Spark SQL 查询 Parquet 文件的性能提升 30%，字节是如何做到的？

本文来自11月举办的 Data + AI Summit 2020（原 Spark+AI Summit），主题为《Improving Spark SQL Performance by 30%: How We Optimize Parquet Filter Pushdown and Parquet Reader》的分享，作者为字节跳动的孙科和郭俊。相关 PPT 可以关注 Java与大数据架构 公众号并回复 9912 获取。

Parquet 是一种非常流行的列式存储格式。Spark 的算子下推（pushdown filters）可以利用 Parquet 文件的统计数据（比如最大最小值统计）来过滤无用数据。另一方面，Spark 用户可以启用 Spark Parquet 的向量化读取器（vectorized reader）来批量读取 parquet 文件。这些特性大大提高了 Spark 的性能，节省了 CPU 和 IO。Parquet 是字节跳动数据仓库的默认数据格式。在实践中，字节团队的同学发现 parquet 的下推过滤器的工作效果很差，其读取了大量没用的统计数据，这些统计数据对过滤 Parquet 的 row groups 无效（当 ETL 作业写入 Parquet 文件时列数据是无序的）。

在过去一年时间里，自己在 Spark 中添加了一系列的优化措施来提升 Parquet 的下推性能。我们开发了一个名为 LocalSort 的特性，在写 Parquet 文件时通过对一些列添加一个排序步骤，从而可以利用这些统计数据明显区分 Parquet Row groups，并提高压缩比（根据历史查询自动进行，不需要修改ETL作业)。此外，我们开发了一个名为 Prewhere 的特性。Prewhere parquet reader 从下推过滤器中选择低开销的列，以批处理方式来读取这些列的数据，并使用下推过滤器过滤数据，同时跳过其他不需要的列。这些努力的直接结果是，我们实现了平均 30% 的查询改进，40%的存储改进，而开销只有5%。

这篇文章将深入介绍 LocalSort 和 Prewhere，同时介绍 LocalSort/Prewhere 都有哪些用户场景，最后也会介绍一些基于历史查询自动建议对列进行排序的相关工作。

本次分享的主题就下面三个：

- Spark SQL 在字节跳动的使用；
- Spark 是如何读取 Parquet 文件的；
- 字节跳动是优化 Parquet Filter Pushdown 和 Parquet Reader 的

字节跳动的 Spark SQL 使用情况