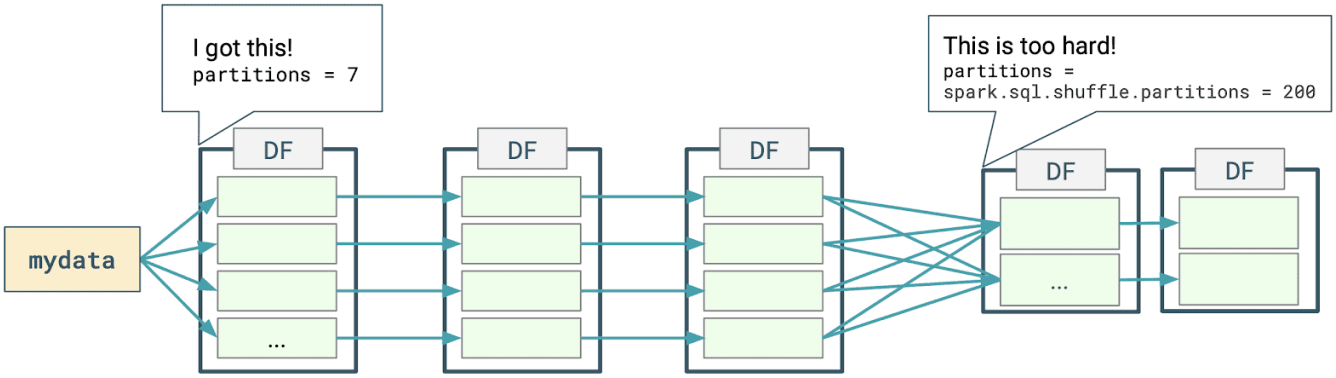
Apache Spark 3.0 是如何提高 SQL 工作负载的性能

在几乎所有处理复杂数据的领域，Spark 已经迅速成为数据和分析生命周期团队的事实上的分布式计算框架。Spark 3.0 最受期待的特性之一是新的自适应查询执行框架(Adaptive Query Execution，AQE)，该框架解决了许多 Spark SQL 工作负载遇到的问题。AQE 在2018年初由英特尔和百度组成的团队最早实现。AQE 最初是在 Spark 2.4 中引入的， Spark 3.0 做了大量的完善和优化。另外，Spark 3.0 的动态分区修剪（Dynamic Partition Pruning，DPP）也为一些工作负载带来了性能提升。本文将详细介绍这两种优化的原理。

Adaptive Query Execution

Catalyst 早期实现的缺陷

下图表示了使用 DataFrames 执行简单的分组计数查询时发生的分布式处理：



如果想及时了解Spark、Hadoop或者HBase相关的文章，欢迎关注微信公众号：过往记忆大数据

Spark 在第一阶段（stage）确定了适当的分区数量，但对于第二阶段，使用默认的分区数 200。也就是 spark.sql.shuffle.partitions 参数的默认值。这个默认值有以下几个问题：

- 200个分区值不太可能是理想的分区数量，而分区数是影响性能的关键因素之一；
- 如果将上图的第二个阶段的数据写到磁盘，那么我们将得到200个小文件。

你可以做的事情是在执行查询之前手动设置这个属性的值，就像这样：

```
spark.conf.set("spark.sql.shuffle.partitions", "2")
```

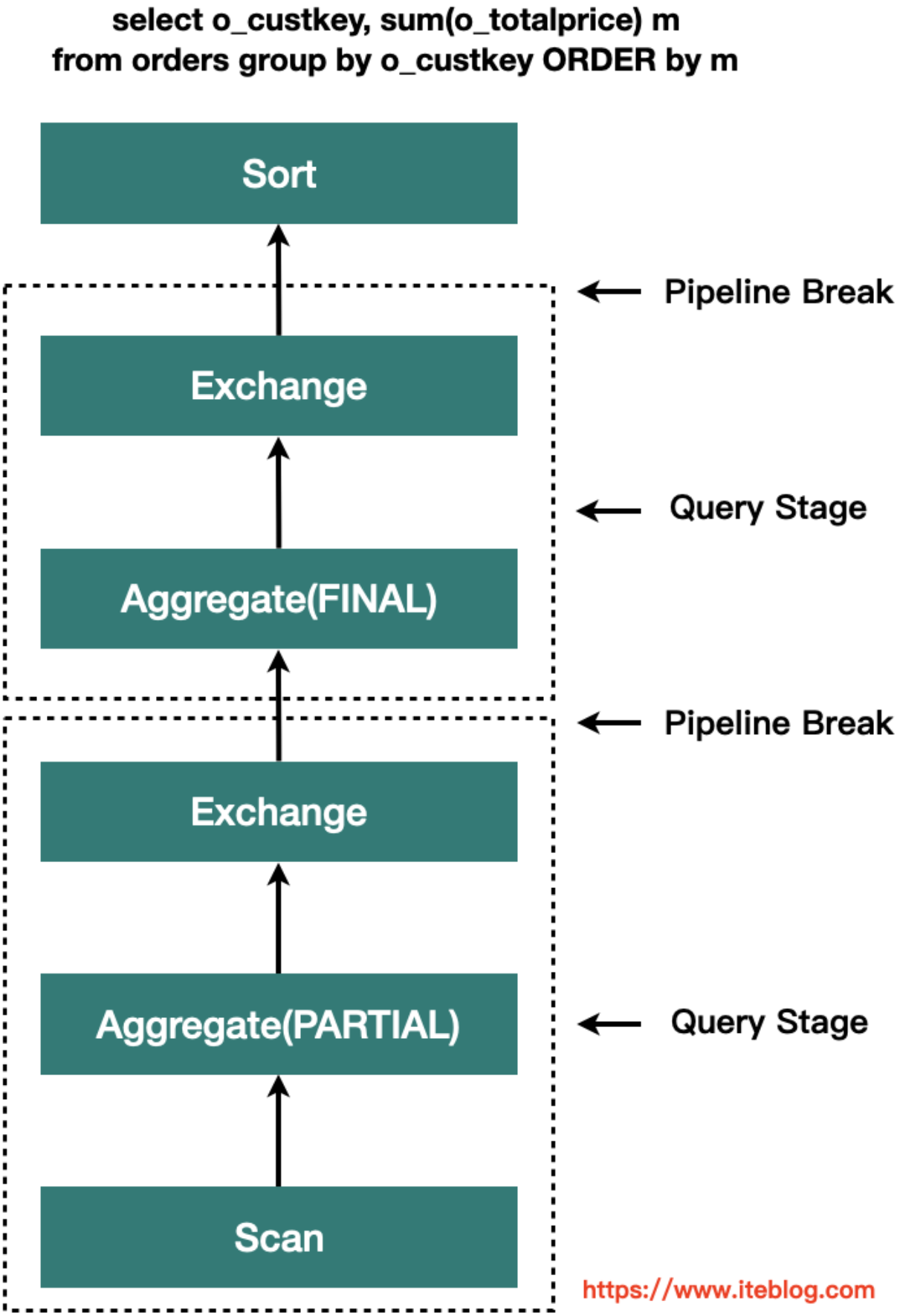
虽然我们可以手动设置分区数，但是也有以下挑战：

- 每个查询都设置这个属性是非常繁琐的；
- 随着业务的变化，之前设置的值可能会过时；
- 这个设置将应用于同一个程序里面的所有 Shuffle 操作中。

在上一个示例的第一个阶段之前，数据的分布和容量我们是可以知道的，Spark 可以为分区数量提供一个合理的值。然而对于第二阶段，经过第一个阶段处理后的数据大小很难准确估计，所以我们只能自己去估计。

自适应查询执行设计原理

AQE 完全基于精确的运行时统计信息进行优化，引入了 Query Stages 的概念 ，并且以 Query Stage 为粒度，进行运行时的优化，其工作原理如下所示：


select o_custkey, sum(o_totalprice) m
from orders group by o_custkey ORDER by m
Sort
Pipeline Break
Exchange
Query Stage
Aggregate(FINAL)
Pipeline Break
Exchange
Aggregate(PARTIAL)
Query Stage
Scan
https://www.iteblog.com

如果想及时了解Spark、Hadoop或者HBase相关的文章，欢迎关注微信公众号：过往记忆大数据

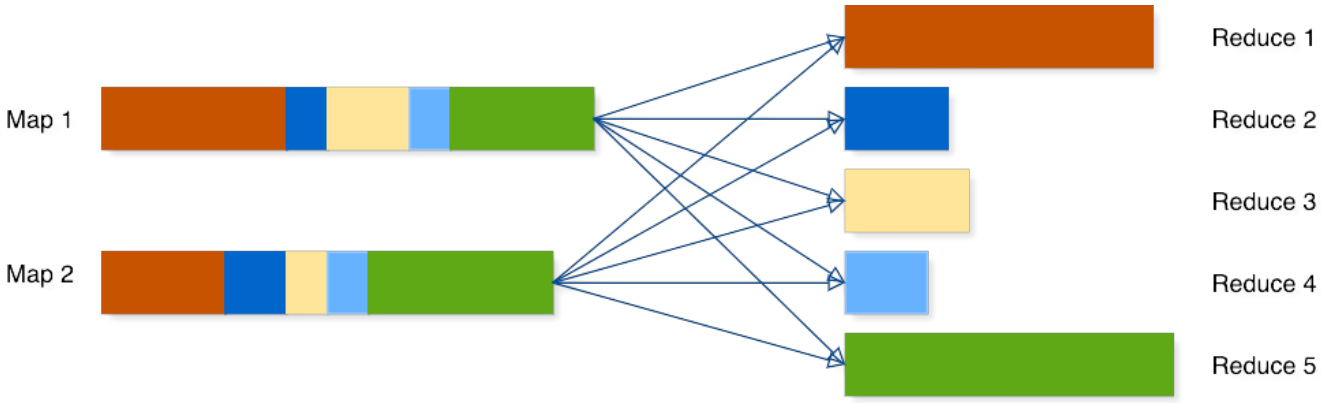
Query Stage 是由 Shuffle 或 broadcast exchange 划分的，在运行下一个 Query Stage 之前，上一个 Query Stage 的计算需要全部完成，这是进行运行时优化的绝佳时机，因为此时所有分区上的数据统计都是可用的，并且后续操作还没有开始。

AQE 可以理解成是 Spark Catalyst 之上的一层，它可以在运行时修改 Spark plan。

自适应调整分区数

这个功能从 Spark 2.4 开始就引入了，当启用 AQE 时，将自动调整 shuffle 分区的数量，不再是默认的200或手动设置的值。要使用上面的功能，需要设置 spark.sql.adaptive.coalescePartitions.enabled 、spark.sql.adaptive.enabled 以及 spark.sql.adaptive.advisoryPartitionSizeInBytes。这时候 Spark 将会把连续的 shuffle partitions 进行合并（coalesce contiguous shuffle partitions）以减少分区数。

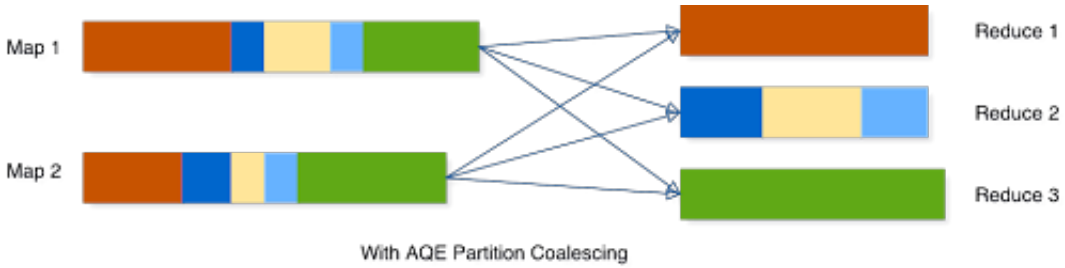
假设我们运行 SELECT max(i)FROM tbl GROUP BY j 查询，tbl 表的输入数据相当小，所以在分组之前只有两个分区。我们把初始的 shuffle 分区数设置为 5，因此在 shuffle 的时候数据被打乱到 5 个分区中。如果没有 AQE，Spark 将启动 5 个任务来完成最后的聚合。然而，这里有三个非常小的分区，为每个分区启动一个单独的任务将是一种浪费。



No AQE Partition Coalescing

如果想及时了解Spark、Hadoop或者HBase相关的文章，欢迎关注微信公众号：iteblog_hadoop

使用 AQE 之后，Spark 将这三个小分区合并为一个，因此，最终的聚合只需要执行三个任务，而不是五个。

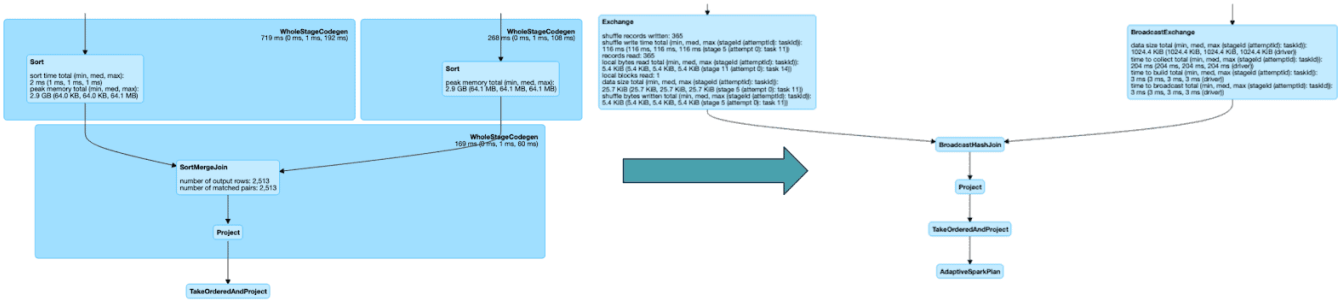


如果想及时了解Spark、Hadoop或者HBase相关的文章，欢迎关注微信公众号：iteblog_hadoop

动态将 Sort Merge Joins 转换成 Broadcast Joins

Spark 支持许多 Join 策略，其中 broadcast hash join 通常是性能最好的，前提是参加 join 的一张表的数据能够装入内存。由于这个原因，当 Spark 估计参加 join 的表数据量小于广播大小的阈值时，其会将 Join 策略调整为 broadcast hash join。但是，很多情况都可能导致这种大小估计出错，比如表的统计信息不准确等。

有了 AQE，Spark 可以利用运行时的统计信息动态调整 Join 方式，只要参与 Join 的任何一方的大小小于广播大小的阈值时，即可将 Join 策略调整为 broadcast hash join。如下图就是利用这个调整 Join 策略的。



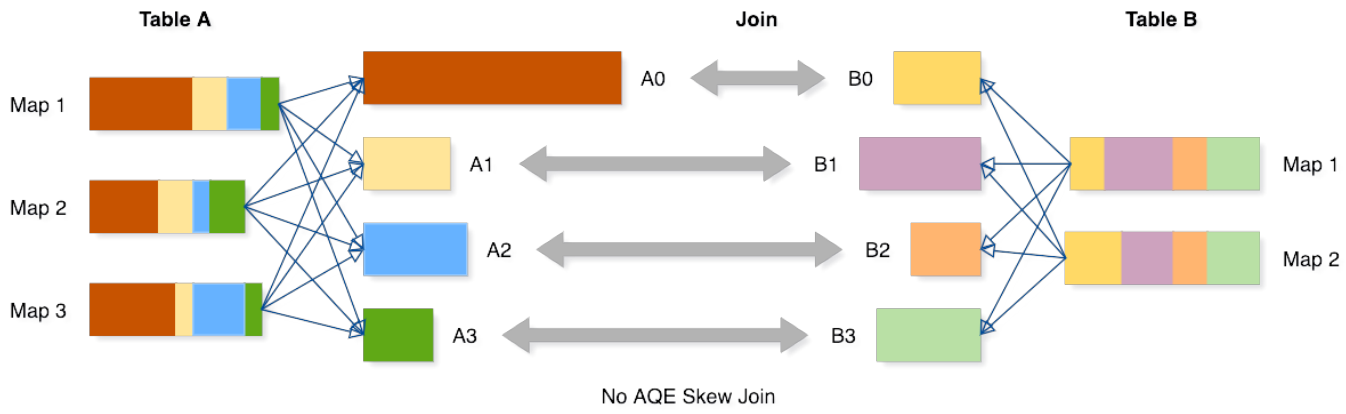
如果想及时了解Spark、Hadoop或者HBase相关的文章，欢迎关注微信公众号：过往记忆大数据

对于在运行时转换的 broadcast hash join ，我们可以进一步将常规的 shuffle 优化为本地化 shuffle来减少网络流量。

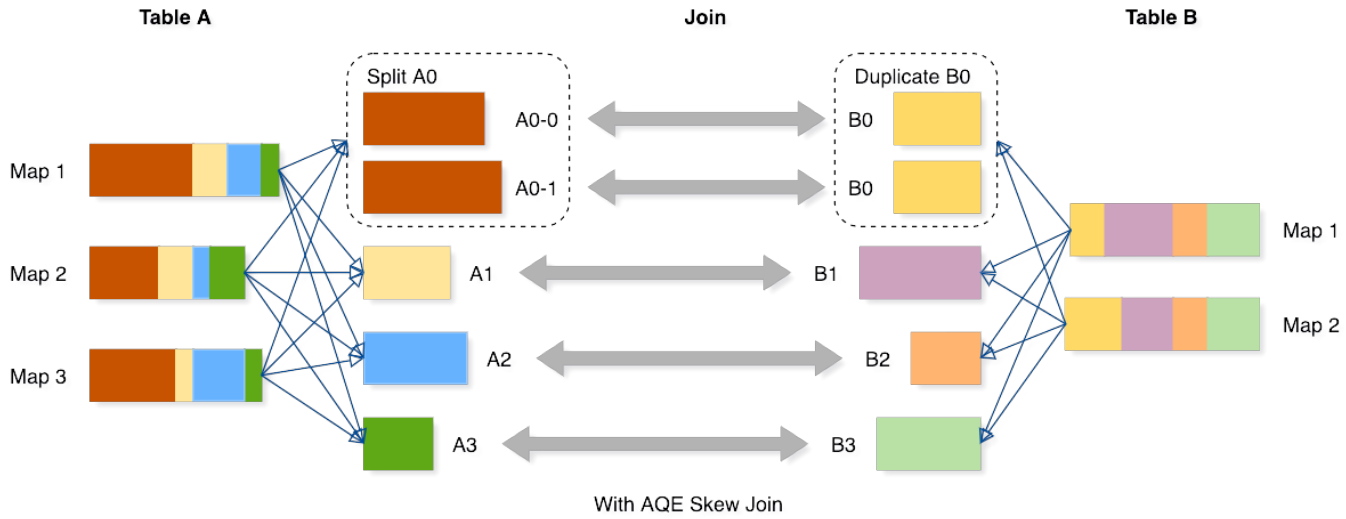
动态优化倾斜的 join

当数据在集群中的分区之间分布不均时，就会发生数据倾斜。严重的倾斜会显著降低查询性能，特别是在进行 Join 操作时。AQE 倾斜 Join 优化从 shuffle 文件统计信息中自动检测到这种倾斜。然后，它将倾斜的分区分割成更小的子分区，这些子分区将分别从另一端连接到相应的分区。

假设表 A join 表B，其中表 A 的分区 A0 里面的数据明显大于其他分区。



skew join optimization 将把分区 A0 分成两个子分区，并将每个子分区 join 表 B 的相应分区 B0。



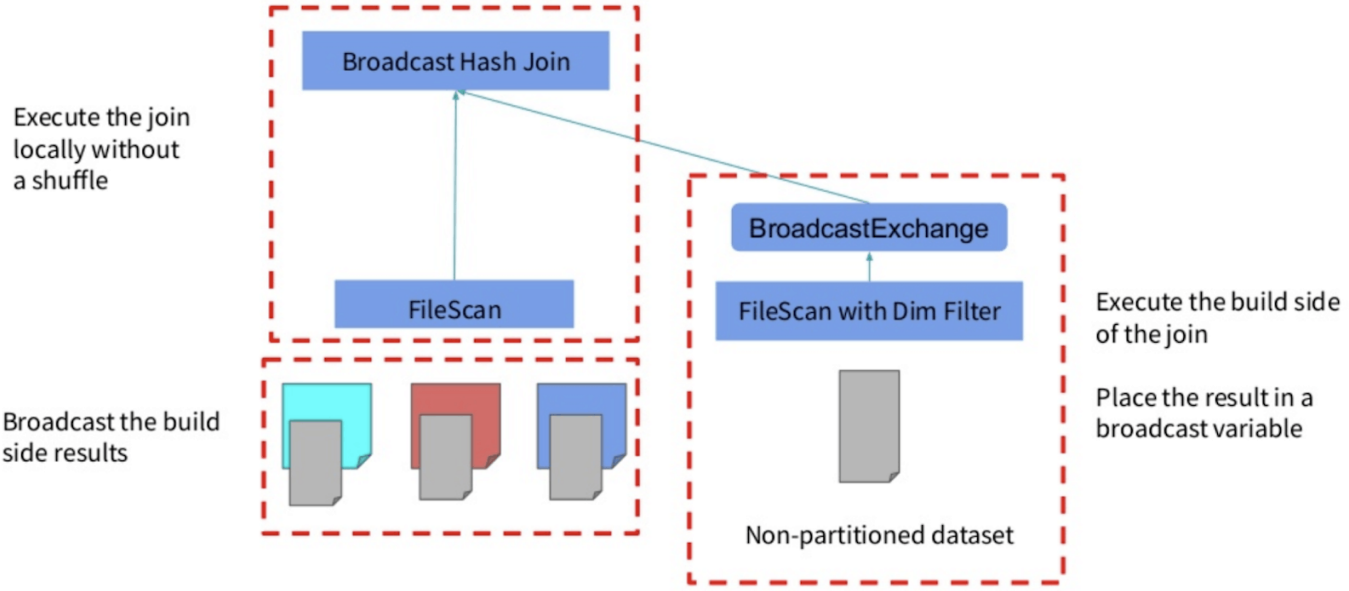
如果没有这个优化，将有四个任务运行 sort merge join，其中一个任务将花费非常长的时间。在此优化之后，将有5个任务运行 join，但每个任务将花费大致相同的时间，从而获得总体更好的性能。要启用上面的功能需要用到下面三个参数：

- spark.sql.adaptive.skewJoin.enabled ：是否启用倾斜 Join 处理；
- spark.sql.adaptive.skewJoin.skewedPartitionFactor：如果一个分区的大小大于这个数乘以分区大小的中值（median partition size），并且也大于spark.sql.adaptive.skewedPartitionThresholdInBytes 这个属性值，那么就认为这个分区是倾斜的。
- spark.sql.adaptive.skewedPartitionThresholdInBytes：判断分区是否倾斜的阈值，默认为 256MB，这个参数的值应该要设置的比 spark.sql.adaptive.advisoryPartitionSizeInBytes 大。

动态分区裁减

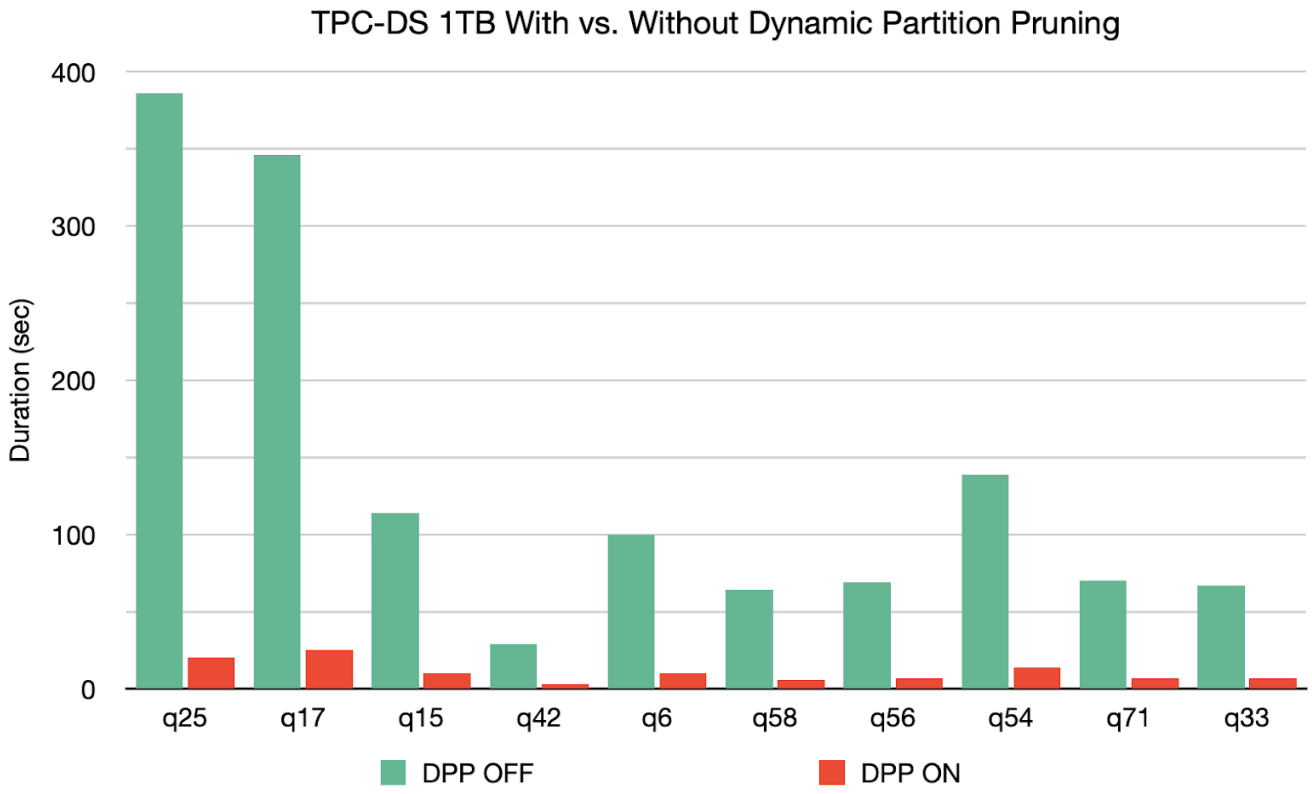
Spark 3.0 的第二个比较重要的性能优化是动态分区裁减（Dynamic Partition Pruning，简称 DPP），需要注意的是，如果开启了动态分区裁减，那么 AQE
将不会被触发。这个优化在逻辑计划和物理计划上都有实现。

- 在逻辑计划层面，通过维度表构造出一个过滤子查询，然后在扫描事实表之前加上这个过滤子查询。通过这种方式，我们在逻辑计划阶段就知道事实表需要扫描哪些分区。但是，物理计划执行起来还是比较低效。因为里面有重复的子查询，我们需要找出一种方法来消除这个重复的子查询。为了做到这一点，Spark 在物理计划阶段做了一些优化。
- 在物理计划层面，在维度表上运行上面构造的过滤，然后将结果广播到事实表端，从而达到避免扫描无用的数据效果。



如果我们将 spark.sql.optimizer.dynamicPartitionPruning.reuseBroadcastOnly 设置为 false，那么 DPP 也可以在其他类型的 Join 上运行，比如 SortMergeJoin。在这种情况下，Spark 将估计 DPP 过滤器是否确实提高了查询性能。DPP 可以极大地提高高度选择性查询的性能，例如，如果我们的查询在5年的价值数据中过滤出其中一个月的数据。

在 TPC-DS 基准测试中，102个查询中的60个得到2到18倍的加速。



如果想及时了
解Spark、Hadoop或者HBase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

借助 AQE、DPP、GPU 的支持以及 Kubernetes
的支持，性能提升的前景非常乐观，我们应该可以看到 Spark 3 将在越来越多的公司使用。

本文主要翻译自：
[How does Apache Spark 3.0 increase the performance of your SQL workloads]