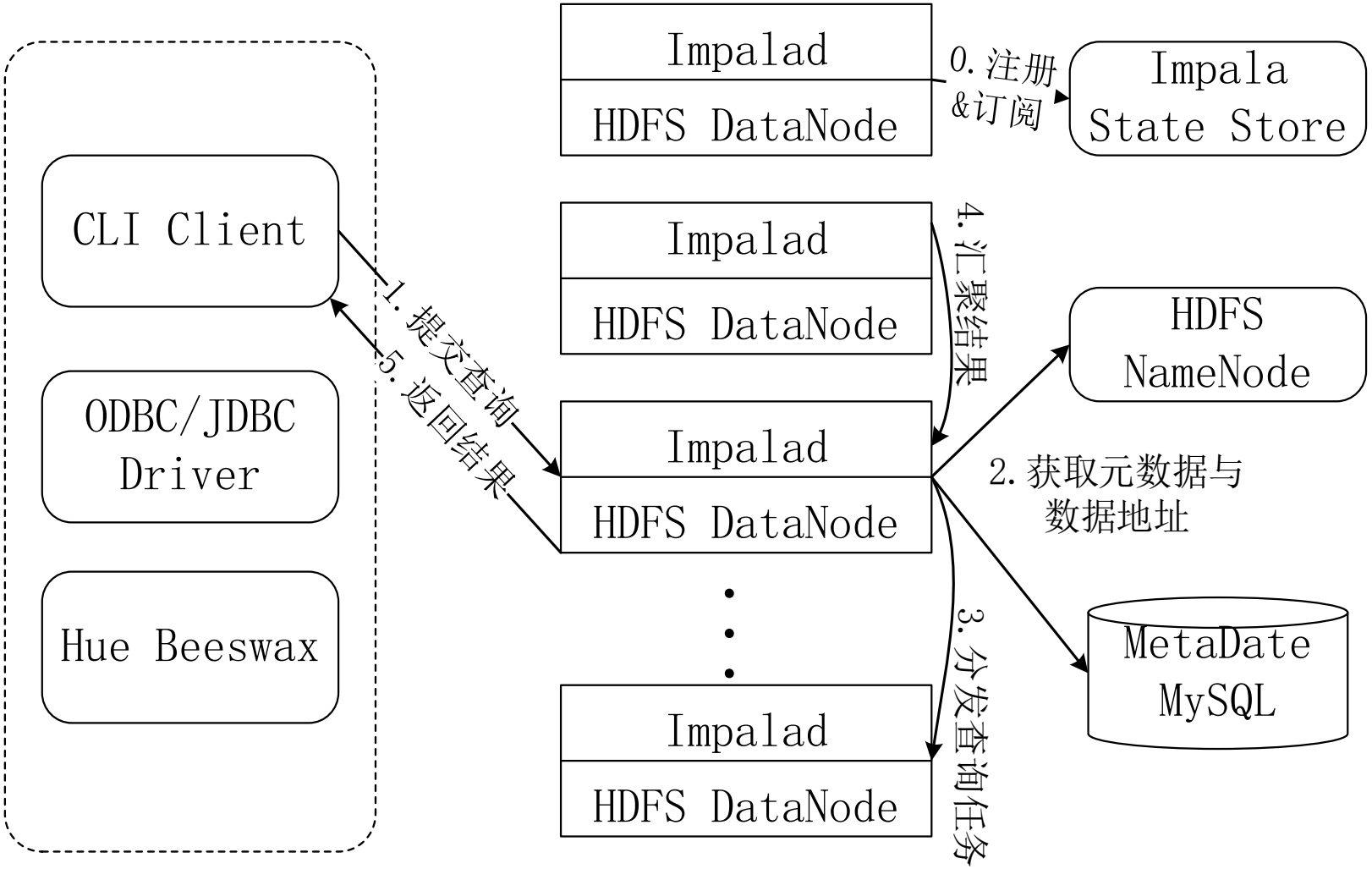


图 Impala查询过程图

9.5.4 Impala与Hive的比较

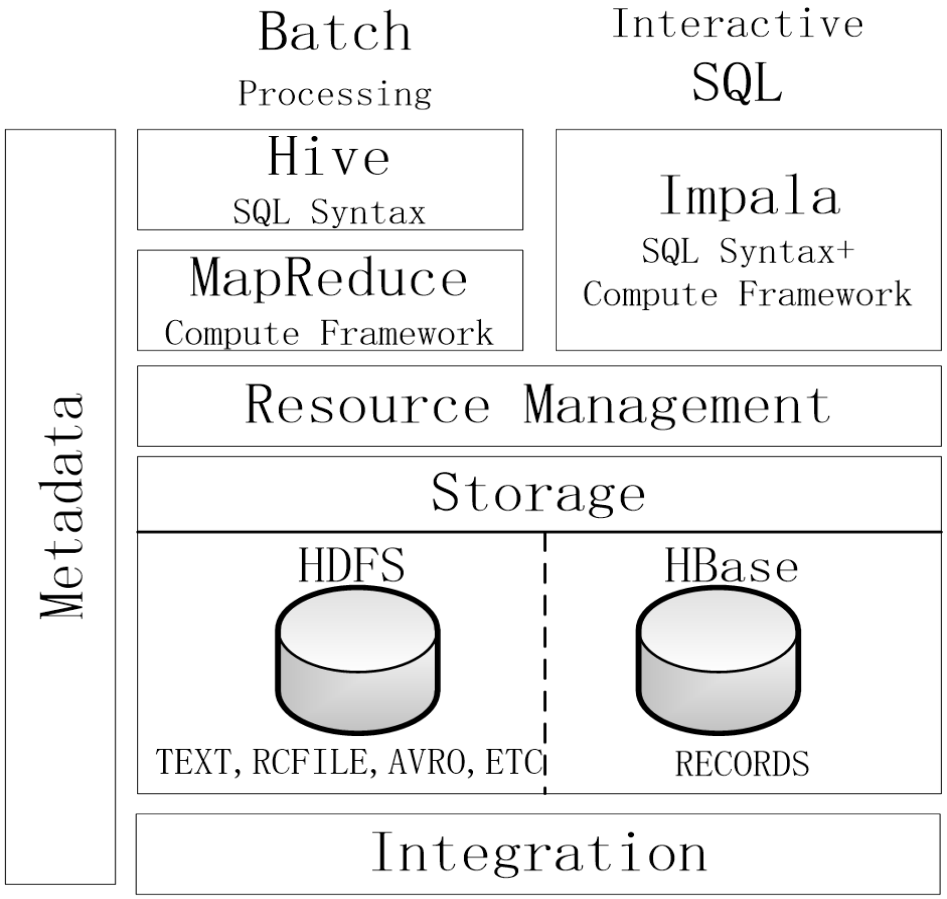


图　Impala与Hive的对比

Hive与Impala的不同点总结如下：

1. Hive适合长时间的批处理查询分析，而Impala适合实时交互式SQL查询

2. Hive依赖于MapReduce计算框架，Impala把执行计划表现为一棵完整的执行计划树，直接分发执行计划到各个Impalad执行查询

3. 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处理查询时会受到一定的限制

9.5.4 Impala与Hive的比较

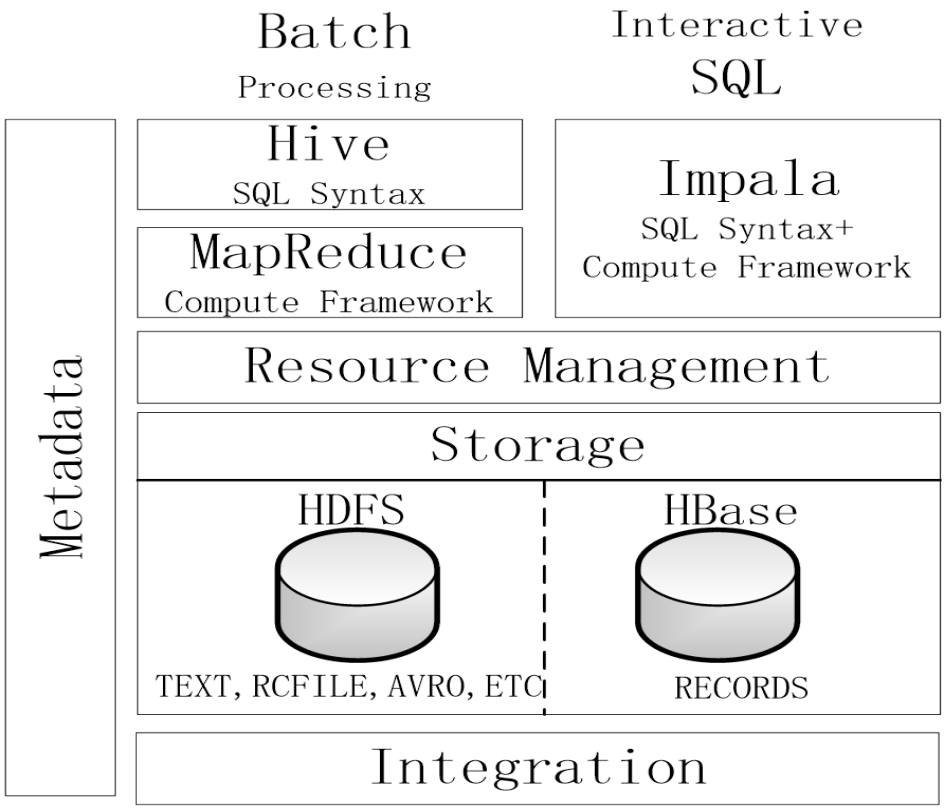


图　Impala与Hive的对比

Hive与Impala的相同点总结如下：

1. 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

2. Hive与Impala使用相同的元数据

3. 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

9.5.4 Impala与Hive的比较

总结

- **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**

9.6 Hive编程实践

9.6.1 Hive的安装与配置

1. Hive安装

安装Hive之前需要安装jdk1.6以上版本以及启动Hadoop

- 下载安装包apache-hive-1.2.1-bin.tar.gz

下载地址：http://www.apache.org/dyn/closer.cgi/hive/

- 解压安装包apache-hive-1.2.1-bin.tar.gz至路径 /usr/local
- 配置系统环境,将hive下的bin目录添加到系统的path中

2. Hive配置

Hive有三种运行模式，单机模式、伪分布式模式、分布式模式。均是通过修改hive-site.xml文件实现，如果 hive-site.xml文件不存在，可以参考$HIVE_HOME/conf目录下的hive-default.xml.template文件新建。

9.6.2 Hive的数据类型

表 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]

9.6.2 Hive的数据类型

表 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)

9.6.3 Hive基本操作

1. create: 创建数据库、表、视图

- 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;
```

9.6.3 Hive基本操作

- 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’;
```

9.6.3 Hive基本操作

- 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;
```

9.6.3 Hive基本操作

2. show：查看数据库、表、视图

- 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.*’;
```

- 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.*’;
```

9.6.3 Hive基本操作

3. 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;
```

9.6.3 Hive基本操作

4. 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;
```

9.6.4 Hive应用实例：WordCount

词频统计任务要求：
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

具体步骤如下：
（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
```

9.6.4 Hive应用实例：WordCount

（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下：

```
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)
```

docs

hello world
hello hadoop

W

word
hello
world
hello
hadoop

9.6.5 Hive的编程优势

WordCount算法在MapReduce中的编程实现和Hive中编程实现的主要不同点：

1. 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

在Hive中只需要编写7行代码

2. 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中不需要

HiveQL语句的最终实现需要转换为MapReduce任务来执行，这都是由Hive框架自动完成的，用户不需要了解具体实现细节。

本章小结

- **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，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，并在性能上比Hive高出3~30倍，甚至在将来的某一天可能会超过Hive的使用率而成为Hadoop上最流行的实时计算平台。**
- **以单词统计为例，详细介绍了如何使用Hive进行简单编程。**