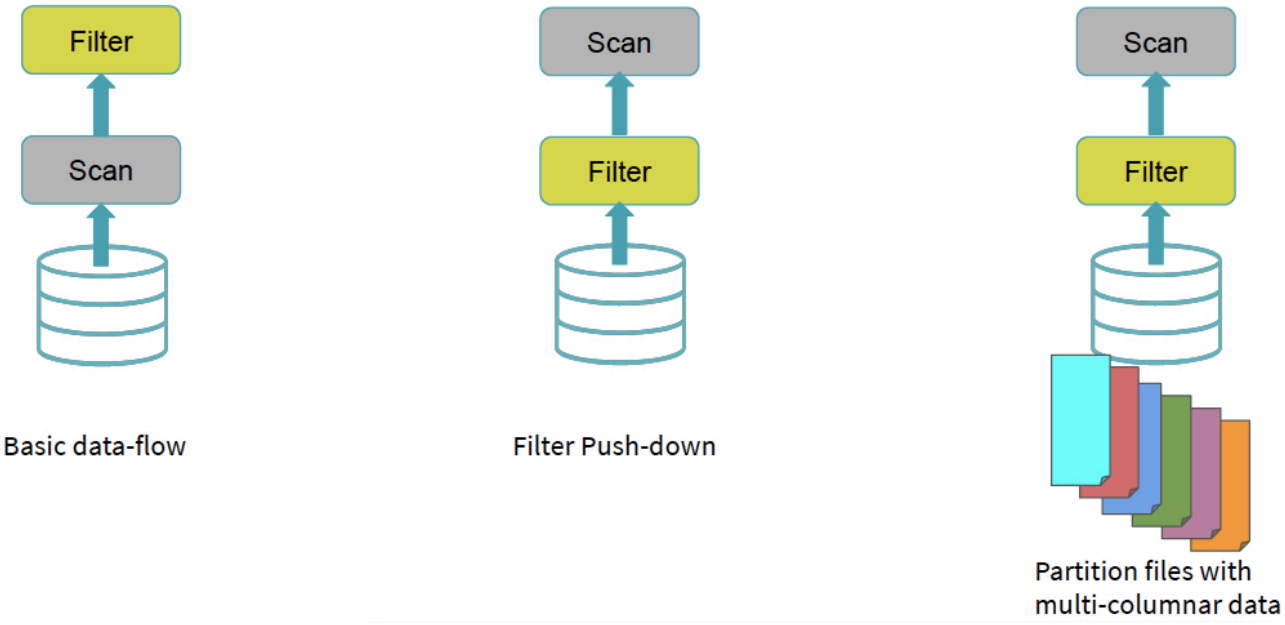
Apache Spark 3.0 动态分区裁剪（Dynamic Partition Pruning）介绍

静态分区裁剪（Static Partition Pruning）

用过 Spark 的同学都知道，Spark SQL
在查询的时候支持分区裁剪，比如我们如果有以下的查询：

SELECT * FROM Sales_iteblog WHERE day_of_week = 'Mon'

Spark 会自动进行以下的优化：



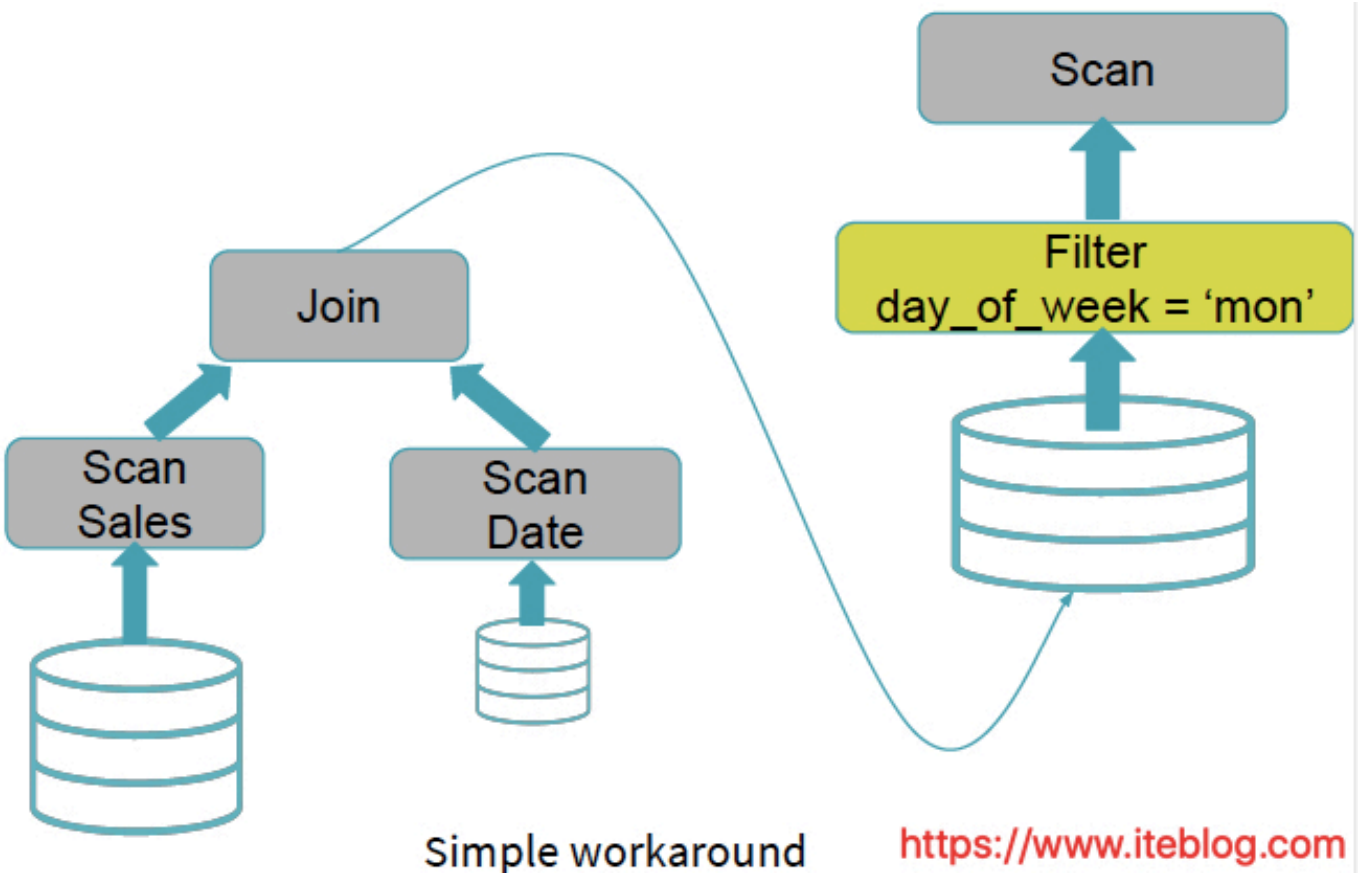
如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

从上图可以看到，Spark 在编译 SQL 的时候自动将 Filter 算子下推到数据源，也就是在 Scan
前进行了 Filter 操作，将 day_of_week = 'Mon'
的数据全部拿出来，其他数据不需要的拿出，这样 Spark SQL 中处理的数据就变少了，整个 SQL
的查询数据就会变快，这一切都是编译的时候（compile
time）进行的，所以这个叫做静态分区裁剪（Static Partition Pruning）。

上面的 SQL 查询在 Spark 进行了算子下推，已经能够满足我们的查询性能的提升。但是现实世界
数据的查询可不会都这么简单，我们来看看下面的查询语句：

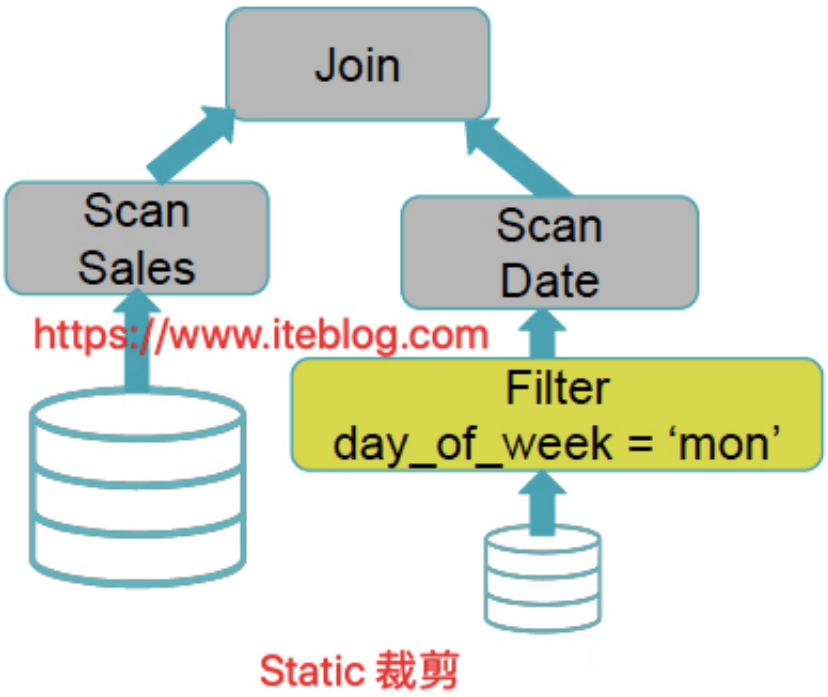
SELECT * FROM Sales JOIN Date WHERE Date.day_of_week = 'Mon'

比较差的查询引擎是这么做的：



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

从上图可以看出，查询引擎直接忽略了 Date.day_of_week = 'Mon' 这个过滤条件，上来就是两表 join，然后 join 的结果再进行过滤，这个可能导致很多无效的计算，如果 Date.day_of_week = 'Mon' 可以过滤掉大量的无用数据，肯定可以提升查询性能。在 Spark SQL
里面能够很好的处理这种情况，它会把 Date.day_of_week = 'Mon' 过滤条件下推到 Date 表的 Scan 之前进行：



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

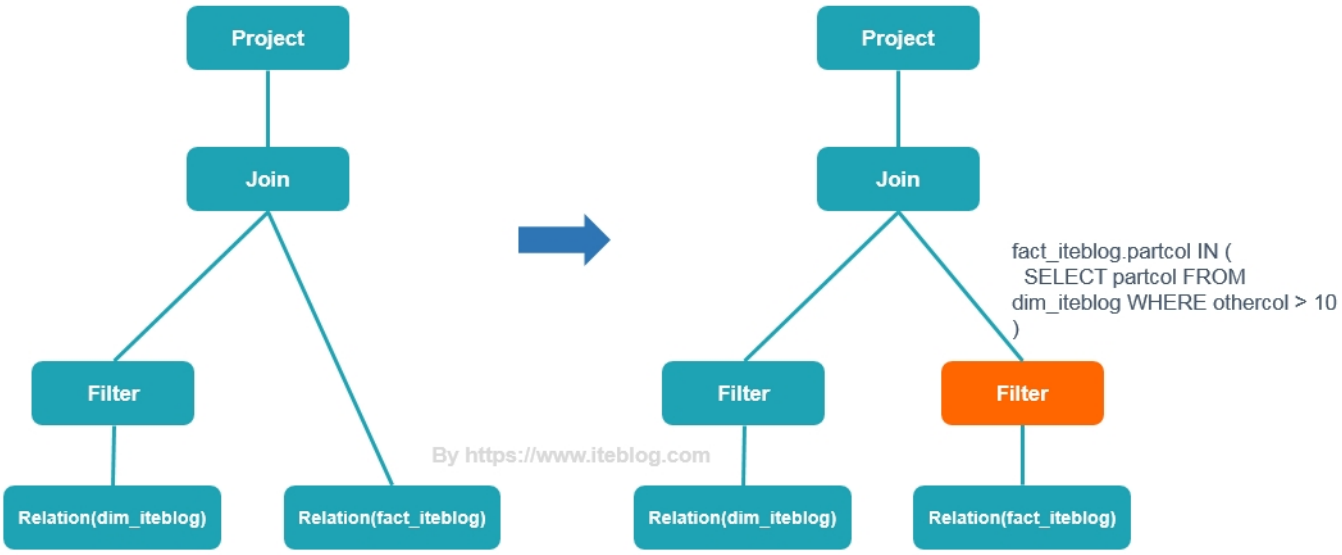
动态分区裁剪（Dynamic Partition Pruning）

上面 Spark SQL 进行的算子下推是不是不能再提升查询性能呢？有没有一种更好的方法进一步过滤掉一些无用的数据？这就是本文要介绍的动态分区裁剪。动态分区裁剪这个功能是 Spark 3.0 引入的，详见 SPARK-11150、SPARK-28888。

什么是动态分区裁剪？所谓的动态分区裁剪就是基于运行时（run
time）推断出来的信息来进一步进行分区裁剪。举个例子，我们有如下的查询：

```
SELECT * FROM dim_iteblog
JOIN fact_iteblog
ON (dim_iteblog.partcol = fact_iteblog.partcol)
WHERE dim_iteblog.othercol > 10
```

动态查询的执行计划如下：



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

从上面可以看出，拥有动态分区裁剪，Spark 能够在运行的时候先对 fact_iteblog 表的 partcol 进行了一次过滤，然后再和 dim_iteblog 表进行
Join，可想而知这个性能一般都会有提升的，特别是在 fact_iteblog
表有很多无用的数据时性能提升会非常大的。

Databricks 公司在10台配置为 i3.xlarge 的集群上进行 TPC-DS 测试，得到的结论是在 102 查询中相比 Spark 2.4 有 60 个查询的查询性能提升了 2 - 18 倍的提升。在 Query 98 的查询中，性能提升了 100 倍！

相关配置

要启用动态分区裁剪需要将 spark.sql.optimizer.dynamicPartitionPruning.enabled 参数设置为 true（默认），其他相关参数：

- spark.sql.optimizer.dynamicPartitionPruning.useStats：true（默认），When true, distinct count statistics will be used for computing the data size of the partitioned table after dynamic partition pruning, in order to evaluate if it is worth adding an extra subquery as the pruning filter if broadcast reuse is not applicable.
- spark.sql.optimizer.dynamicPartitionPruning.fallbackFilterRatio：0.5，When statistics are not available or configured not to be used, this config will be used as the fallback filter ratio for computing the data size of the partitioned table after dynamic partition pruning, in order to evaluate if it is worth adding an extra subquery as the pruning filter if broadcast reuse is not applicable.
- spark.sql.optimizer.dynamicPartitionPruning.reuseBroadcast：默认为 true，When true, dynamic partition pruning will seek to reuse the broadcast results from a broadcast hash join operation.