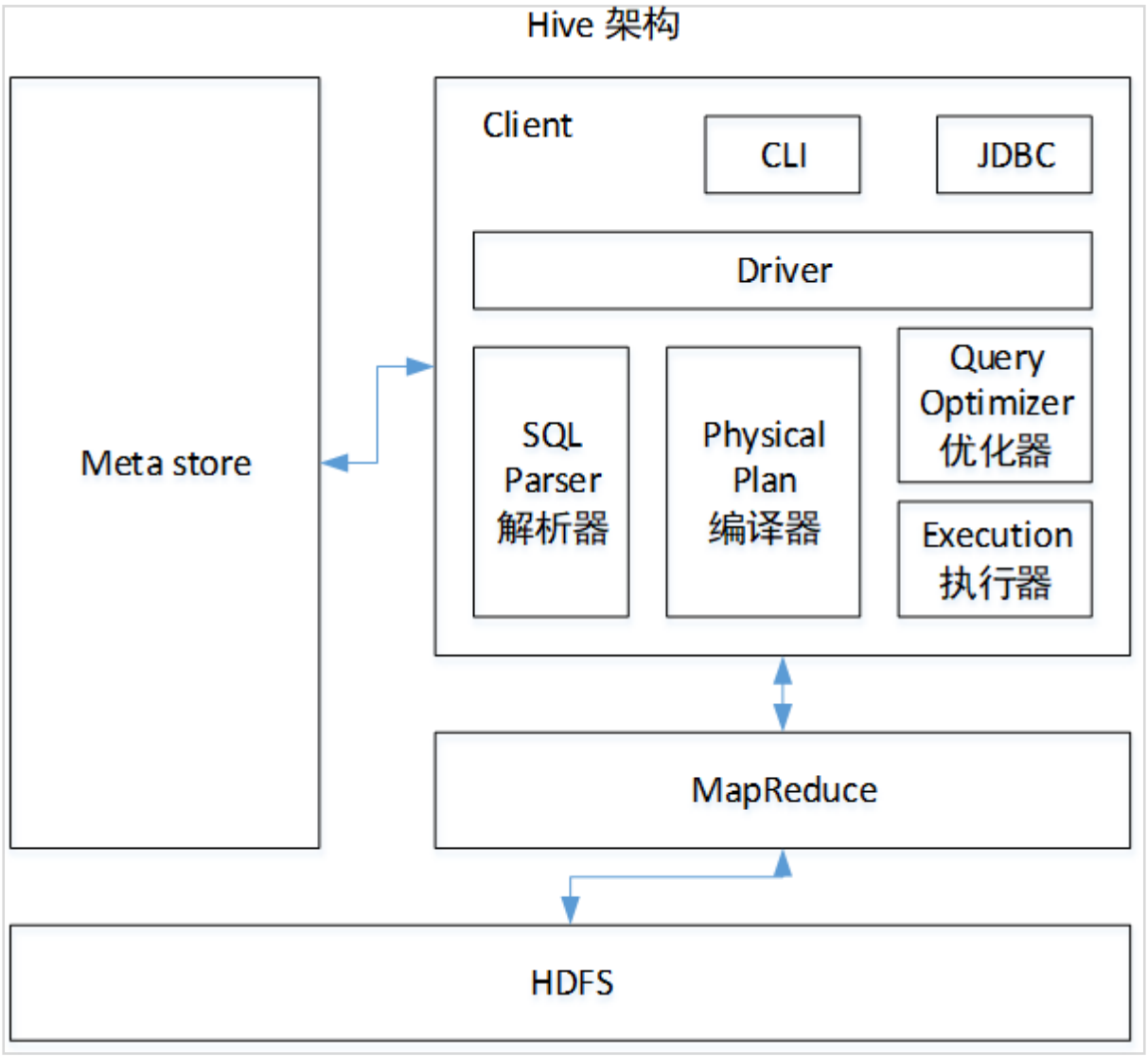
Hadoop-Hive

架构原理



与传统数据库区别

- Hive是读时模式，传统数据库是写时模式
- 读时模式：查询数据时对数据进行验证，加载数据时很快
- 写时模式：写入数据验证，不符合表模式则拒绝加载，查询数据会很快
- 数据类型不同

Hive表

Hive表分两种，外部和托管，主要区别在LOAD（移动）和DROP（删除）上，

- 外部表（也叫管理表，托管表，load数据时数据目录不变）
  - 关键字EXTERNAL
  - hive到仓库目录以外的位置访问数据
  - 执行DROP，只删除元数据，数据本身不会受影响

    - 建表的时候需要指明数据路径
- 内部表（load数据时数据从HDFS目录下移动到Hive目录下）
    - hive把数据移入到仓库目录
    - hive管理这些数据
    - 执行DROP，数据彻底消失

- 大部分时间两种表没有明显差别，如果所有处理都由Hive处理，应该使用托管表，如果需要Hive和其他工具一起处理，使用外部表
- 大部分数据存在HDFS上，Hive建立托管表

分区&分桶

分区

- 关键字：PARTITION BY
- 作用：高效处理数据，加快数据分片（slice）速度
- 可根据时间、日期等对表进行粗略划分
- 可根据多维度分区，对每个分区进行子分区（对日志进行日期分区，国家子分区）
- 创建表时显式制定分区，加载数据的时候指定分区名

静态分区&动态分区

静态分区：需要用户在插入数据的时候手动指定hive的分区字段，导致复杂度提高
动态分区：根据某一个或者几个字段插入到不同目录，无需手动输入

动态分区配置

```
--hive设置hive动态分区开启
  set hive.exec.dynamic.partition=true;
  默认：true
--hive的动态分区模式
  set hive.exec.dynamic.partition.mode=nostrict;
  默认：strict（至少有一个分区列是静态分区）
--每一个执行mr节点上，允许创建的动态分区的最大数量(100)
  set hive.exec.max.dynamic.partitions.pernode;
--所有执行mr节点上，允许创建的所有动态分区的最大数量(1000)
  set hive.exec.max.dynamic.partitions;
--所有的mr job允许创建的文件的最大数量(100000)
```

```
set hive.exec.max.created.files;
```

分桶

- 关键字：CLUSTERED BY
- 可根据ID取余进行划分
- 数据量太大的时候方便进行测试（抽样）
- 高效处理，join分桶的表可以在map端更高效
- 桶中数据可以根据某列进行排序

桶表理解

- 对数据hash取值，分发到不同文件
- 数据加载到桶表时，会对字段取hash值，然后与桶的数量取模。把数据放到对应的文件中。物理上，每个桶就是表(或分区）目录里的一个文件
- 桶数量与MapReduce数相同
- 专门用于抽样调查，不是日常存储数据的表

存储格式

- 行格式，默认为行
- 二进制存储格式
- 定制的SerDe，效率低

数据倾斜问题

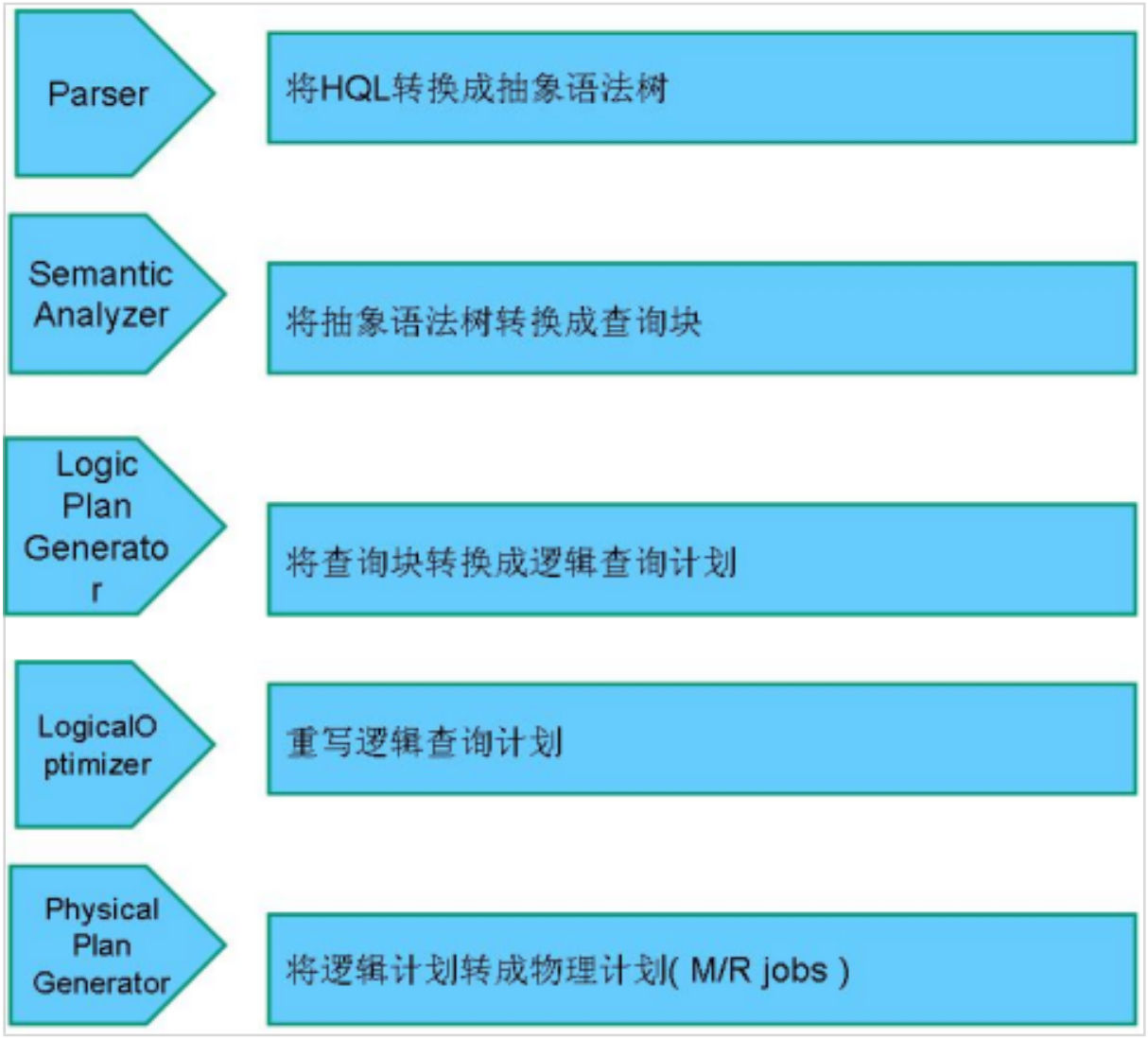
- 产生原因：map输出数据按key Hash的分配到reduce中，由于key分布不均匀、业务数据本身的特、建表时考虑不周、等原因造成的reduce 上的数据量差异过大
  - key分布不均
  - 业务本身特性
  - 建表时考虑不周
  - SQL问题
- 解决
  - 参数调节，会生成两个MR job，第一个将Map结果均匀分布到Reduce中，第二个将第一步预处理的结果按key分布到Reduce中，完成操作

```
 hive.map.aggr = true
```

```
hive.groupby.skewindata=true
```

- SQL语句调节，本质上是多表join，尽量减少join之后的行数
  - 选用join key分布最均匀的表作为驱动表。做好列裁剪和filter操作，以达到两表做join 的时候，数据量相对变小的效果。
  - 大小表Join，使用map join让小的维度表（1000 条以下的记录条数）先进内存。在map端完成reduce。
  - 大表Join大表，把空值的key变成一个字符串加上随机数，把倾斜的数据分到不同的reduce上，由于null 值关联不上，处理后并不影响最终结果。
  - count distinct大量相同特殊值，count distinct 时，将值为空的情况单独处理，如果是计算count distinct，可以不用处理，直接过滤，在最后结果中加1。如果还有其他计算，需要进行group by，可以先将值为空的记录单独处理，再和其他计算结果进行union。

HQL与MapReduce转化



SQL Parser：Antlr定义SQL的语法规则，完成SQL词法，语法解析，将SQL转化为抽象语法树AST Tree；
Semantic Analyzer：遍历AST Tree，抽象出查询的基本组成单元QueryBlock；
Logical plan：遍历QueryBlock，翻译为执行操作树OperatorTree；
Logical plan optimizer: 逻辑层优化器进行OperatorTree变换，合并不必要的ReduceSinkOperator，减少shuffle数据量；
Physical plan：遍历OperatorTree，翻译为MapReduce任务；
Logical plan optimizer：物理层优化器进行MapReduce任务的变换，生成最终的执行计划

Hive元数据

Hive中元数据需要经常更新，不适合存在HDFS中，所以将其存在关系型数据库中，元数据信息包括：存在的表、表的列、权限和更多的其他信息。

与关系型数据库区别

比较项	SQL	HiveQL
ANSI SQL	支持	不完全支持
更新	UPDATE\INSERT\DELETE	insert OVERWRITE\INTO TABLE
事务	支持	不支持
模式	写模式	读模式
数据保存	块设备、本地文件系统	HDFS
延时	低	高
多表插入	不支持	支持
子查询	完全支持	只能用在From子句中
视图	Updatable	Read-only
可扩展性	低	高
数据规模	小	大
....	.....	.....

Serde

Serde方式可以利用正则表达式读取数据，不必要的数据不显示出来

例：

```
# 原数据
```

```
192.168.57.4 - - [29/Feb/2019:18:14:35 +0800] "GET /bg-upper.png HTTP/1.1" 304
-
192.168.57.4 - - [29/Feb/2019:18:14:35 +0800] "GET /bg-nav.png HTTP/1.1" 304 -
192.168.57.4 - - [29/Feb/2019:18:14:35 +0800] "GET /asf-logo.png HTTP/1.1" 304
-

# 操作
--创建表
  CREATE TABLE logtbl (
      host STRING,
      identity STRING,
      t_user STRING,
      time STRING,
      request STRING,
      referer STRING,
      agent STRING)
    ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
    WITH SERDEPROPERTIES (
      "input.regex" = "([^ ]*) ([^ ]*) ([^ ]*) \\[(.*)\\] \"(.*)\" (-|[0-9]*)
(-|[0-    9]*)"
    )
  STORED AS TEXTFILE;
--加载数据
  load data local inpath '/root/data/log' into table logtbl;
--查询操作
  select * from logtbl;

# 现在数据 不包括 ‘【】和“ ’
192.168.57.4  -  -  29/Feb/2019:18:14:35 +0800  GET /bg-upper.png HTTP/1.1
  304 -
192.168.57.4  -  -  29/Feb/2019:18:14:35 +0800  GET /bg-nav.png HTTP/1.1 304
  -
192.168.57.4  -  -  29/Feb/2019:18:14:35 +0800  GET /asf-logo.png HTTP/1.1
  304 -
```

Hive语句转化

满足下面两点，Hive不会转化为MapReduce

- select仅支持本表字段
- where仅对本表字段过滤

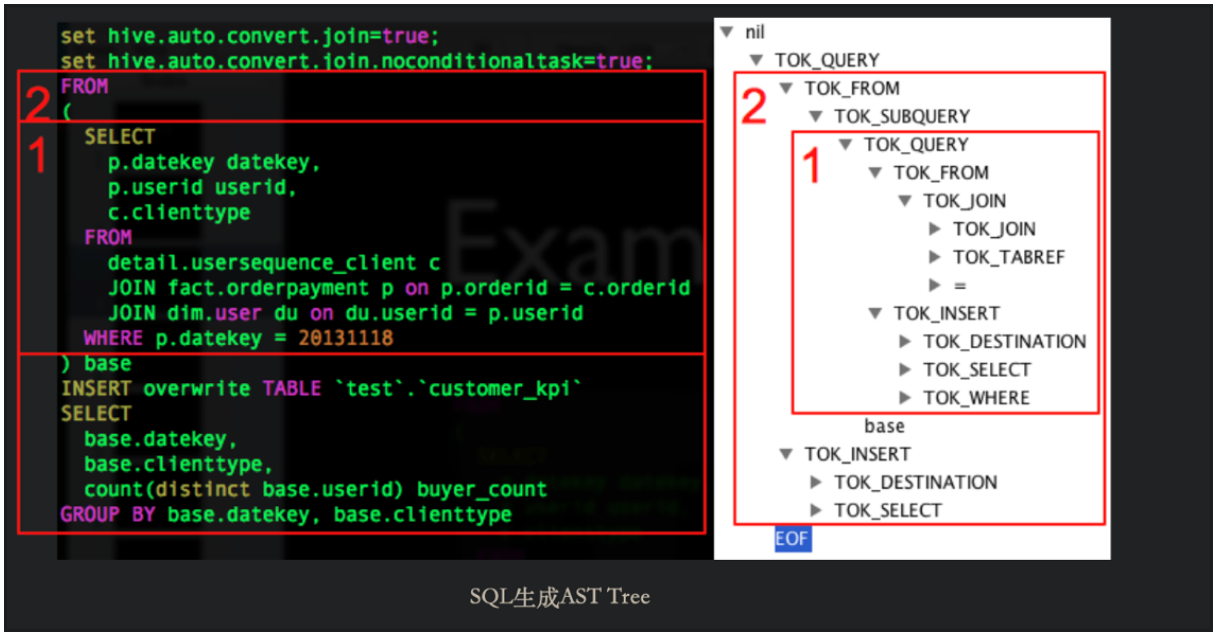
Hive排序策略

- Order By：全表排序，只允许一个Reduce处理
- Sort By：对单一Reduce排序
- Distribute：分区排序
- Cluster By：相当于 Sort By + Distribute By

SQL转化为MapReduce过程

- Antlr解析语法，SQL转化为AST Tree
- 遍历AST Tree，找出基本单元QueryBook
- 遍历QueryBook，翻译为执行操作树OperatorTree
- 逻辑优化，合并不需要的ReduceOperator，减少shuffle
- 遍历OperatorTree，翻译为MapReduce任务
- 物理优化

Antlr语法解析



SQL生成AST Tree

组成基本单元QueryBook

AST Tree生成QueryBlock的过程是一个递归的过程，先序遍历AST Tree，遇到不同的Token节点，保存到相应的属性中，最后生成两个QB对象，代表两个子查询，外层和内层

OperatorTree

遍历QB，将QB分解成【join、Groupby、OrderBy、Scan。。。】等操作，将这些操作生成操作树

逻辑优化

名称	作用
② SimpleFetchOptimizer	优化没有GroupBy表达式的聚合查询
② MapJoinProcessor	MapJoin，需要SQL中提供hint，0.11版本已不用
② BucketMapJoinOptimizer	BucketMapJoin
② GroupByOptimizer	Map端聚合
① ReduceSinkDeDuplication	合并线性的OperatorTree中partition/sort key相同的reduce
① PredicatePushDown	谓词前置
① CorrelationOptimizer	利用查询中的相关性，合并有相关性的Job，HIVE-2206
ColumnPruner	字段剪枝

通过分析，利用查询中的相关性，合并Job，让Job尽量干更多的事，减少Shuffle数据传输量

- PredicatePushDown谓词前置：断言判断改变执行流程，提前某操作

OperatorTree生成MapReduceJob过程

物理优化

HQL排序

Order by：一个reducer，保证全局有序，很慢，如果是严格模式，需要加limit限制

sort by：多个reducer，保证reducer内有序，不能保证全局有序，

distribute by：相同或相同范围内的key值分发到一个reducer处理，不能保证reducer有序，可以和sort by配合使用

cluster by：如果sort和distribute是相同字段，可用cluster，可以保证数据全局有序

HQL Join

- Hive支持MySQL各种连接
- Hive连接必须用join on，如果用join where会先计算笛卡尔积，再过滤结果，外连接时还可能丢失数据
- 进行多表Join时，Hive会放在同一个map / reduce上，尽量先小表在大表，hive会对小表进行缓存，效率提升

  left semi join：
  - 右边的表，过滤条件只能在on中设置，
  - 查询结果只包含左表数据，所以只能select左表列
