第九章 数据仓库Hive

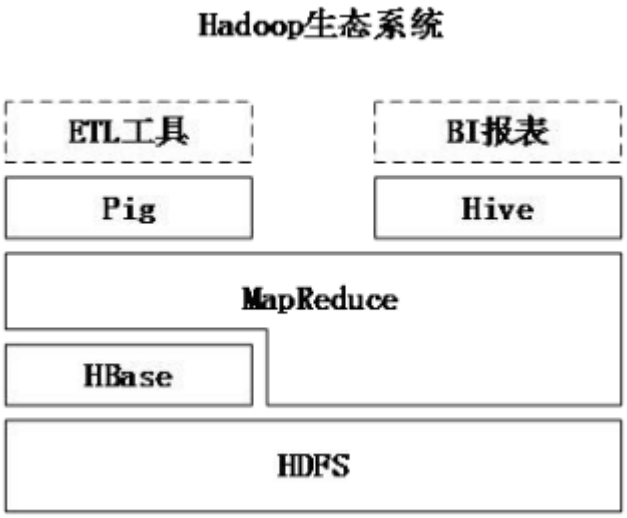
第九章考点

- Hive特点
- Hive与Hadoop的关系
- Hive与传统数据库的区别
- Hive系统架构（Hive HA，Impala

一、Hive概述

1. 数据仓库的概念
   数据仓库是一个面向主题的、集成的、相对稳定的、反映历史变化的数据集合，用于支持管理决策
2. 数据仓库与传统数据库的区别，数据库只能保存某一时刻状态的信息
3. 基于关系数据库的传统数据仓库面临的挑战：
   - 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和处理结构化数据。
   - 无法满足快速增长的海量数据存储需求：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，纵向扩展性有限。
   - 计算和处理能力不足。
4. Hive介绍
   Hive是一个构建于Hadoop顶层的数据仓库工具
   某种程度上可以看做是用户编程接口，本身并不存储和处理数据
   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SQL查询语言——HiveQL
   用户可以通过编写HiveQL语句运行MapReduce任务
   Hive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
5. **Hive特点**
   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数据转换成MapReduce任务运行
     数据仓库处理的是静态数据，对于静态数据的分析适合采用批处理的方式，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； 非常适合数据仓库应用程序维护海量数据、对数据进行挖掘、形成意见和报告等

6. **Hive在Hadoop生态系统中的关系**



- Hive依赖HDFS存储数据：
  HDFS作为可靠性的底层存储，是用来存储海量数据
- Hive依赖于MapReduce处理数据：
  MapReduce对这些海量数据进行处理，实现高 性能计算，用HiveQL语句编写的处理逻辑最终均要转化为MapReduce任务来运行
- Pig可以作为Hive的替代工具
  pig是一种数据流语言和运行环境，适合 用于Hadoop和MapReduce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，然后转换为用户期待的数据格式
  Pig适合进行实时交互分析。 Hive适合海量数据批处理分析
- HBase 提供数据的实时访问
  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，所以HBase与Hive 的功能是互补的，它实现了Hive不能提供功能。

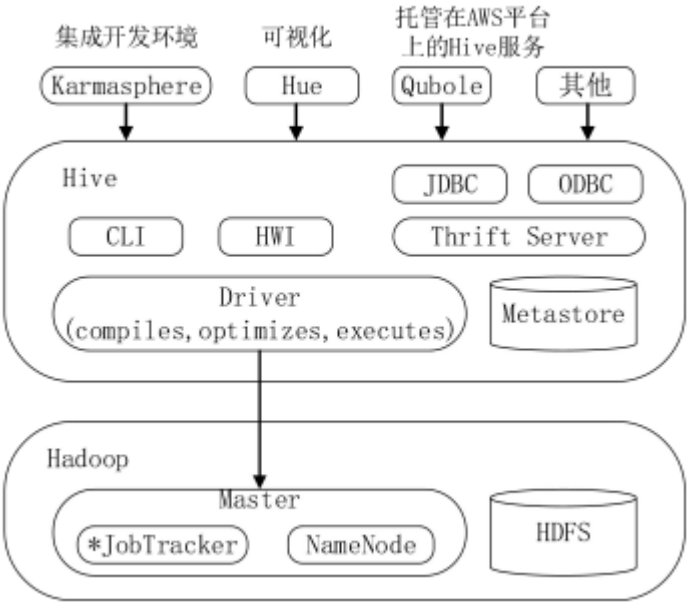
7. **Hive与传统数据库之间的区别**

- 数据插入：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。
- 数据更新：更新是传统数据库中很重要的特性，Hive不支持数据更新。 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。
- 索引：索引也是传统数据库中很重要的特性，Hive在hive 0.7版本后已 经可以支持索引了。但Hive不像传统的关系型数据库那样有键的概念，它只提 供有限的索引功能，使用户可以在某些列上创建索引来加速一些查询操作， Hive中给一个表创建的索引数据被保存在另外的表中。
- 分区：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，可管理性和提高数据库效率。Hive也支持分区功能，Hive表 组织成分区的形式，根据分区列的值对表进行粗略的划分，使用分区可以加快 数据的查询速度。
- 执行延迟：因为Hive构建于HDFS与MapReduce上，所以对比传统数据库来 说Hive的延迟比较高，传统的SQL语句的延迟少于一秒，而HiveQL语句的延迟会达到分钟级。

- 扩展性：传统关系数据库很难横向扩展，纵向扩展的空间也很有限。相反Hive的开发环境是基于集群的，所以具有较好的可扩展性。

二、Hive系统架构

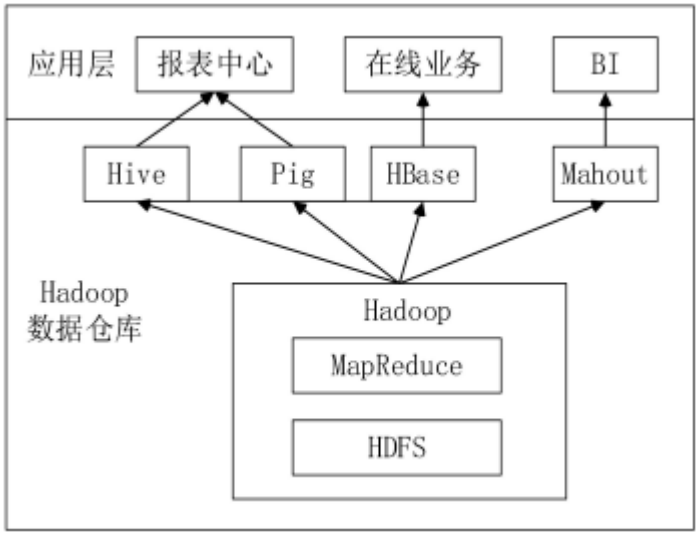
1. 系统架构



   - 用户接口模块：包括CLI、HWI、JDBC、ODBC 、Thrift Server等。 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。
   - 驱动模块：包括编译器、优化器、执行器等。 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，对需求的计算进 行优化，然后按照指定的步骤进行执行
   - 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。
2. SQL转化为MapReduce的原理
   SQL的表连接操作，分组操作，查询操作
   这部分具体查看PPT相关内容
   - 当启动MapReduce程序时，Hive本身是不会生成MapReduce算法程序的。
   - Hive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NameNode节点管理

三、Hive的应用

1. Hive数据仓库生态体系
   构建于Hadoop上的数据仓库，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。



   - 在Hadoop数据仓库中， Hive和Pig主要应用在报表中心上，Hive主要用 于报表分析，Pig主要用于报表中数据的转换工作；
   - HBase主要用于在线业务，因为HDFS缺乏随机读写操作，而HBase正是 为此开发的，支持实时访问数据。
   - Mahout常用于BI（商务智能） ，Mahout提供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智能应用程序。
2. **Hive HA**
   Hive的功能十分强大，可以采用支持SQL的方式查询Hadoop平台上的数据，但是实际应用中，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的出现就是为了解决这类问题
   - Hive在报表中心的应用流程：
     - 在Hadoop集群上构建的数据仓库由多个Hive进行管理；
     - 由HAProxy提供一个接口，对Hive实例进行访问；
     - 由Hive处理后得到的各种数据信息，或存放在MySQL数据库中，或直接以报表的形式展现；
     - 不同人员会根据所需数据进行相应工作
   - Hive HA原理
     - 资源池机制：将若干个hive 实例纳入一个资源池，然后对外提供一个唯一的接口。
     - 逻辑抽象：对于程序开发人员，就把它认为是一台超强“hive" 。每次它接收到一个 Hive查询连接后，都会轮询资源池里可用的hive 资源。

四、Impala简介

1. 概述与定位
   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。
   Impala的目的不在于替换现有的MapReduce工具，而是提供一个统一的平台用于实时查询
2. **Impala与Hive的初步比较**
   - 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。
   - Impala通过与商用并行关系数据库中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，从而大大降低了延迟，主要用于实时查询。
   - Impala和Hive采用相同的SQL语法、ODBC 驱动程序和用户接口。
3. 核心架构组成
   Impala主要由Impalad，State Store和CLI三部分组成。
   - Impalad是Impala的一个进程
     - 与HDFS的数据节点运行在同一节点上。
     - 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。
     - 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。
   - State Store负责收集分布在集群中各个impalad进程的资源信息，用于查询的调度。
     State Store会创建一个statestored进程。
     - 跟踪集群中的Impalad的健康状态及位置信息。
     - 创建多个线程来处理Impalad的注册订阅和与各类Impalad保持心跳连接。 当State Store离线后，Impalad一旦发现State Store处于离线时，就会进入recovery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给用户提供查询使用的命令行工具
     - 提供了Hue、JDBC及ODBC的使用接口。
4. 元数据与集成
   Impala采用与Hive相同的元数据、SQL语法、ODBC驱动程序和用户接口， 这样做的主要原因是在使用同一公司Hadoop产品时，批处理和实时查询的 平台是统一的。
   Impala中表的元数据存储采用的是Hive的元数据存储方式。
5. **Hive与Impala的比较**
   - 不同点：
   - Hive适合于长时间的批处理查询分析，而Impala适合于实时交互式SQL查询。
   - 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，Impala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。
   - 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。

- 相同点：
- 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格式数据， HBase存储表中记录。
- Hive与Impala使用相同的元数据。
- 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。

五、Hive实践

1. create：创建数据库、表、视图
   - 创建数据库
     `create database hive;` 创建数据库hive
     `create database if not exists hive;` 如果名为hive的数据库不存在则创建hive数据库，否则不进行操作
   - 创建表
     `use hive` 首先选定数据库
     `create table if not exists usr(id bigint,name string,age int);` 如果不存在usr用户表就创建，后面是表中的属性： `id,name,age`
     同时也可以这样进行，来指定创建位置
     ```
     create table if not exists usr(id bigint,name string,age int)
     location 'usr/local/hive/warehouse/hive/usr';
     ```
     创建外部表usr，包含表中的属性： `id,name,age` ，读取路径 `/usr/local/data` 中用“ , ” 分割的数据
     ```
     create table if not exists usr(id bigint,name string,age int)
     row format delimited fields terminated by ' , '
     location '/usr/local/data';
     ```
     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
     ```
     create table hive.usr(id bigint,name string,age int) partition by(sex
     boolean);
     ```
     在hive数据库中，创建分区表usr1，它通过复制表usr得到
     ```
     use hive;
     create table if not exists usr1 like usr;
     ```
   - 创建视图
     创建视图little_usr，只包含usr表中的id，age属性
     `create view little_usr as select id,age from usr;`
2. drop：删除数据库、表、视图
   - 删除数据库
     删除数据库hive，如果不存在会出现警告
     `drop database hive`
     如果有 `if exists` 关键字，即使不存在也不会抛出异常
     `drop database if exists hive;`
   - 删除表
     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

```
drop table if exists usr;
```

- 删除视图

  删除视图little_usr

  ```
  drop view if exists little_usr;
  ```

3. alter：修改表、视图
   - 修改表

     重命名表usr为user

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

     为表usr添加新的分区

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

     删除表中的分区

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

     把表中usr列名name修改为username，并把该列置于age列之后

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

     在表usr分区字段前，增加一个新列sex

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

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

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

   - 修改视图

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

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

4. show：查看数据库、表、视图
   - 查看数据库

     查看Hive中包含的所有数据库

     ```
     show databases;
     ```

     查看Hive中以h开头的所有数据库

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

   - 查看表和视图

     查看数据库hive中的所有表和视图

     ```
     use hive;
     show tables;
     ```

     查看数据库hive中以u开头的所有表和视图

     ```
     show tables in hive like 'u.*';
     ```

5. describe：描述数据库，表，视图
   - 描述数据库

     查看数据库hive的基本信息，包括数据库中文件的位置信息等

     ```
     describe database hive;
     ```

     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

     ```
     describe database extended hive;
     ```

- 描述表和视图
  查看表usr和视图little_usr的基本信息，包括列信息等
  `describe hive.usr/hive.little_usr`
  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
  `describe extended hive.usr/hive.little_usr`
  查看表usr中列id的信息
  `describe extended hive.usr.id;`

6. load：向表中装载数据
   - 把目录 `/usr/local/data` 下的数据文件中的数据装载进usr表中并覆盖原有数据
     `load data local inpath '/usr/local/data' overwrite into table usr;`
   - 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
     `load data local inpath ‘/usr/local/data’ into table usr;`
   - 把分布式文件系统目录 `hdfs://master_srever/usr/local/data` 下的数据 文件数据装载进usr表并覆盖原有数据
     `load data inpath 'hdfs://master_srever/usr/local/data'`
     `overwrite into table usr;`
7. Insert：向表中插入数据或从表中导入数据
   - 向表usr1中插入来自usr表的数据并覆盖原有数据
     `insert overwrite table usr1`
     `select * from usr where age=10;`
   - 向表usr1中插入来自usr表的数据并追加在原有数据后
     `insert into table usr1`
     `select * from usr`
     `where age=10;`