千锋教育
www.mobiletrain.org

千 锋 强 力 推 出

逆战班

停课不停学　品质不打折

鼓励全国学子在疫情中坚持学习

好程序员
从平凡到卓越，为梦想而拼搏
GoodProgrammer.org

好程序员大数据学院出品

Spark 阶段知识串讲之SparkSQL

DESIGN BY Goodprogrammer

第三章

Spark SQL 回顾

第三部分 Spark SQL 回顾

概述 →

1、RDD、DataFrame&Dataset的区别与联系
2、SparkSession 与 SparkContext的区别与联系
3、Dataset、DataFrame常见的Action操作有哪些
4、Dateset、DataFrame常见的Transformation有哪些
5、简述Row、Schema
6、SQL语句执行过程
7、Spark SQL中join的实现方式

1、RDD、DataFrame & Dataset 的区别与联系

共性：

- RDD、DataFrame、Dataset都是spark平台下的分布式弹性数据集，为处理超大型数据提供便利；
- 三者都有惰性机制。在进行创建、转换时，不会立即执行;只有在遇到Action时，才会开始遍历运算。如果代码里面仅有创建、转换，后面没有任何Action算子，代码不会执行；
- 三者都有partition的概念，进行缓存操作、还可以进行检查点操作；
- 三者有许多相似的函数，如map、filter，排序等；
- 在对DataFrame和Dataset进行操作时，很多情况下需要 spark.implicits._ 进行支持；

区别：

- DataFrame每一行的类型固定为Row，只有通过解析才能获取各个字段的值；
- DataFrame与Dataset均支持spark sql的操作，比如select，groupBy之类，还能注册临时视图，进行sql语句操作；
- DataFrame与Dataset支持一些特别方便的保存方式，比如保存成csv，可以带上表头，这样每一列的字段名一目了然；

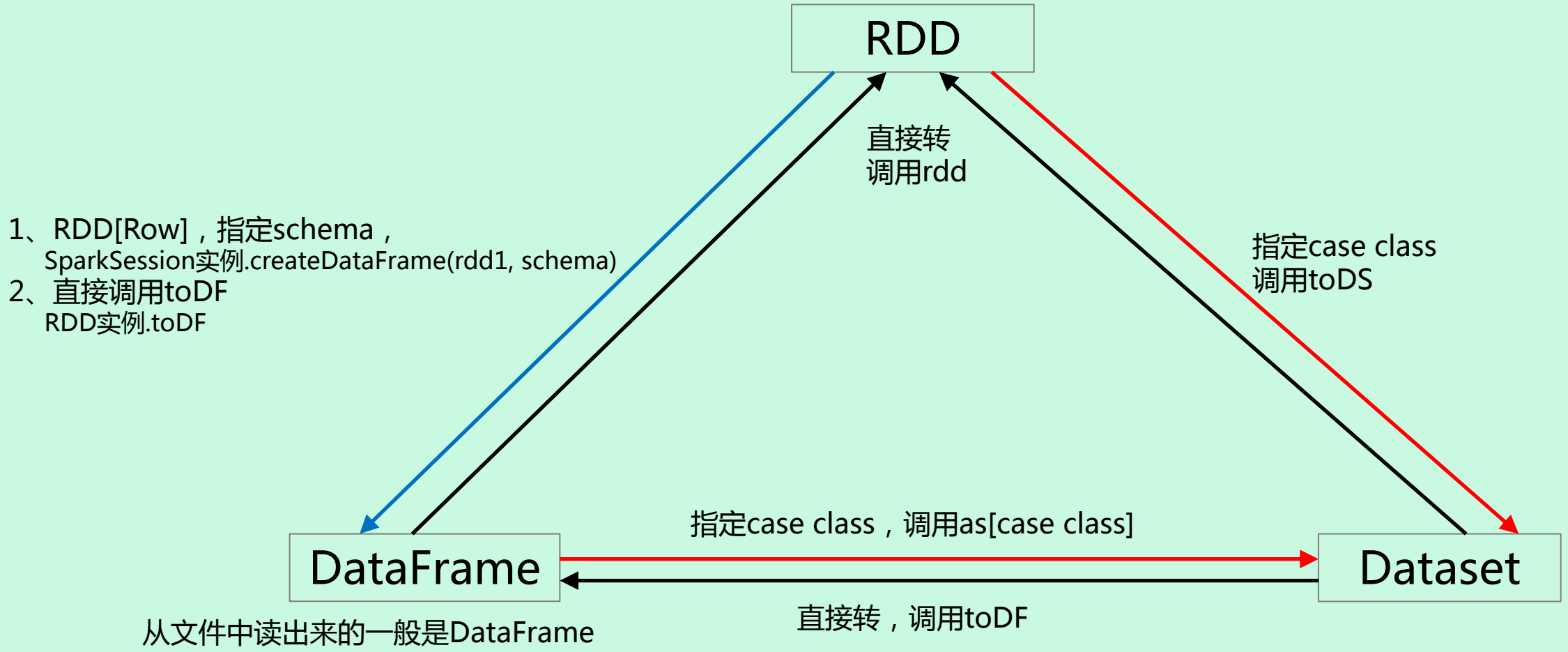
简单的说：

DataFrame = RDD[Row] + schema

Dataset = RDD[case class].toDS

1、RDD、DataFrame & Dataset 的区别与联系（续）

三者之间的转换：



2、SparkSession 与 SparkContext的区别与联系

- SparkSession 是 Spark 2.0中新的入口点；
- SparkSession内部封装了sparkContext，计算实际上是由sparkContext完成的。
- SparkSession实质上是SQLContext和HiveContext的组合；

```
class HiveContext private[hive](_sparkSession: SparkSession)
extends SQLContext(_sparkSession) with Logging {
...
```

- 在SQLContext和HiveContext上可用的API在SparkSession上同样是可以使用；

3、Dataset、DataFrame常见的Action操作有哪些

Action:

- show
- collect、collectAsList
- head、first、take、takeAsList
- printSchema
- explain (将物理计划打印到控制台以进行调试)
- reduce、count

4、Dateset、DataFrame常见的Transformation有哪些

- map、flatMap、filter
- select、selectExpr、drop、withColumn(通过添加列或替换已有的列来返回新的数据集同样的名字)
- join
- where
- groupBy、agg(在整个数据集上聚合，不包含组)
- orderBy、sort
- union、intersect(求交集)、except（求差集）

5、简述Row、Schema

Row是一个泛化的无类型JVM object；

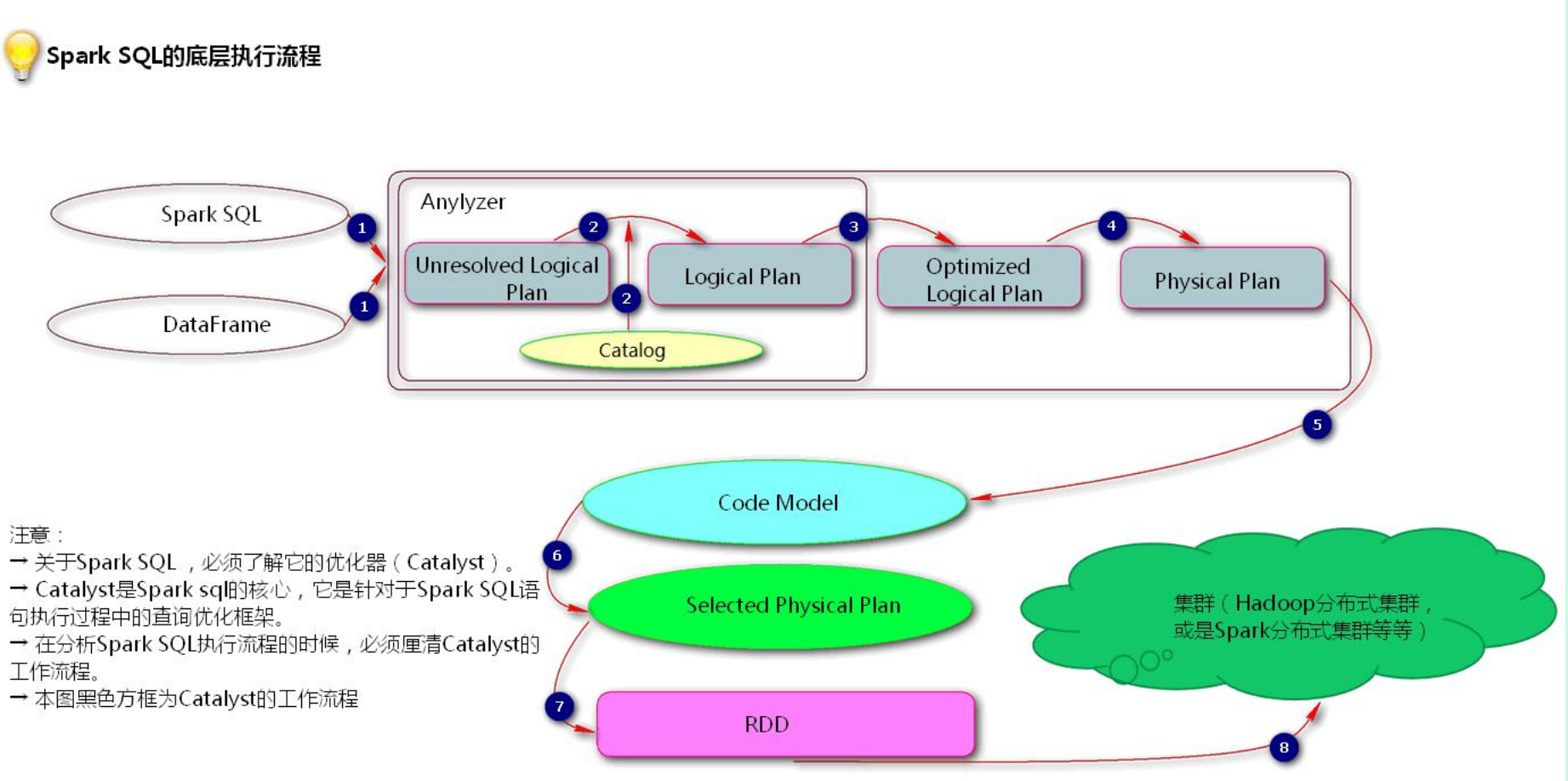
```
import org.apache.spark.sql.Row
val row1 = Row(1,"abc", 1.2)
// Row 的访问方法
row1(0)
row1(1)
row1(2)
row1.getInt(0)
row1.getString(1)
row1.getDouble(2)
```

Schema是 ，每列的名称和数据类型；DataFrame 中的数据结构信息。通过Schema可以知道数据集中包含哪些列，这些列的数据类型是什么。

6、SQL语句执行过程可分为哪几步

- 提交SQL，解析器（ SqlParser）将SQL文本解析成逻辑计划(parsed Logical Plan AST语法树 ，而这颗语法树是(Unresolved Logical Plan)）；
- 分析器（analyzer ）结合Catalog（存储了表信息）对逻辑计划做进一步分析，验证表是否存在，操作是否支持等（生成 Analyzer Logical Plan AST语法树，这颗树是（resolved Logical Plan））；
- 优化器（optimizer）对分析器分析的逻辑计划做进一步优化，如将过滤逻辑下推到子查询，子查询共用等；（生成 optimized LogicalPlan）
- Planner再将优化后的逻辑计划根据预先设定的映射逻辑转换为物理执行计划（PhysicalPlan）；
- 物理执行计划做RDD计算，最终向用户返回查询数据。

6、SQL语句执行过程可分为哪几步（续）



7、join的实现有哪几种方式

SparkSQL提供了全面的join支持，并在内部实现上做了很多优化，了解join的实现将有助于了解应用程序的运行轨迹。

SparkSQL来说有3种Join的实现，每种Join对应的不同的应用场景(SparkSQL自动决策使用哪种实现方式)：

- Broadcast Hash Join：适合一张很小的表和一张大表进行Join；
- Shuffle Hash Join：适合一张小表(比上一个大一点)和一张大表进行Join；
- Sort Merge Join：适合两张大表进行Join；

前两者都是基于Hash Join的，只不过Hash Join之前需要先shuffle还是先brocadcast。

参考：https://blog.csdn.net/baichoufei90/article/details/89609999

7、join的实现有哪几种方式（续）

Hash Join

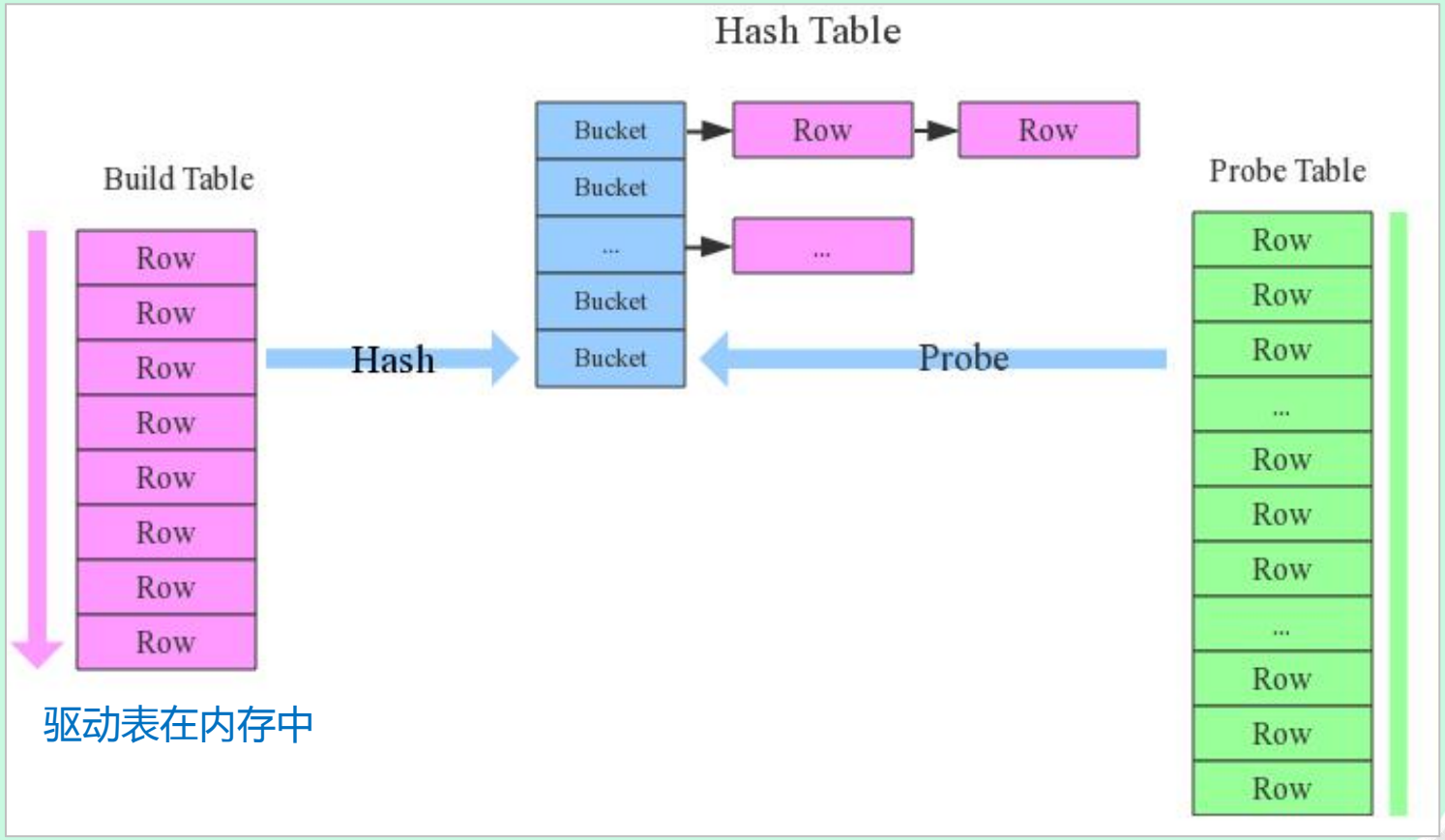
- 小表会被作为驱动表(Build Table)，加载到内存中【驱动表不能太大，否则不能加载到内存中】
- 大的表会被作为探测表(Probe Table)，逐行遍历

总结→

hash join是传统数据库中的单机join算法，在分布式环境下需要经过一定的分布式改造，说到底就是尽可能利用分布式计算资源进行并行化计算，提高总体效率。hash join分布式改造一般有两种经典方案：

①broadcast hash join

②shuffler hash join



7、join的实现有哪几种方式（续）

Broadcast Hash Join的步骤：

①broadcast阶段：将小表广播分发到大表所在的所有主机。广播算法可以有很多，最简单的是先发给driver，driver再统一分发给所有executor；要不就是基于bittorrete的p2p思路

②hash join阶段：在每个executor上执行单机版hash join，小表映射，大表试探

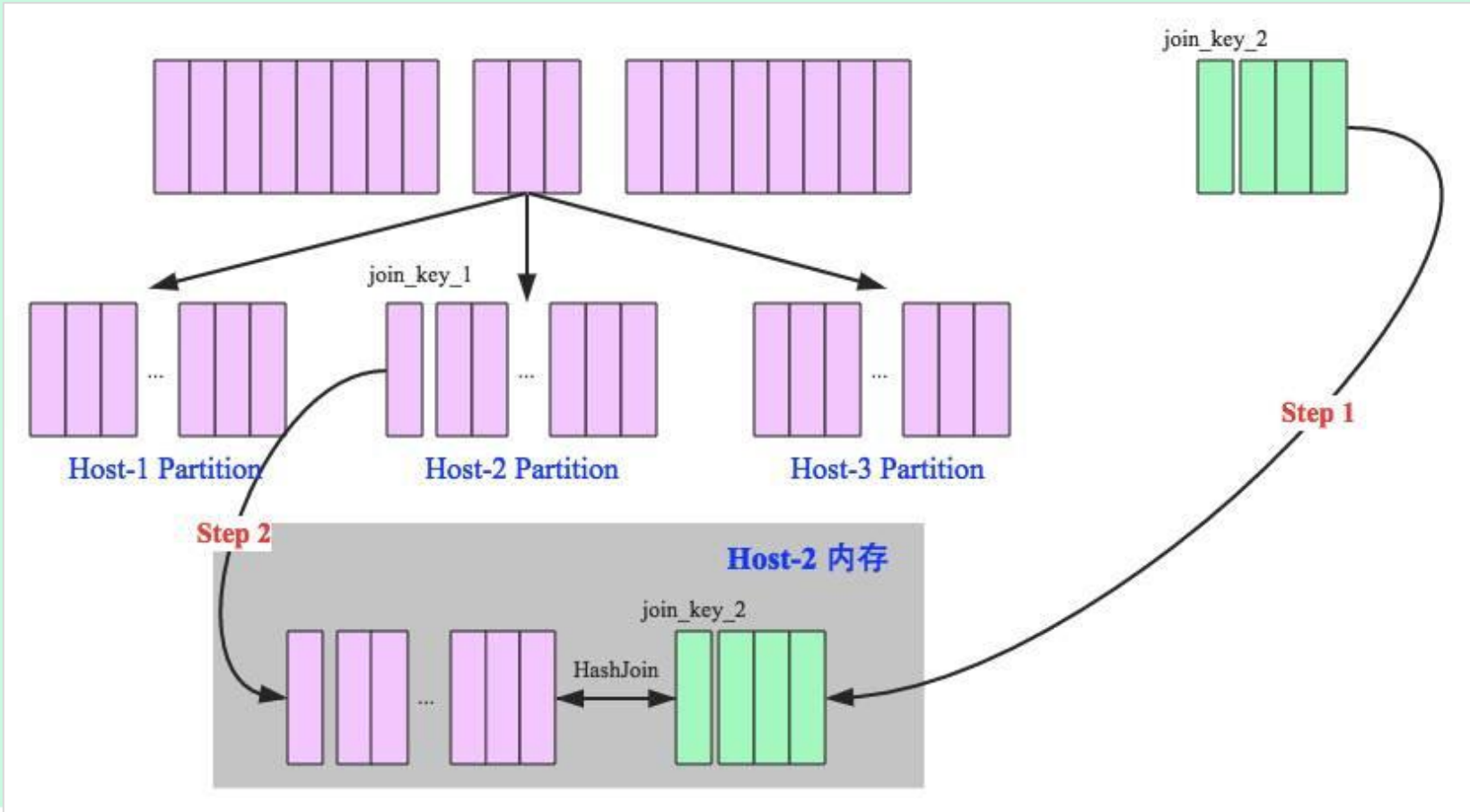
broadcast hash join→

将其中一张小表广播分发到另一张大表所在的分区节点上，分别并发地与其上的分区记录进行hash join。broadcast适用于小表很小，可以直接广播的场景

Broadcast Hash Join的条件有以下几个：

1、被广播的表需要小于spark.sql.autoBroadcastJoinThreshold 所配置的信息，默认是10M；

2、基表不能被广播，比如left outer join时，只能广播右表；



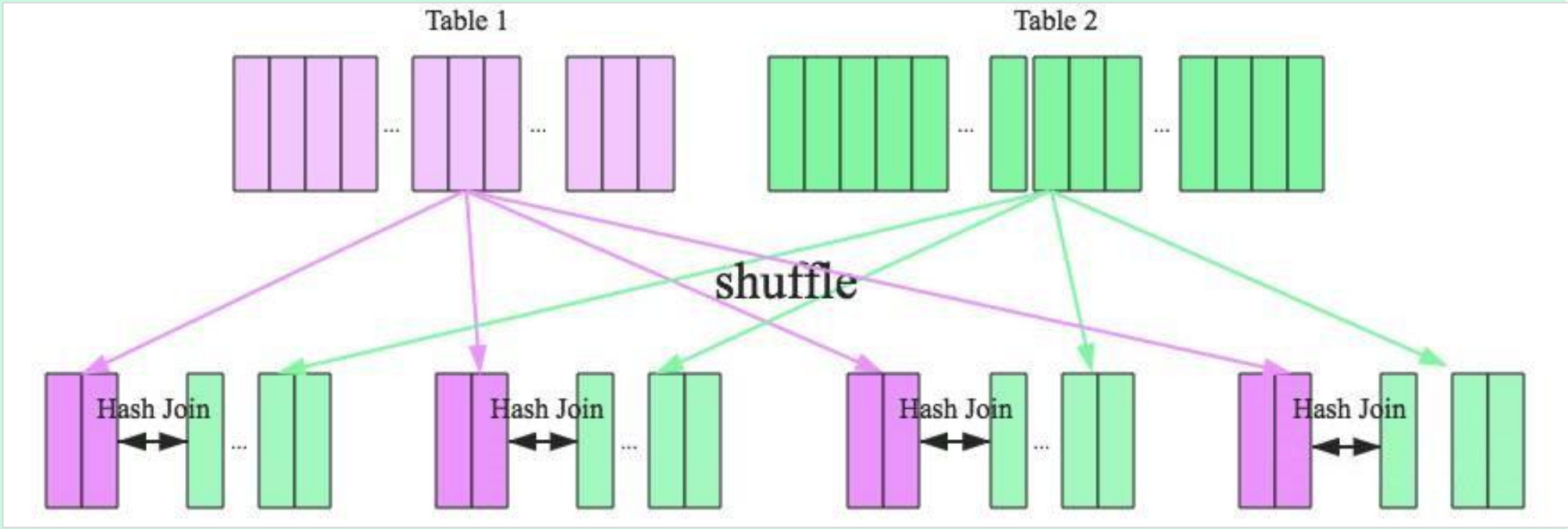
7、join的实现有哪几种方式（续）

shuffle hash join也可以分为两步：

①shuffle阶段：分别将两个表按照join key进行分区，将相同join key的记录重分布到同一节点，两张表的数据会被重分布到集群中所有节点。这个过程称为shuffle；
②hash join阶段：每个分区节点上的数据单独执行单机hash join算法（小表映射，大表试探）。

shuffler hash join→

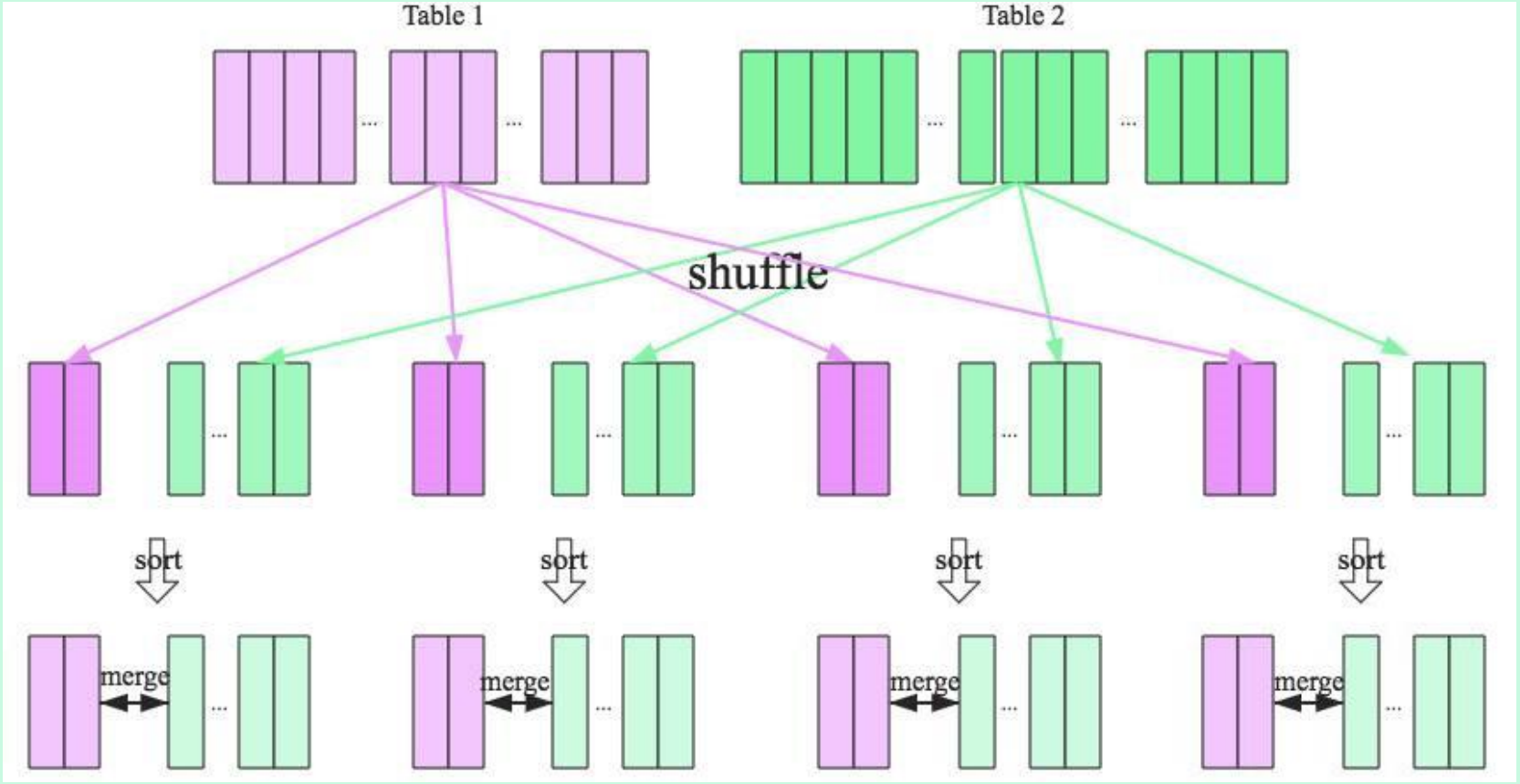
一旦小表数据量较大，此时就不再适合进行广播分发。这种情况下，可以根据join key相同必然分区相同的原理，将两张表分别按照join key进行重新组织分区，这样就可以将join分而治之，划分为很多小join，充分利用集群资源并行化。



7、join的实现有哪几种方式（续）

Sort Merge Join→

- 当两个表都非常大时，SparkSQL采用Sort Merge Join的方式进行表连接。这种方式需要在join前将数据进行排序。
- 首先将两张表按照join key进行重新shuffle，保证join key值相同的记录会被分在相应的分区，分区后对每个分区内的数据进行排序，排序后再对相应的分区内的记录进行连接。



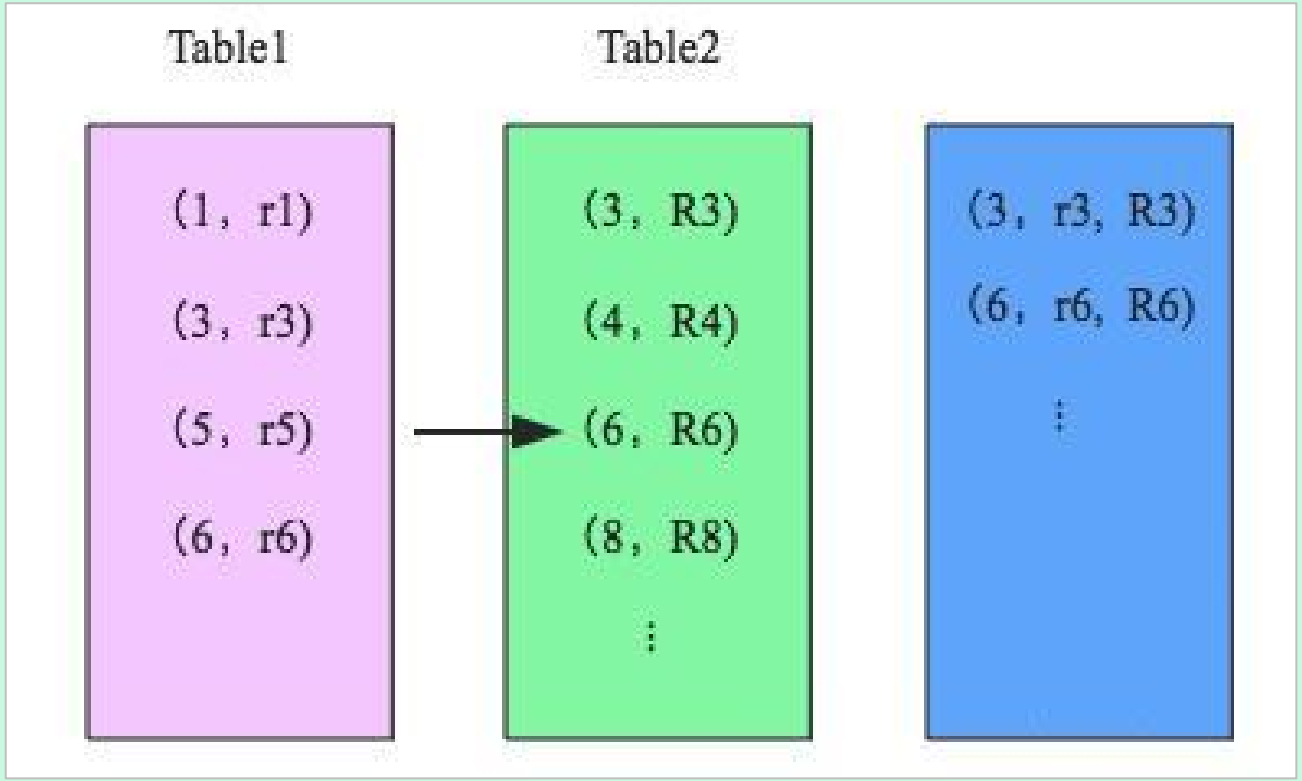
7、join的实现有哪几种方式（续）

Sort Merge Join的执行步骤：

①shuffle阶段：将两张大表根据join key进行重新分区，两张表数据会分布到整个集群，以便分布式并行处理；

②sort阶段：对单个分区节点的两表数据，分别进行排序；

③merge阶段：对排好序的两张分区表数据执行join操作。join操作很简单，分别遍历两个有序序列，碰到相同join key就merge输出。



深究：

sort-merge join的代价并不比shuffle hash join小，反而是多了很多。那为什么SparkSQL还会在两张大表的场景下选择使用sort-merge join算法呢？

→ 这和Spark的shuffle实现有关，目前spark的shuffle实现都使用sort-based shuffle算法，因此在经过shuffle之后partition数据都是按照key排序的。因此理论上可以认为数据经过shuffle之后是不需要sort的，可以直接merge。

→ 可以明确每种Join算法都有自己的适用场景，数据仓库设计时最好避免大表与大表的join查询，SparkSQL也可以根据内存资源、带宽资源适量将参数spark.sql.autoBroadcastJoinThreshold调大，让更多join实际执行为broadcast hash join。

Spark SQL面试题

1、 SparkSQL和Hive区别

① SparkSQL实际上是使用sql解释器将sql语句转化为一系列的RDD转换与落地过程，而hive SQL则是将sql语句转换成多个MR job顺序执行

② 受限于MR模型，hive sql只能将sql语句按map和reduce的粒度切分，无法对其中的更细粒度的input、sort、shuffle等进行自由组合;相反spark RDD是一系列细粒度算子的转换流程，因此可以方便地对各个部分进行细致的优化和自由组合。

③ Spark RDD尽管会在不得已时将中间结果落地磁盘(shuffle)，但它不会将RDD的转换结果主动落地磁盘，因此在多个RDD之间变换时不需要重新进行IO；而HQL在多个job间必须不断重复IO过程

④ 由于以上原因，SparkSql在实战时对子查询的执行效率远比HQL高，这也使得它很适合迭代型的sql统计

→ 举例

HQL的窗口函数的发生时间在having之后，因此它的窗口函数可以统计group聚合后的数据，相反Spark没有job之间的先后关系，因此它的窗口函数只能使用聚合前的数据。这在统计排名函数时可以很直观的看出来：

--以下统计各省用户登录次数前3

HQL(下述HQL不能在sparkSQL中运行，会提示count(1)不属于可用的字段):

```
select * from (
select id,province,time,row_number() over(
        partition by province
        order by count(1) DESC
        ) as rank
from table
group by id,province
)t
where rank<=3
```

Spark SQL面试题

1、 SparkSQL和Hive区别（续）

而下述语句SparkSQL:

```
select
   id,
   province,
  row_number() over(partition by province order by total DESC
) as rank
from table
having rank<=3
```

在HQL中不能运行，会提示having字段错误

→ 总结：

推荐采用DSL+SQL的混合写法书写SparkSQL，将普通的转换算子和聚合算子用DSL风格书写，将窗口函数用SQL风格书写，可以避开二者的不同之处。

Spark SQL面试题

2、 DF和DS的区别

DataFrame = DataSet[Row]

DataSet是强类型JVM object的集合，DataFrame则是由每行弱类型的JVM object组成，二者都是按列存储，前者是结构化数据集，后者则是半结构化数据集

3、RDD与DataFrame区别，什么场景用RDD什么场景用DataFrame？

→ 区别：

RDD是分布式的不可变的抽象的数据集，比如，RDD[Person]是以Person为类型参数，但是，Person类的内部结构对于RDD而言却是不可知的。DataFrame是以RDD为基础的分布式的抽象数据集，也就是分布式的Row类型的集合（每个Row对象代表一行记录），提供了详细的结构信息，即Schema信息。Spark SQL可以清楚地知道该数据集中包含哪些列、每列的名称和类型。

→ 应用场景：

RDD的使用场景：

需要使用low-level的transformation和action来控制你的数据集；

得到的数据集是非结构化的，比如，流媒体或者文本流；

想使用函数式编程来操作你得数据，而不是用特定领域语言（DSL）表达；

不在乎schema，比如，当通过名字或者列处理（或访问）数据属性不在意列式存储格式；

放弃使用DataFrame和Dataset来优化结构化和半结构化数据集；

Spark SQL面试题

3、RDD与DataFrame区别，什么场景用RDD什么场景用DataFrame（续）？

→ 应用场景（续）：

DataFrame的使用场景：

想使用丰富的语义，high-level抽象，和特定领域语言API，那你可DataFrame或者Dataset；

处理的半结构化数据集需要high-level表达， filter，map，aggregation，average，sum ，SQL 查询，列式访问和使用lambda函数，那你可DataFrame或者Dataset；

想利用编译时高度的type-safety，Catalyst优化和Tungsten的code生成，那你可DataFrame或者Dataset；

想统一和简化API使用跨Spark的Library，那你可DataFrame或者Dataset；

如果是一个R使用者，建议优先选用DataFrame或者Dataset；

如果是一个Python使用者，建议优先选用DataFrame或者Dataset。

4、Spark SQL中的自定义函数

Spark SQL支持三种自定义函数，UDF、UDAF（用户自定义聚合函数）、UDTF（用户自定义生成函数,User-Defined Table-Generating Functions）

UDAF和UDTF都需要继承对应的自定义函数类，实现相应的抽象方法才可以使用

UDF则可以在spark.udf.register方法中使用函数直接注册使用。