数据库系统概论新技术篇

SQL on Hadoop Systems

覃雄派

中国人民大学信息学院

2016年12月

SQL on Hadoop Systems

本讲的具体内容，包括如下几个部分

1.SQL语言的优势和SQL on Hadoop系统的诞生
2. SQL on Hadoop系统的分类
3. Hive on MapReduce与Hive on Tez
4. HDFS上的列存储结构RCFile、ORC、Parquet
5. Impala系统
6. Presto系统
7. VectorH系统
8. 一些性能评测结果

SQL on Hadoop Systems

1.SQL语言的优势和SQL on Hadoop系统的诞生

- 对结构化数据进行分析查询的主流语言，是关系数据库的SQL语言。

- SQL语言简单易学。

  随着关系数据库技术的兴起和流行，大量的开发者、数据库管理员、甚至普通用户通过SQL语言对数据库进行操作，以及进行简单的数据分析，他们熟悉SQL语言。

  换句话说，SQL语言具有广泛的用户基础。

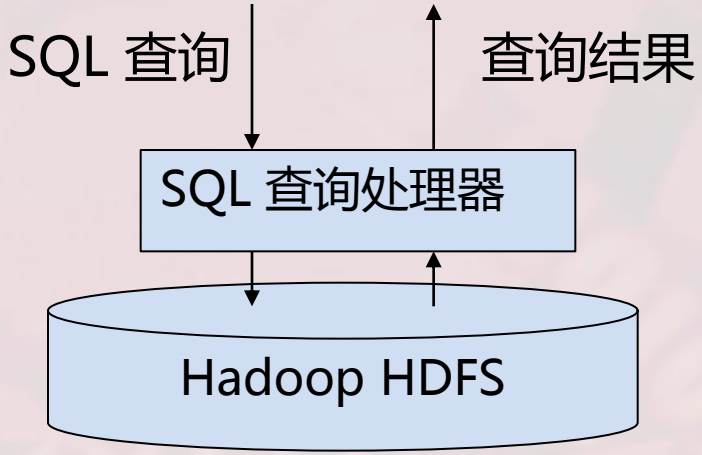
- Hadoop大数据平台及生态系统，使得人们可以对大规模的数据进行分析处理。但是使用MapReduce(Java)编程模型进行编程、或者使用过程性语言比如Pig进行编程，对于广大用户来说，仍然是具有挑战性的任务。

  人们希望能够使用熟悉的SQL语言，对Hadoop平台上的数据进行分析处理。这就是SQL On Hadoop系统诞生的背景。

SQL on Hadoop Systems

2. SQL on Hadoop系统的分类

- SQL on Hadoop系统是一类系统的统称，这类系统利用Hadoop实现大量数据的管理，具体是利用HDFS实现高度可扩展的数据存储。
- 在HDFS之上，实现SQL的查询引擎，使得用户可以使用SQL语言，对存储在HDFS上的数据进行分析。



SQL on Hadoop Systems

2. SQL on Hadoop系统的分类

SQL on Hadoop系统可以分为如下几类：

- **(1) Connector to Hadoop：**典型的代表，包括Oracle SQL Connector for Hadoop HDFS、Teradata Connector for Hadoop 等，这些连接器使得DBMS系统可以存取Hadoop里的数据，查询处理由RDBMS完成。
- **(2) SQL and Hadoop:** 修改现有的SQL Engine，决定查询的哪部分由RDBMS执行，哪部分由Hadoop(MapReduce)执行。这类系统包括 Hadapt、RainStor、Citus Data、Splice Machine、PolyBase等，其中PolyBase来自微软，它对SQL Server进行了修改。
- **(3) SQL on Hadoop:** 这类系统一般拥有一个全新设计的 SQL Engine，能够直接处理HDFS里的数据，甚至无需使用MapReduce计算模型。这类系统包括**Hive**、**Impala**、**Presto**、**VectorH**、 SparkSQL、Drill、HAWQ、IBM的Big SQL等。

在这里，我们所说的SQL on Hadoop系统，主要指的是第3类系统。在本讲中，我们将介绍其中几个重要系统，并展示第三方的一些性能对比试验结果。

SQL on Hadoop Systems

3. Hive on MapReduce与Hive on Tez

- Hive是Hadoop平台上的数据仓库系统，它提供类似SQL的查询语言HQL(Hive Query Language)，供用户进行数据分析。Hive是最先实现的SQL on Hadoop系统。

Hive的早期版本，把HQL查询翻译成一系列的MapReduce Job，在Hadoop上执行，查询执行的效率不高。

新版本的Hive运行在Tez之上，获得较高的执行效率。Tez是Hadoop平台上的新一代查询执行引擎。

- 关于MapReduce计算模型、运行时(run time)等相关内容，请参见另外一讲“大数据处理平台Hadoop & Spark及其生态系统”。

接下来，我们介绍Hive系统架构、Tez的原理，Hive On Tez和Hive on MapReduce的区别等。

SQL on Hadoop Systems

❖ 3. Hive on MapReduce与Hive on Tez

- Hive与关系数据库的区别
- 存储引擎不同
  - HDFS和专用的本地文件结构
- 计算模型
  - MapReduce与关系数据库则是自身的计算模型
- 实时性
  - Hive实时性差，关系数据库实时性好
- 扩展性
  - Hive扩展性好，关系数据库实时性差

❖ 3. Hive on MapReduce与Hive on Tez

- 小结
- **What Apache Hive is**
  - Tool used for **data warehouse** infrastructure
  - This tool is designed for **structured data** only
  - It stores and processes structured data residing in **HDFS**
  - Internally uses Hadoop **MapReduce** for Data Processing
- **What Apache Hive is not**
  - It is **not a Relational DB** like MySQL, Oracle, Postgres etc..
  - It is **not designed for real-time query** processing
  - It **doesn't support transactions, updates or delete at row level**

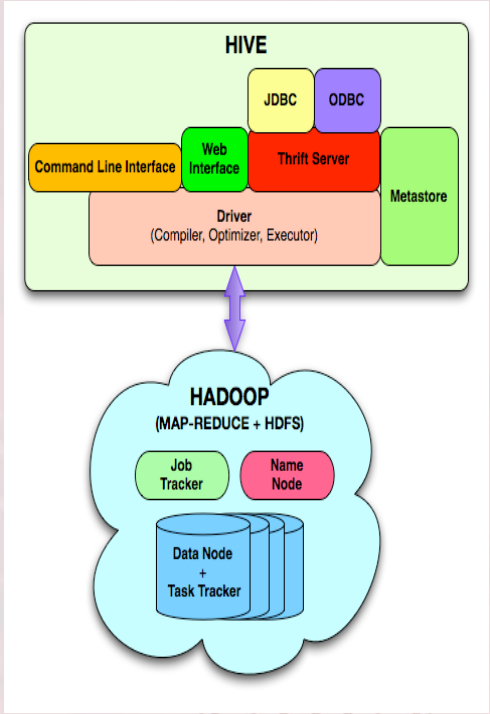
3. Hive on MapReduce与Hive on Tez

Hive系统架构

包括两类组件，即服务端组件和客户端组件。

Hive的服务端组件包括：

- **Driver组件**：该组件包括编译器Complier、优化器Optimizer、和执行器Executor，它的作用是将我们写的HQL(类SQL)语句进行解析、编译优化，生成执行计划，然后调用底层的MapReduce计算模型来执行。
- **MetaStore组件**：是元数据组件，负责存储Hive的元数据。Hive的元数据存储在关系数据库里，Hive支持的关系数据库有Derby、MySQL等。元数据对于Hive的正确运行，举足轻重。Hive支持把MetaStore服务独立出来，安装到远程的服务器集群里，从而解耦Hive服务和MetaStore服务，保证Hive运行的健壮性。
- **Thrift服务**：Thrift是Facebook开发的一个软件框架。用于开发可扩展的、跨语言的服务接口。Hive集成了Thrift服务，能让不同的编程语言调用Hive的接口。



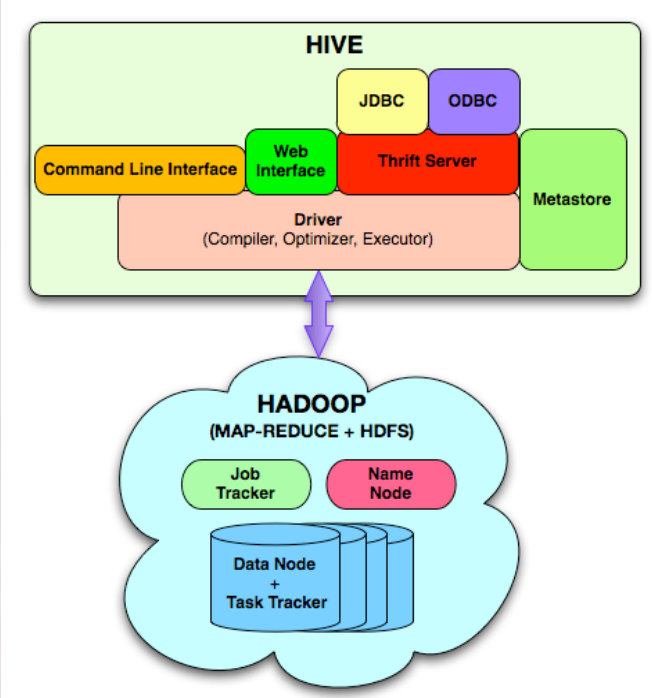
SQL on Hadoop Systems

3. Hive on MapReduce与Hive on Tez

- Hive系统架构

Hive的客户端组件，包括：

- **CLI**：Command Line Interface，命令行接口。
- **Thrift客户端**：Hive架构的许多客户端接口是建立在Thrift客户端之上，包括JDBC和ODBC接口。
- **WEB GUI**：Hive客户端提供了一种通过网页的方式，访问Hive所提供的服务。这个接口对应Hive的HWI(Hive Web Interface)，使用前要启动HWI服务。



3. Hive on MapReduce与Hive on Tez

- Hive支持的数据类型
- 包括整型Integer、浮点型Float、双精度浮点型Double，以及字符串String。
- Hive还支持更加复杂的数据类型，包括映射Map、列表List和结构Struct。这些复杂类型可以通过嵌套，表达更加复杂的类型。
- 除此之外，Hive允许用户自己定义类型Types和函数Functions，来扩展系统。

3. Hive on MapReduce与Hive on Tez

- Hive的**数据组织**

Hive使用传统数据库使用的表格Table、行Row、列Column、分区Partition等概念，易于理解。Hive的数据模型包括几个主要的管理层次，分别是Database、Table、Partition、Bucket等。

- **数据库(Database)：**相当于关系数据库里的命名空间(namespace)。它的作用是将不同用户的数据库应用，隔离到不同的数据库和模式中。
- **表格(Table)：**Hive的表，逻辑上由存储的数据和描述表格中的数据形式的相关元数据组成。表格数据存放在分布式文件系统里，即HDFS，元数据存储在关系数据库里。当创建一张Hive表，还没有为表加载数据的时候，该表在分布式文件系统，即 HDFS上就是一个文件夹(文件目录)。
- **分区(Partition)：**Hive里分区的概念是根据“分区列”的值对表的数据进行粗略划分的机制。在Hive存储上，表现为表的主目录(Hive的表实际对应一个文件夹)下的一个子目录，这个文件夹的名字就是我们定义的分区列的名字。
- **桶(bucket)：**Table和Partition都是目录级别的拆分数据。使用桶的表，会将数据文件按一定规律拆分成多个文件，每个桶就是表目录(或者分区子目录)里的一个文件。数据的分桶，一般通过Hash函数实现。创建表的时候，用户指定需要的桶的数量，以及使用哪个数据列进行分桶操作(Bucket the Data)。

SQL on Hadoop Systems

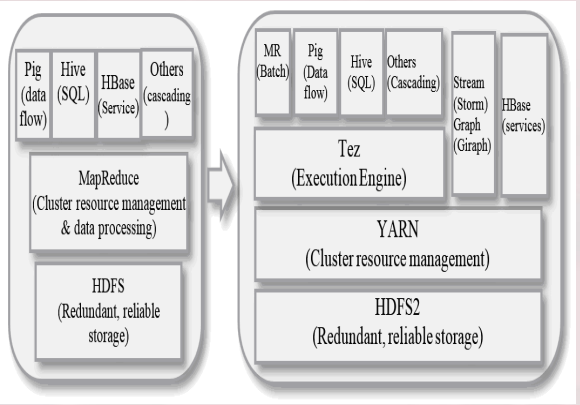
❖3. Hive on MapReduce与Hive on Tez

■ Hive on MapReduce

- Hive(<1.3）运行在MapReduce上
- SQL查询被转换成一系列的MapReduce Job，依次执行
- Hive利用HDFS的扩展能力、以及MapReduce计算模型的并行性，实现大数据的查询处理
- 受限于MapReduce计算模型，Hive的性能比MPP数据库差很多

SQL on Hadoop Systems

- 3. Hive on MapReduce与Hive on Tez
- Tez的原理
  - Tez是Hadoop2.0的新的查询执行引擎，Tez运行在YARN之上。
  - MapReduce作业、Hive查询、Pig程序等，运行在Tez上，获得更高的性能。
  - Tez把数据的分析处理工作，表达成DAG(directed acyclic graph) 形式。
- Tez的特点
  - 通过把一系列的MapReduce Job以一个Tez Job来运行，通过消除不必要的任务、任务同步机制(synchronization barriers)、以及中间结果存盘等，Tez提高了数据处理的速度。
  - 数据处理任务，包括小规模数据上的低延迟查询，以及大量数据上的高吞吐量负载，都从Tez获得性能收益。



3. Hive on MapReduce与Hive on Tez

- Tez的DAG(Directed Acyclic Graph)
- DAG是一个无环有向图，一个DAG定义一个数据处理应用程序的数据处理流程。即一个DAG，把数据处理逻辑表示成一张图。

  其中，DAG的顶点表示数据处理任务，它反映了一定的业务逻辑(Business Logic)，即如何对数据进行转换和分析。DAG的边，表示数据在不同顶点间的传递。
- Tez把每个顶点建模成Input、Processor、和Output模块的组合。Input和Output模块决定了数据的格式，以及数据应该从哪里读，写入到哪里去。Processor则包装了(hold)数据转换和处理的逻辑，它从一个或者多个Input消费数据，进行相关处理，然后产生多个Output。用户可以通过把不同Input、Processor、Output模块组合成顶点，在此之上创建DAG数据处理工作流，执行任意数据处理任务。

❖ 3. Hive on MapReduce与Hive on Tez

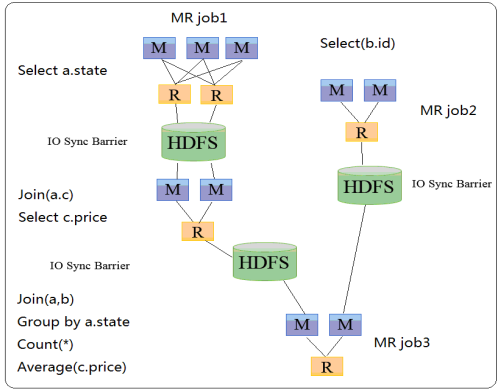
- Hive on Tez
- Tez 是 Hive 1.3 的新的执行引擎

Tez需要更少的作业(Tez需要一个作业，MapReduce需要3个作业)，无需任何的IO同步Barrier(MapReduce作业通过把中间结果写入HDFS和从中读出来实现同步)

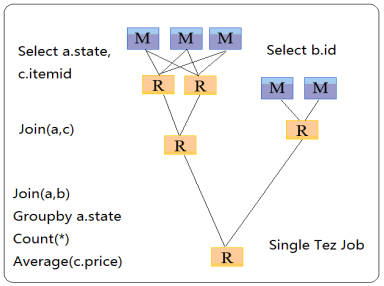
```
SELECT a.state, COUNT (*),
AVERAGE(c.price)

FROM a JOIN b ON (a.id = b.id)
JOIN c ON (a.itemId = c.itemId)

GROUP BY a.state
```



查询对应的MapReduce Job



查询对应的Tez DAG

4. HDFS上的列存储结构RCFile、ORC、Parquet

- 列存储的优势
- 1. 读取数据的时候可以把映射(Project Pushdown)下推，只需要读取查询需要的列，这样可以大大减少每次查询的I/O数据量。甚至可以支持谓词(Predicate Pushdown)下推，跳过不满足条件的列。
- 2. 由于每一列中的数据类型相同，可以使用针对性的编码和压缩方式，这样可以大大降低数据存储空间。

- HDFS上的列存储结构，有RCFile、ORC、Parquet等

4. HDFS上的列存储结构RCFile、ORC、Parquet

RCFile

❖RCFile(Record-Columnar File Format)是Facebook、Ohio州立大学、中科院计算所等合作研发的列存储文件格式。他们的成果以论文“RCFile: a Fast and Space-efficient Data Placement Structure in MapReduce-based Warehouse systems”发表在ICDE 2011(IEEE International Conference on Data Engineering)会议上。

❖Hive数据仓库使用RCFile文件格式进行数据存储，第一次在Hadoop中引入了列存储格式，提高了数据分析的性能。测试显示，在多数情况下它比其它已有的文件结构，拥有更好的性能。

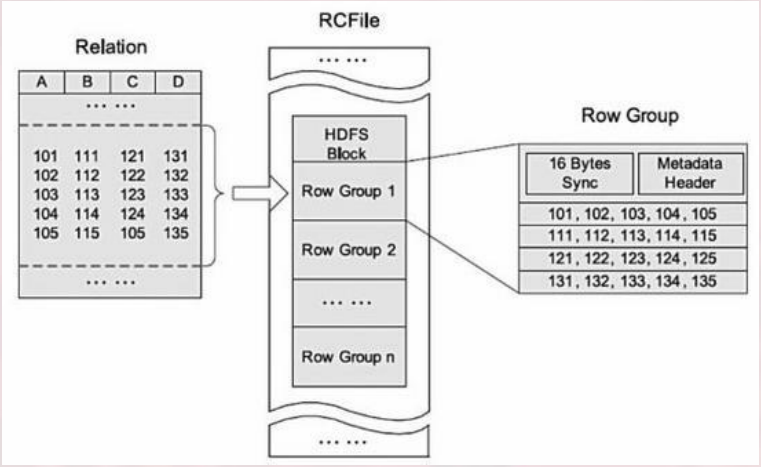
❖RCFile具有若干优势。首先，RCFile具备相当于行存储的数据加载速度和负载适应能力。其次，RCFile的读优化可以在扫描表格时避免不必要的列读取。最后，RCFile使用列维度的压缩，因此能够有效提升存储空间利用率。

❖为了提高存储空间利用率，Facebook各产品线应用产生的数据，从2010年起均采用RCFile结构存储，按行存储(Sequence File/Text File)结构保存的数据集，也转存为RCFile格式。

4. HDFS上的列存储结构RCFile、ORC、Parquet

RCFile

❖为了把表格的数据保存到文件中，RCFile首先横向分割表格，然后纵向分割表格。对表格的横向分割，把表格划分成多个行组(Row Group)，行组的大小可以由用户进行指定。在每个Row Group内部，RCFile按照列存储的一般做法，把各个列数据划分开，分别连续保存。



4. HDFS上的列存储结构RCFile、ORC、Parquet

ORC

❖ORC File(Optimized Row Columnar File)，是对RCFile做出优化的一种存储结构。ORC于File 2013于年开始研发，它的设计目标，是克服Hive其它格式的缺陷，进一步提高Hive查询处理速度、以及Hadoop的存储效率。运用ORC File可以高效地存储Hive数据，提高Hive的读、写操作、以及处理数据的性能。

❖和RCFile相比，ORC文件格式具有若干优势：

- (1) ORC支持更丰富的数据类型，包括Date Time、Decimal，以及Hive的复杂类型(Complex Types)，包括Struct、List、Map、Union等。
- (2) ORC File是一个自描述的(self-describing)、和类型感知的(type-aware)列存储文件格式。它为流式地读取(Streaming Reads)操作进行了优化，同时支持快速查找少数的数据行。
- ORC File是类型感知的，它对RCFile的优化在于，在文件写入的时候，Writer针对不同的数据列的数据类型，使用不同的编码器进行编码，从而提高压缩的比率，比如针对整数类型，使用变长压缩方法(Variable Length Compression)、针对字符串类型，使用词典编码(Dictionary Encoding)。
- (3) ORC File引入了轻量级的索引、以及基本的统计信息，包括各个数据列的Min、Max、Sum、Count等信息，于是在查询的处理过程中，可以忽略大量不符合查询条件的记录。
- 通过谓词下推(Predicate Pushdown)，查询处理器使用这些索引，确定哪些Stripe需要读取，并且利用Row Index进一步限定扫描的范围到最小由10000行构成的记录集合。

4. HDFS上的列存储结构RCFile、ORC、Parquet

ORC

❖一个ORC文件由多个Stripe、一个包含辅助信息的File Footer、以及Postscript构成。

❖(1) 每个Stripe包括索引数据Index Data、行数据Row Data、及一个Stripe Footer。

- ORC File将整张表划分成的一系列的行组(Row Group)，Stripe实际上就是一个Row Group。默认情况下，一个Stripe的大小为256MB, 其大小可以扩展的长度只受HDFS的约束。大尺寸的Stripe，针对串行I/O做出优化，使得读取数据的吞吐量提高了，需要读取的文件更少，并且减轻了Name Node的负载。
- Index Data部分，包含每个列的最小与最大值。此外，一系列的行索引条目(Row Index Entries)记录压缩块的偏移量，利用它可以跳到正确的压缩块位置。行索引条目，使得查询处理器可以跳过一些行，实现快速的数据读取。缺省情况下，每次最多可以跳过10000行。
- Row Data部分，包含每个列的数据，由若干数据流构成(Data Stream)。
- Stripe Footer部分，包含数据流的位置信息、以及每一列数据的编码方式。

❖(2) 在File Footer里面包含了该ORC File文件中所有Stripes的元信息，即每个Stripe的位置、每个Stripe中有多少行、以及每列的数据类型等。它里面还包含了列级别的一些聚集的果，比如Count、Min、Max、Sum等。

❖(3) 在文件的末尾，有一个称为Postscript的结构，它用来存储压缩参数、及Footer的长度(被压缩过的Footer的大小)。

SQL on Hadoop Systems

4. HDFS上

ORC

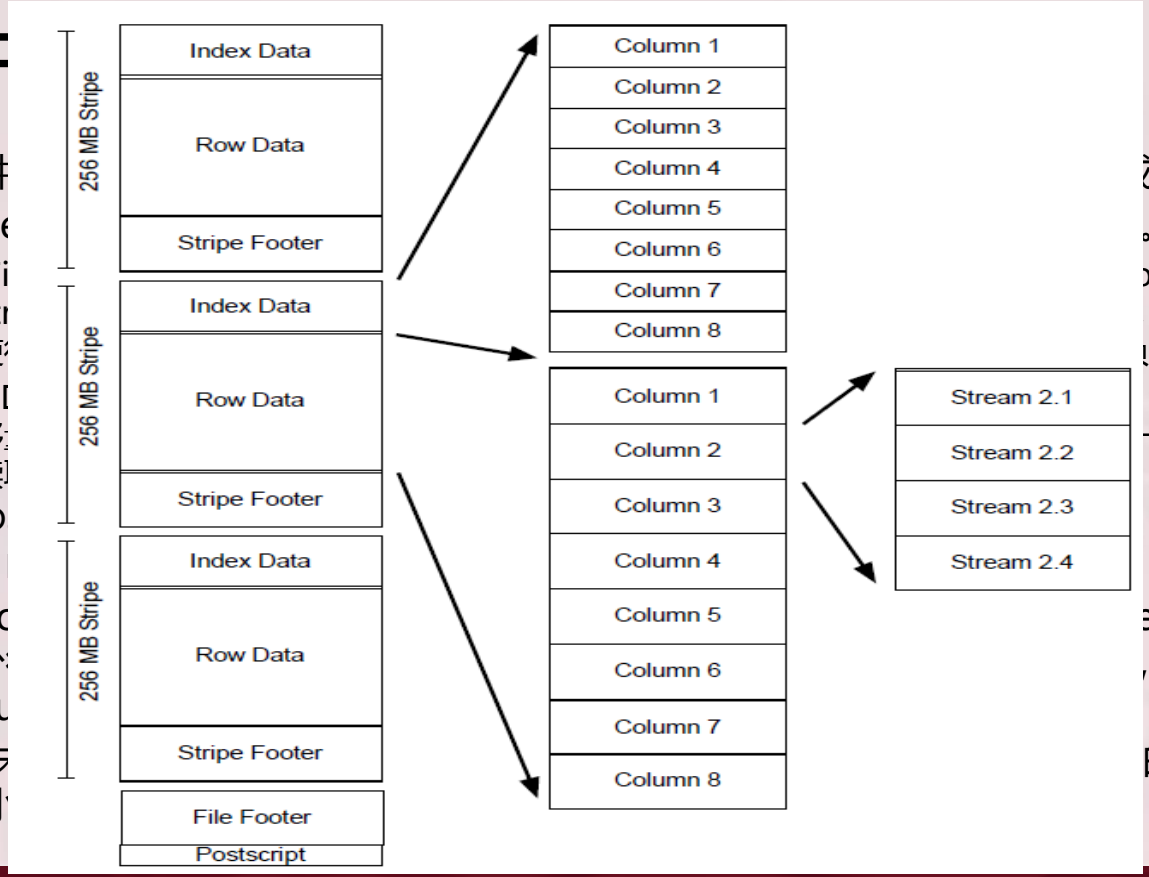
❖一个ORC文件

❖(1) 每个Stripe

- ORC Fi
  一个Str
  化，使
- Index D
  的偏移
  数据读
- Row D
- Stripe

❖(2) 在File Foo
Stripe中有多少
Min、Max、Su

❖(3) 在文件的末
的Footer的大小



。

。

up。默认情况下，
针对串行I/O做出优
载。

Entries)记录压缩块
些行，实现快速的

的位置、每个
比如Count、

的长度(被压缩过

4. HDFS上的列存储结构RCFile、ORC、Parquet

Parquet

❖Parquet是一种供Hadoop使用的列式存储格式。Parquet的灵感来自于2010年Google发表的Dremel论文。该论文介绍了一种支持嵌套结构的存储格式，并且使用了列存储的方式，提升查询性能。

❖Parquet为Hadoop生态系统中的所有项目提供支持高效率压缩的列存储格式，它兼容各种数据处理框架、对象模型。与编程语言无关，支持各种查询引擎(Hive、Impala、Presto等)。

❖Parquet最初是由Twitter和Cloudera(Impala的开发者)合作开发完成，并且开源。2015年5月从Apache的孵化器里孵化完成，成为Apache顶级项目。

SQL on Hadoop Systems

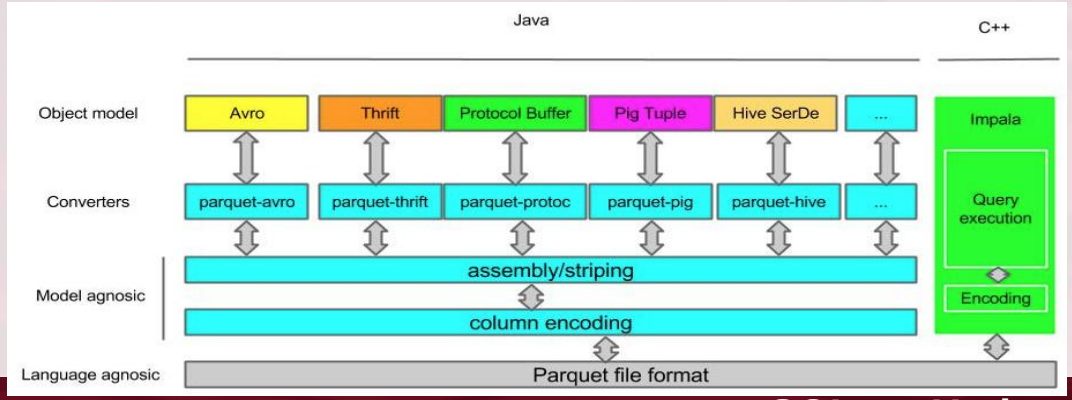
4. HDFS上的列存储结构RCFile、ORC、Parquet

Parquet在系统中的位置 - Parquet是一种文件的结构，处于存储层

❖(1) 存储格式(Storage Format)：Parquet-format定义了Parquet内部的数据类型、存储格式等。

❖(2) 对象模型转换器(Object Model Converters)：这部分功能由Parquet-mr来实现，完成外部对象模型与Parquet内部数据类型的映射。

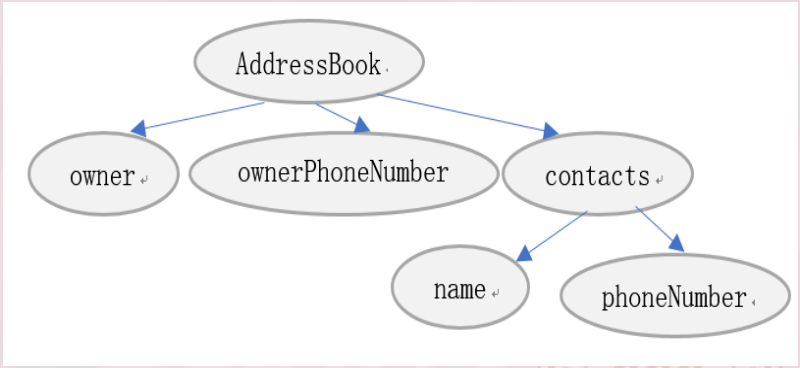
❖(3) 对象模型(Object Models)：对象模型可以简单理解为内存中的数据表示，包括Avro、Thrift、Protocol Buffers、Hive SerDe、Pig Tuple、Spark SQL Internal Row等对象模型。



4. HDFS上的列存储结构RCFile、ORC、Parquet

- Parquet数据模型
  - **AddressBook实例**

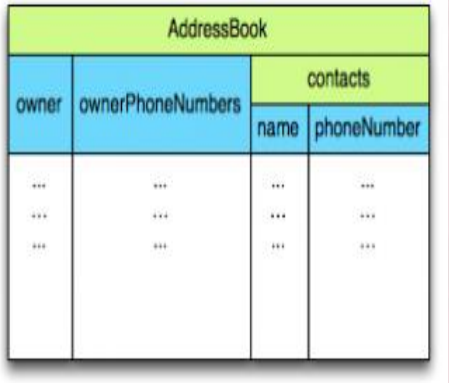
```
message AddressBook
{
        required string owner;
        repeated string ownerPhoneNumbers;
        repeated group contacts
    {
                required string name;
                optional string phoneNumber;
    }
}
```



4. HDFS上的列存储结构RCFile、ORC、Parquet

- Parquet数据模型
- 如何把内存中的每个AddressBook对象，按照列存储格式保存到硬盘文件中？
- 在Parquet格式的存储中，一个数据模式(schema)的树结构有几个叶子节点，实际的存储中就会有多少列(column)。于是，上述实例中的数据模式，在存储上，共有四个列(Column)

Column	Type
owner	string
ownerPhoneNumbers	string
contacts.name	string
contacts.phoneNumber	string



4. HDFS上的列存储结构RCFile、ORC、Parquet

Parquet数据模型

- Parquet文件的结构涉及如下几个重要的层次。
- (1) HDFS文件(File)：一个HDFS的文件，包括数据和元数据，数据分散存储在多个Block中。
- (2) HDFS块(Block)：它是HDFS上的最小的副本单位，HDFS会把一个Block保存到本地的一个文件，并且在不同的机器上的维护多个副本。通常情况下，一个Block的大小为256M、512M等。
- (3) 行组(Row Group)：按照行将数据在物理上划分为多个单元，每一个行组包含一定的行数。一个行组包含这个行组对应的区间内的所有列的列块。一般建议采用更大的行组大小(512MB-1GB)。更大的行组意味着更大的列块，有利于在硬盘上做更大的串行I/O。
  - 由于一次可能需要读取整个行组，所以一般让一个行组刚好在一个HDFS块中。因此，HDFS块的大小也需要被设得更大。比如，采用1GB的行组，1GB的HDFS块，1个HDFS块放一个HDFS文件。
- (4) 列块(Column Chunk)：在一个行组中，每一列保存在一个列块中。行组中的所有列连续的存储在这个行组文件中。不同的列块，可能使用不同的算法进行压缩。一个列块由多个页组成。
- (5) 页(Page)：每一个列块划分为多个页，页是压缩和编码的单元。在同一个列块内的不同页，可能使用不同的编码方式。一般建议一个页的大小为8KB。

SQL on Hadoop Systems

4. HDFS_
Parquet数

❖ Parquet文

❖(1) HDFS文 ...中。

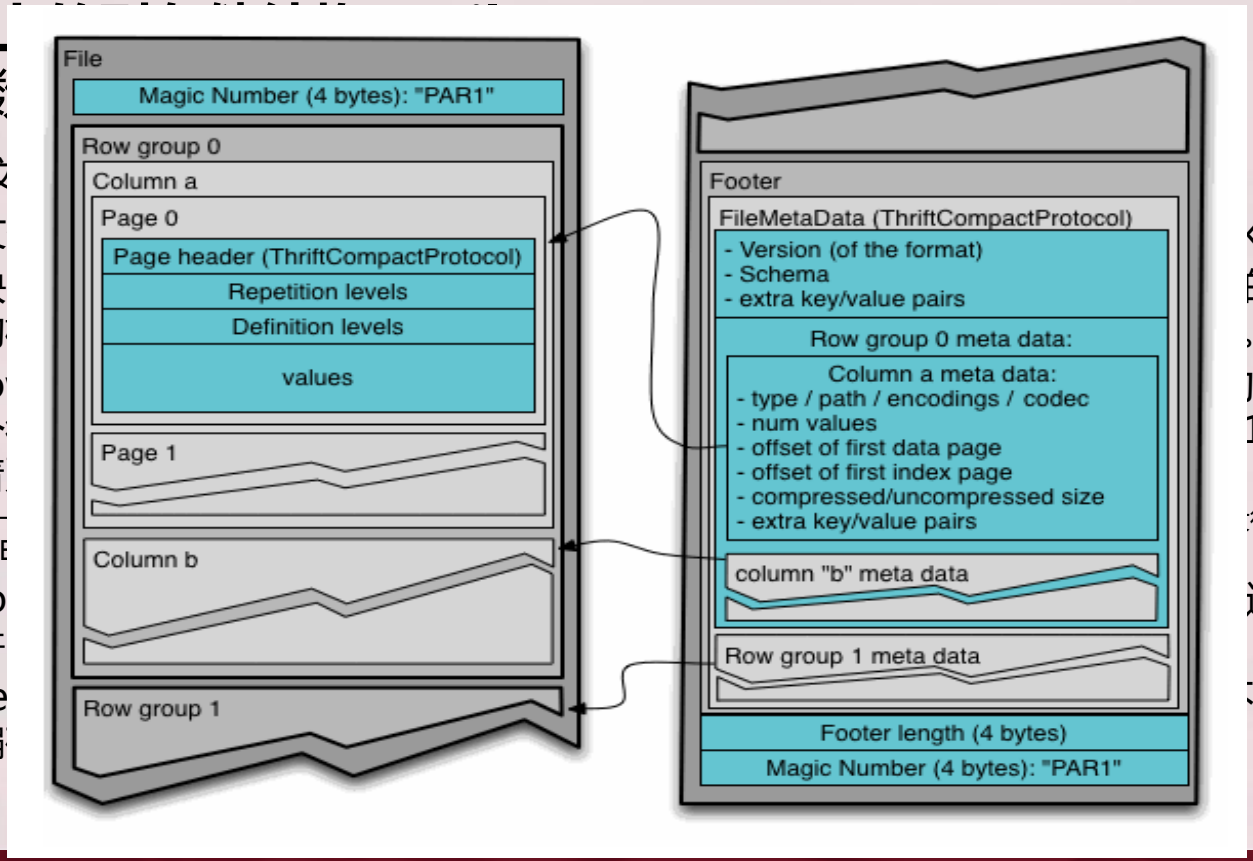
❖(2) HDFS块 ...的一个文件，并且在不同的...

❖(3) 行组(Ro ...行数。一个行组包含这个... ...1GB)。更大的行组意味着...

- 由于一... ...得更大。比如，采用1GB...

❖(4) 列块(Co ...连续的存储在这个行组文件...

❖(5) 页(Page ...同页，可能使用不同的编...



4. HDFS上的列存储结构RCFile、ORC、Parquet

❖ Parquet查询处理

❖谓词下推(Predicate Pushdown)

- 在数据库系统中，最常用的查询优化手段之一是谓词下推。通过将一些过滤条件尽可能的尽快执行，可以减少查询计划后续需要处理的数据量，从而提升性能。比如，对于SQL查询“select Count(1) from A Join B on A.id = B.id where A.a > 10 and B.b < 100”，如果在处理Join操作之前，首先对A和B执行Table Scan操作，然后再进行Join，再执行过滤，最后计算聚合函数返回，效率是不高的。但是，如果把过滤条件A.a > 10和B.b < 100，分别移到A表的Table Scan和B表的Table Scan的时候执行，遴选出符合条件的记录，就可以大大降低Join操作的输入数据，提高查询效率。
- 在Parquet中，每一个Row Group的每一个Column Chunk在存储的时候，都计算对应的统计信息，包括该Column Chunk的最大值、最小值和空值个数。通过这些统计值，和查询在该列上的过滤条件，就可以判断该Row Group/Column Chunk是否需要扫描。Parquet的新版本，将增加诸如Bloom Filter等索引结构，帮助完成数据的过滤。

SQL on Hadoop Systems

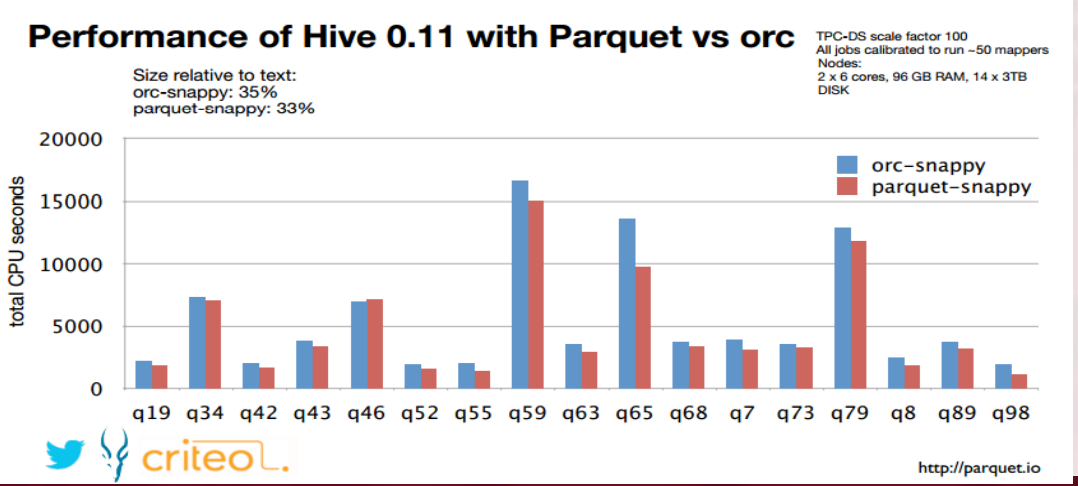
4. HDFS上的列存储结构RCFile、ORC、Parquet

- Parquet查询处理
- 映射下推(Project Pushdown)
  - 对于列存储来讲，映射下推是其最突出的优势。在获取表中原始数据时，只需要扫描查询中需要的列。由于避免了不必要的数据列的提取，查询的效率就能够得到提高。

4. HDFS上的列存储结构RCFile、ORC、Parquet

Parquet查询处理

❖Criteo公司在Hive中使用ORC和Parquet两种列式存储格式执行TPC-DS基准测试的结果。从测试结果可以看出，在数据存储方面，两种存储格式在都是用snappy压缩的情况下，不同存储格式占用的空间相差并不大，在查询性能上Parquet格式稍好于ORC格式。

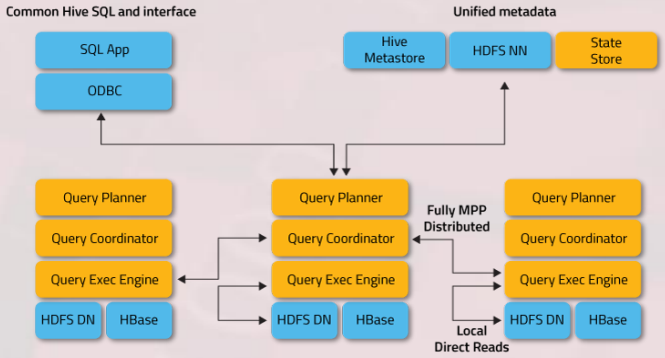


SQL on Hadoop Systems

5. Impala

- Impala是Cloudera公司开发的大数据实时查询系统。它提供SQL查询接口，能够查询存储在Hadoop的HDFS 和HBase中的PB级大数据。
- Impala的实现，借鉴了Google的Dremel系统

- Impalad包含 Query Planner、Query Coordinator和Query Exec Engine三个模块。Query Planner接收来自客户端的SQL查询请求，然后将其查询转换为许多子查询。Query Coordinator将这些子查询分发到各个节点上，由各个节点上的Query Exec Engine负责子查询的执行，最后返回子查询的结果。这些中间结果经过聚集之后，最终返回给用户。



5. Impala

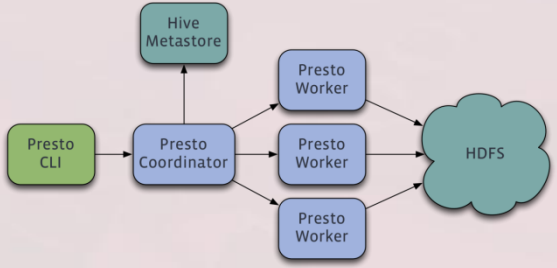
Impala的特点

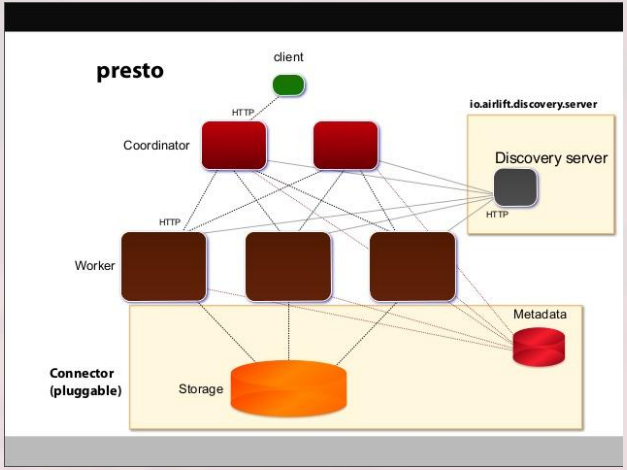
- (1) Impala不需要把中间结果写入磁盘，节省了大量的I/O开销。
- (2) Impala像Dremel一样，借鉴了MPP并行数据库查询处理技术的思想，抛弃MapReduce计算模型，对查询进行深入的优化，省略不必要的Shuffle、Sort等操作。
- (3) MapReduce启动task的速度是很慢的(默认每个心跳间隔是3秒钟)，Impala直接通过服务进程来进行作业调度，节省了MapReduce作业的任务启动的开销，速度快了很多。
- (4) Impala使用LLVM(Low Level Virtual Machine)来编译运行时代码，提高了查询计划的执行效率。
- (5) Impala使用C++语言实现，针对硬件进行了针对性的优化，包括使用SSE指令集，提高数据处理效率等。
- (6) Impala基于数据的局部性(Data Locality)进行I/O调度，尽量把数据和计算，分配在同一个节点上，较少了网络开销。

SQL on Hadoop Systems

6. Presto

- Presto是由Facebook贡献出来的开源SQL on Hadoop项目(2012年开发，2013年开源)，它的目标和Impala是类似的，就是要提供在Hadoop平台上对结构化数据的交互式(interactive)查询能力
- Presto采用Master-Slave的架构，它由一个Coordinator节点，多个Worker节点组成。Coordinator节点中通常内嵌一个Discovery Server。Coordinator负责解析SQL语句，生成执行计划，分发执行任务给Worker节点执行。
- Worker节点负责实际执行查询任务。Worker节点启动后向Discovery Server服务注册，Coordinator从Discovery Server获得可以正常工作的Worker节点
- Presto通过Connector插件方式，支持不同的存储层

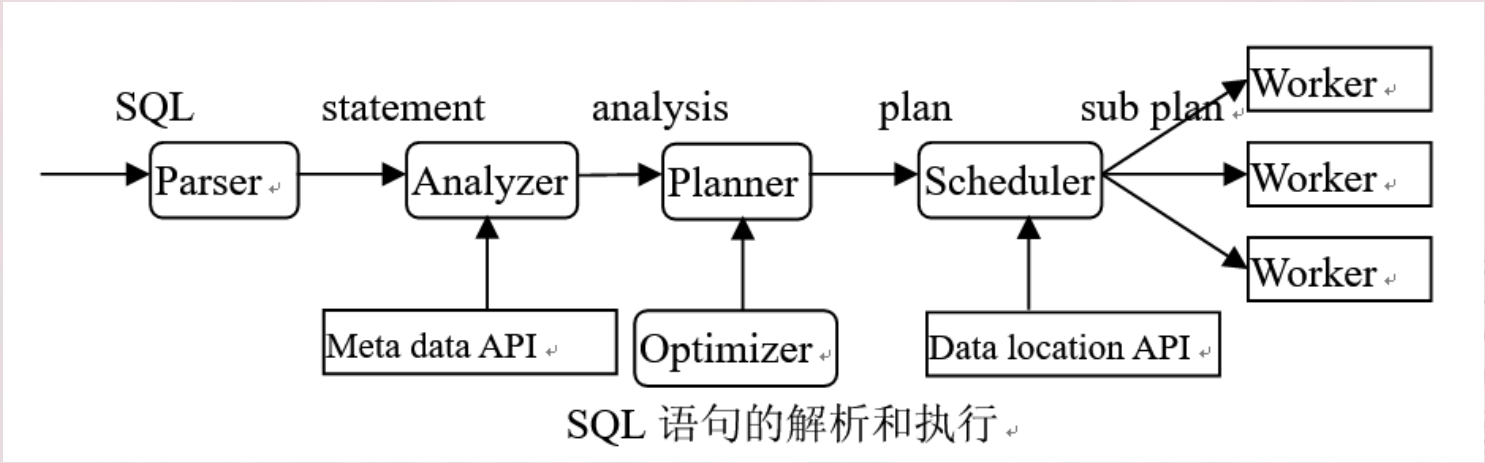




SQL on Hadoop Systems

6. Presto

- SQL语句的解析过程
- Presto服务器对其接收到的SQL查询，大致经过如下步骤对其进行解析和执行。首先Parser对SQL语句进行词法和语法解析，生成statement结构。然后Analyzer利用元信息，对语法树进行检查，生成analysis结构。Planner利用Optimizer的帮助，进行查询优化，并生成逻辑计划plan。Scheduler根据数据的位置信息，把逻辑计划划分成物理计划片段sub plan，分发给若干Worker节点，由其执行

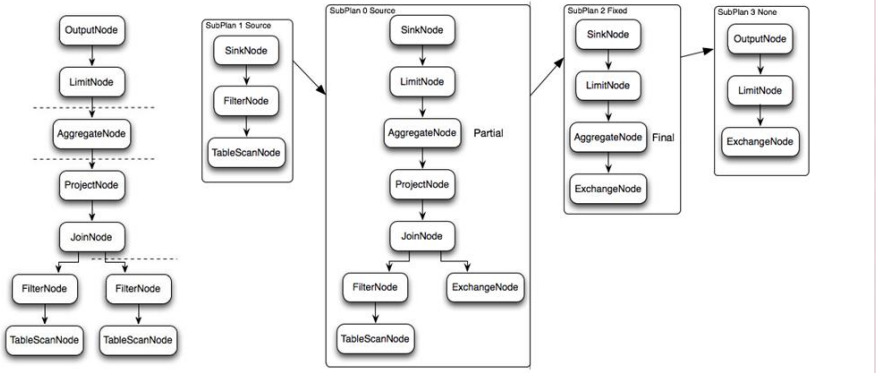


SQL 语句的解析和执行

SQL on Hadoop Systems

6. Presto

- 一个SQL查询语句对应的逻辑执行计划(Logical Execution Plan)如图所示（左边）。
- 逻辑执行计划图中的虚线，是Presto对逻辑执行计划的切分点。这些切分点把逻辑计划，划分成四个子计划Sub Plan，每个Sub Plan交给一个或者多个Worker节点运行。先导和后续Sub Plan之间需要进行数据交换，如图所示（右边）。



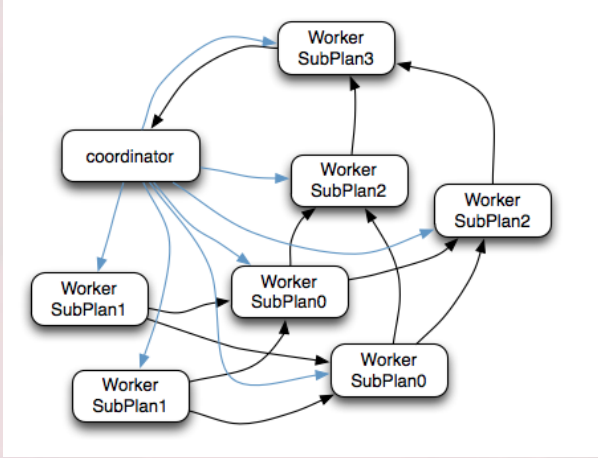
一个 SQL 的逻辑执行计划

SQL on Hadoop Systems

6. Presto

Presto并行查询执行过程

- (1)Coordinator通过HTTP协议调用Worker节点的任务(task)接口，将执行计划分配给Worker节点(图中蓝色箭头)。
- (2)执行SubPlan1的每个Worker节点，读取一个Split的数据，并过滤后将数据分发给每个SubPlan0节点，进行Join操作和Partial Aggregation操作。
- (3)执行SubPlan0的每个Worker节点，计算完成后，按Group By Key的Hash值，将数据分发到不同的SubPlan2节点。
- (4)执行SubPlan2的每个Worker节点，进行全局聚集，进行计算完成，后将数据分发到执行SubPlan3的Worker节点，执行limit操作。
- (5) 执行SubPlan3的Worker节点计算完成后，通知Coordinator结束查询，并将数据发送给Coordinator，由其返回客户端程序



SQL on Hadoop Systems

7. VectorH

MonetDB/Vectorwise/VectorH

- MonetDB是荷兰研究机构CWI(Centrum Wiskunde & Informatica)于2003年开始研发的基于列存储的面向分析型应用的内存数据库系统

- 2008年，CWI基于MonetDB，创立了Vectorwise公司，实现MonetDB的商业化。后来Vectorwise被Actian收购。

- 在Vectorwise数据库基础上，Actian研发了MPP并行数据库原型，这是一个采用Shared Nothing架构的并行数据库系统。

  基于这个MPP数据库原型，Actian公司把Vectorwise迁移到了Hadoop平台上，利用HDFS作为存储层，研发了SQL on Hadoop系统VectorH。VectorH扩展了Vectorwise MPP并行数据库原型系统，利用Hadoop 平台的YARN资源管理器，实现工作负载和资源调度

  - (1) VectorH对HDFS的块复制策略(HDFS replication policy)进行了干预，控制数据块如何在各个节点间进行复制和放置，目的是优化读操作的局部性(Read Locality)
  - (2) 虽然HDFS是只能进行数据添加的(Append)，但是VectorH通过基于位置的Delta树(Positional Delta Trees, PDT)，实现了数据更新(Update)功能

SQL on Hadoop Systems

7. VectorH
MonetDB/Vectorwise/VectorH

- **VectorH采用**向量化的查询处理模式(Vectorized query processing)。所谓向量化的查询处理模式，是每个属性列的每100-1000个值构成以一个向量。执行查询的时候，一系列的向量以流水线的方式流过，对该属性列进行操作的一系列操作符，以致最后完成查询。对属性列进行操作的若干操作符，一次处理一个向量。

- 向量的大小设定，考虑了CPU Cache的大小，很容易装载到CPU Cache中，提高Cache命中率。
- 这种查询处理模式，可以充分利用现代CPU的SIMD指令(single instruction, multiple data)，通过并行操作，快速完成数据的处理。

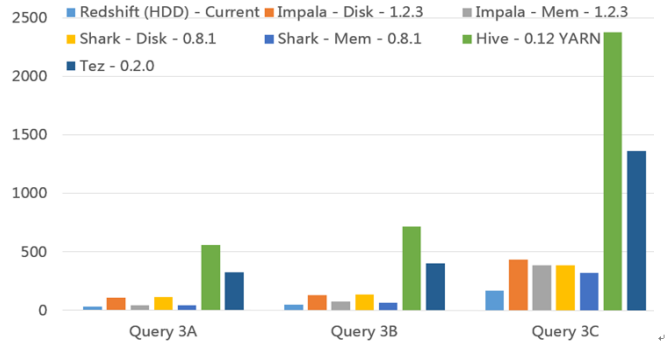
SQL on Hadoop Systems

- **8. 一些性能评测结果**
- **性能评测1**
- 加州大学Berkeley分校的AMP Lab于2014年初，对Redshift、Hive on MapReduce(v0.12)、Shark(v0.8.1)、Impala(v1.2.3)、Hive on Tez(v0.2.0)等6套结构化数据分析系统，进行了性能评测。其中，Redshift是一个来自Amazon的基于ParAccel数据仓库技术的MPP数据库。Shark为SparkSQL的前身，它是Spark平台上和Hive兼容的SQL查询引擎。
- 他们的测试数据集，包括三个数据库表。Ranking 表保存各个页面和它们的Page Ranks。UserVisits表保存对每个Web页面的用户存取日志。Documents表保存每个页面的非结构化的页面HTML内容。
- 他们使用Scan Query、Aggregation Query、Join Query等不同查询负载，对目标系统进行了性能测试

当选择率比较低的时候，Spark(in Memory, 表示数据驻留在内存中)获得和Redshift和Impala (in-memory)类似的响应时间，并且比其它系统(配置)的性能都要好。

当选择率比较高的时候，Spark (in-memory)比其它系统(配置)的性能都要好，除了Redshift。Redshift总是后的比Spark更高的性能。

从图中，我们可以看出，Redshift的性能是最好的，Impala和Shark性能接近，两者的性能都比Hive高出很多。Hive (on MapReduce)的性能最差，Hive on Tez的性能相对于Hive有所改进。



各个系统(配置)执行 Query 3(不同选择率)的中位数响应时间(Median Response Times)

SQL on Hadoop Systems

- **8. 一些性能评测结果**
- **性能评测2**
- Actian公司2016年对VectorH(5.0)进行了性能评测，并且和Impala(2.3)、Hive(1.2.1)、HAWQ(1.3.1)以及SparkSQL(1.5.2)进行了对比。它们使用TPC-H数据集(Scale Factor为1000)。他们使用的集群由10个节点组成，运行Hadoop 2.6.0，其中一个节点作为name Node，其余9个节点用于进行SQL on Hadoop系统的性能试验

表 6. TPC-H(Scale Factor 1000)数据集上个各个查询的响应时间(秒)

VectorH	1.5	1.14	3.16	0.17	1.94	0.31	2.75	1.31	11.11	1.21	1.69	0.34	3.66	0.83	1.63	1.68	1.24	0.99	1.32	2.15	1.48	2.84
HAWQ	158.2	21.46	32.06	38.21	36.38	20.19	44.74	48.38	766.4	32.97	12.48	31.75	27.97	19.47	31.58	14.17	173.2	87.08	24.82	42.84	84.7	29.44
SparkSQL	155.4	74.98	62.38	68.27	146.5	5.1	180.2	174.6	264.0	56.62	30.28	66.97	47.65	6.92	11.16	33.81	244.9	254.7	24.89	31.56	1614	91.18
Impala	585.4	81.81	167.7	163.18	242.5	1.81	369.0	276.2	1242.9	69.97	35.04	45.67	180.8	13.95	15.19	47.52	581.53	1234	714.7	74.25	880.8	34.81
Hive	490.1	63.57	266.6	59.08	DNF	63.63	721.8	625.6	1077	230.5	246.1	65.78	140.7	53.23	556.5	92.51	711.7	454.5	1010	100.5	247.7	81.11

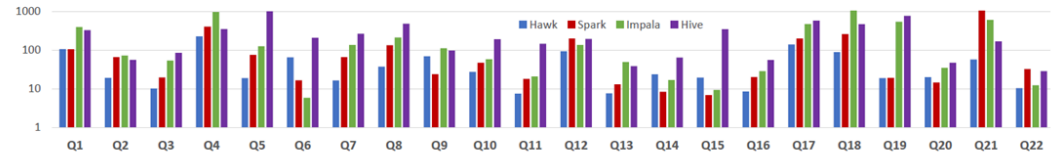


图 23. VectorH 比各个系统快多少倍

VectorH系统获得了秒级的响应时间，比HAWQ、SparkSQL、Impala、Hive等系统快超过一个数量级(甚至达到2-3个数量级)

SQL on Hadoop Systems

- **8. 一些性能评测结果**
- **性能评测3**
- 2016年底AtScale公司发布了他们针对主流的SQL on Hadoop结构化大数据分析平台的性能评测结果，他们评测的系统包括SparkSQL(2.0.1)、Impala(2.6)、Hive/Tez(2.1,Tez版本为0.8.5)、和Presto(0.152)。
- 他们进行测试的集群包括12个节点，其中1个节点为master node，1个节点作为gateway node，其余10个节点作为data nodes。
- 他们采用SSB(Star Schema Benchmark)测试基准进行评测。在其中的一个评测中，LINEORDERS的记录数达到60亿(6 Billion)。

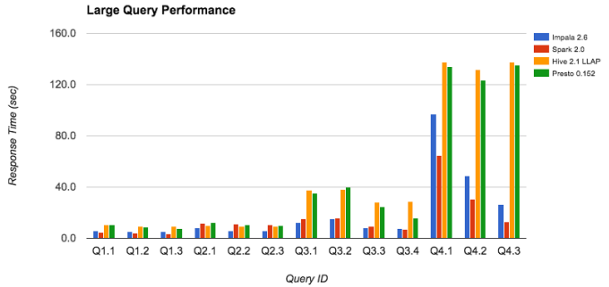
基于该评测结果，他们的结论是：

(1) 没有一个SQL on Hadoop 在所有的查询上获得最好的结果。SparkSQL 和Impala比Hive要快的多。在多数查询上，Impala和SparkSQL的性能差异很小。Presto 0.152获得了和Hive 2.1相似的性能。

(2) 随着一个查询里Join操作的数量的增加，查询处理时间也相应增加。当Join操作的数量从1增加到3，Hive和Presto的查询响应时间的变化较大。而Impala 和Spark SQL 则比Presto和Hive 获得更好的性能。

(3) 随着查询选择率的增加，查询响应时间也相应增加。Hive和Presto对查询选择率的敏感度相对较弱，也就是选择率增加，响应时间的变化没有那么大。

(4) 在所有的系统上，在两张大表之间进行Join操作，查询的响应时间都会变慢。一般来讲，对两张大表进行Join操作，比如CUSTOMERS达到1 0亿行记录(1 Billion)，是一个代价极大的操作。



采用SSB测试基准对四个系统进行评测的结果