



下载APP



13 | 广播变量（二）：有哪些途径让Spark SQL选择Broadcast Joins?

2021-04-12 吴磊

Spark性能调优实战

[进入课程 >](#)**讲述：吴磊**

时长 16:01 大小 14.68M



你好，我是吴磊。

上一讲我们说到，在数据关联场景中，广播变量是克制 Shuffle 的杀手锏，用 Broadcast Joins 取代 Shuffle Joins 可以大幅提升执行性能。但是，很多同学只会使用默认的广播变量，不会去调优。那么，我们该怎么保证 Spark 在运行时优先选择 Broadcast Joins 策略呢？

今天这一讲，我就围绕着数据关联场景，从配置项和开发 API 两个方面，帮你梳理出调优手段，让你能够游刃有余地运用广播变量。



利用配置项强制广播

我们先来从配置项的角度说一说，有哪些办法可以让 Spark 优先选择 Broadcast Joins。在 Spark SQL 配置项那一讲，我们提到过 `spark.sql.autoBroadcastJoinThreshold` 这个配置项。它的设置值是存储大小，默认是 10MB。它的含义是，**对于参与 Join 的两张表来说，任意一张表的尺寸小于 10MB，Spark 就在运行时采用 Broadcast Joins 的实现方式去做数据关联。**另外，AQE 在运行时尝试动态调整 Join 策略时，也是基于这个参数来判定过滤后的数据表是否足够小，从而把原本的 Shuffle Joins 调整为 Broadcast Joins。

配置项	含义
<code>spark.sql.autoBroadcastJoinThreshold</code>	数据表采用Broadcast Join实现方式的最低阈值



为了方便你理解，我来举个例子。在数据仓库中，我们经常会看到两张表：一张是订单事实表，为了方便，我们把它记成 Fact；另一张是用户维度表，记成 Dim。事实表体量很大在 100GB 量级，维度表很小在 1GB 左右。两张表的 Schema 如下所示：

[复制代码](#)

```
1 //订单事实表Schema
2 orderID: Int
3 userID: Int
4 trxDate: Timestamp
5 productID: Int
6 price: Float
7 volume: Int
8
9 //用户维度表Schema
10 userID: Int
11 name: String
12 age: Int
13 gender: String
```

当 Fact 表和 Dim 表基于 userID 做关联的时候，由于两张表的尺寸大小都远超 `spark.sql.autoBroadcastJoinThreshold` 参数的默认值 10MB，因此 Spark 不得不选择 Shuffle Joins 的实现方式。但如果我们把这个参数的值调整为 2GB，因为 Dim 表的尺寸比 2GB 小，所以，Spark 在运行时会选择把 Dim 表封装到广播变量里，并采用 Broadcast Join 的方式来完成两张表的数据关联。

显然，对于绝大多数的 Join 场景来说，`autoBroadcastJoinThreshold` 参数的默认值 10MB 太低了，因为现在企业的数据体量都在 TB，甚至 PB 级别。因此，想要有效地利用 Broadcast Joins，我们需要把参数值调大，一般来说，2GB 左右是个不错的选择。

现在我们已经知道了，**使用广播阈值配置项让 Spark 优先选择 Broadcast Joins 的关键，就是要确保至少有一张表的存储尺寸小于广播阈值。**

但是，在设置广播阈值的时候，不少同学都跟我抱怨：“我的数据量明明小于 `autoBroadcastJoinThreshold` 参数设定的广播阈值，为什么 Spark SQL 在运行时并没有选择 Broadcast Joins 呢？”

详细了解后我才知道，这些同学所说的数据量，**其实指的是数据表在磁盘上的存储大小**，比如用 `ls` 或是 `du -sh` 等系统命令查看文件得到的结果。要知道，同一份数据在内存中的存储大小往往会比磁盘中的存储大小膨胀数倍，甚至十数倍。这主要有两方面原因。

一方面，为了提升存储和访问效率，开发者一般采用 Parquet 或是 ORC 等压缩格式把数据落盘。这些高压缩比的磁盘数据展开到内存之后，数据量往往会翻上数倍。

另一方面，受限于对象管理机制，在堆内内存中，JVM 往往需要比数据原始尺寸更大的内存空间来存储对象。


我们来举个例子，字符串 “abcd” 按理说只需要消耗 4 个字节，但是，JVM 在堆内存储这 4 个字符串总共需要消耗 48 个字节！那在运行时，一份看上去不大的磁盘数据展开到内存，翻上个 4、5 倍并不稀奇。因此，如果你按照磁盘上的存储大小去配置 `autoBroadcastJoinThreshold` 广播阈值，大概率也会遇到同样的窘境。

那么问题来了，有什么办法能准确地预估一张表在内存中的存储大小呢？

首先，我们要避开一个坑。我发现，有很多资料推荐用 Spark 内置的 `SizeEstimator` 去预估分布式数据集的存储大小。结合多次实战和踩坑经验，咱们必须要指出，**`SizeEstimator` 的估算结果不准确**。因此，你可以直接跳过这种方法，这也能节省你调优的时间和精力。

我认为比较靠谱的办法是：**第一步，把要预估大小的数据表缓存到内存，比如直接在 `DataFrame` 或是 `Dataset` 上调用 `cache` 方法；第二步，读取 Spark SQL 执行计划的统**

计数据。这是因为，Spark SQL 在运行时，就是靠这些统计数据来制定和调整执行策略的。

 复制代码

```
1 val df: DataFrame = _
2 df.cache.count
3
4 val plan = df.queryExecution.logical
5 val estimated: BigInt = spark
6   .sessionState
7   .executePlan(plan)
8   .optimizedPlan
9   .stats
10  .sizeInBytes
```

你可能会说：“这种方法虽然精确，但是这么做，实际上已经是在运行时进行调优了。把数据先缓存到内存，再去计算它的存储尺寸，当然更准确了。”没错，采用这种计算方式，调优所需花费的时间和精力确实更多，但在很多情况下，尤其是 Shuffle Joins 的执行效率让你痛不欲生的时候，这样的付出是值得的。

利用 API 强制广播


既然数据量的预估这么麻烦，有没有什么办法，不需要配置广播阈值，就可以让 Spark SQL 选择 Broadcast Joins？还真有，而且办法还不止一种。

开发者可以通过 Join Hints 或是 SQL functions 中的 broadcast 函数，来强制 Spark SQL 在运行时采用 Broadcast Joins 的方式做数据关联。下面我就来分别讲一讲它们的含义和作用，以及该如何使用。必须要说明的是，这两种方式是等价的，并无优劣之分，只不过有了多样化的选择之后，你就可以根据自己的偏好和习惯来灵活地进行开发。

用 Join Hints 强制广播

Join Hints 中的 Hints 表示“提示”，它指的是在开发过程中使用特殊的语法，明确告知 Spark SQL 在运行时采用哪种 Join 策略。一旦你启用了 Join Hints，不管你的数据表是不是满足广播阈值，Spark SQL 都会尽可能地尊重你的意愿和选择，使用 Broadcast Joins 去完成数据关联。

我们来举个例子，假设有两张表，一张表的内存大小在 100GB 量级，另一张小一些，2GB 左右。在广播阈值被设置为 2GB 的情况下，并没有触发 Broadcast Joins，但我们又不想花费时间和精力去精确计算小表的内存占用到底是多大。在这种情况下，我们就可以用 Join Hints 来帮我们做优化，仅仅几句提示就可以帮我们达到目的。

 复制代码

```
1 val table1: DataFrame = spark.read.parquet(path1)
2 val table2: DataFrame = spark.read.parquet(path2)
3 table1.createOrReplaceTempView("t1")
4 table2.createOrReplaceTempView("t2")
5
6 val query: String = "select /*+ broadcast(t2) */ * from t1 inner join t2 on t1"
7 val queryResults: DataFrame = spark.sql(query)
```

你看，在上面的代码示例中，只要在 SQL 结构化查询语句里面加上一句 `/*+ broadcast(t2) */` 提示，我们就可以强制 Spark SQL 对小表 t2 进行广播，在运行时选择 Broadcast Joins 的实现方式。提示语句中的关键字，除了使用 broadcast 外，我们还可以用 broadcastjoin 或者 mapjoin，它们实现的效果都一样。

如果你不喜欢用 SQL 结构化查询语句，尤其不想频繁地在 Spark SQL 上下文中注册数据表，你也可以在 DataFrame 的 DSL 语法中使用 Join Hints。

 复制代码

```
1 table1.join(table2.hint("broadcast"), Seq("key"), "inner")
2
```


在上面的 DSL 语句中，我们只要在 table2 上调用 hint 方法，然后指定 broadcast 关键字，就可以同样达到强制广播表 2 的效果。

总之，Join Hints 让开发者可以灵活地选择运行时的 Join 策略，对于熟悉业务、了解数据的同学来说，Join Hints 允许开发者把专家经验凌驾于 Spark SQL 的优化引擎之上，更好地服务业务。

不过，Join Hints 也有个小缺陷。如果关键字拼写错误，Spark SQL 在运行时并不会显示地抛出异常，而是默默地忽略掉拼写错误的 hints，假装它压根不存在。因此，在使用 Join Hints 的时候，需要我们在编译时自行确认 Debug 和纠错。

用 broadcast 函数强制广播

如果你不想等到运行时才发现问题，想让编译器帮你检查类似的拼写错误，那么你可以使用强制广播的第二种方式：broadcast 函数。这个函数是类库 org.apache.spark.sql.functions 中的 broadcast 函数。调用方式非常简单，比 Join Hints 还要方便，只需要用 broadcast 函数封装需要广播的数据表即可，如下所示。

 复制代码

```
1 import org.apache.spark.sql.functions.broadcast
2 table1.join(broadcast(table2), Seq("key"), "inner")
```

你可能会问：“既然开发者可以通过 Join Hints 和 broadcast 函数强制 Spark SQL 选择 Broadcast Joins，那我是不是就可以不用理会广播阈值的配置项了？”其实还真不是。我认为，**以广播阈值配置为主，以强制广播为辅**，往往是不错的选择。

广播阈值的设置，更多的是把选择权交给 Spark SQL，尤其是在 AQE 的机制下，动态 Join 策略调整需要这样的设置在运行时做出选择。强制广播更多的是开发者以专家经验去指导 Spark SQL 该如何选择运行时策略。二者相辅相成，并不冲突，开发者灵活地运用就能平衡 Spark SQL 优化策略与专家经验在应用中的比例。


广播变量不是银弹

不过，虽然我们一直在强调，数据关联场景中广播变量是克制 Shuffle 的杀手锏，但广播变量并不是银弹。

就像有的同学会说：“开发者有这么多项，甚至可以强制 Spark 选择 Broadcast Joins，那我们是不是可以把所有 Join 操作都用 Broadcast Joins 来实现？”答案当然是否定的，广播变量不能解决所有的数据关联问题。

首先，从性能上来讲，Driver 在创建广播变量的过程中，需要拉取分布式数据集所有的数据分片。在这个过程中，网络开销和 Driver 内存会成为性能隐患。广播变量尺寸越大，额外引入的性能开销就会越多。更何况，如果广播变量大小超过 8GB，Spark 会直接抛异常中断任务执行。

其次，从功能上来讲，并不是所有的 Joins 类型都可以转换为 Broadcast Joins。一来，Broadcast Joins 不支持全连接（Full Outer Joins）；二来，在所有的数据关联中，我们不能广播基表。或者说，即便开发者强制广播基表，也无济于事。比如说，在左连接（Left Outer Join）中，我们只能广播右表；在右连接（Right Outer Join）中，我们只能广播左表。在下面的代码中，即便我们强制用 broadcast 函数进行广播，Spark SQL 在运行时还是会选择 Shuffle Joins。

 复制代码

```
1 import org.apache.spark.sql.functions.broadcast
2 broadcast (table1).join(table2, Seq("key"), "left")
3 table1.join(broadcast(table2), Seq("key"), "right")
```

小结

这一讲，我们总结了 2 种方法，让 Spark SQL 在运行时能够选择 Broadcast Joins 策略，分别是设置配置项和用 API 强制广播。

首先，设置配置项主要是设置 autoBroadcastJoinThreshold 配置项。开发者通过这个配置项指示 Spark SQL 优化器。只要参与 Join 的两张表中，有一张表的尺寸小于这个参数值，就在运行时采用 Broadcast Joins 的实现方式。

为了让 Spark SQL 采用 Broadcast Joins，开发者要做的，就是让数据表在内存中的尺寸小于 autoBroadcastJoinThreshold 参数的设定值。

除此之外，在设置广播阈值的时候，因为磁盘数据展开到内存的时候，存储大小会成倍增加，往往导致 Spark SQL 无法采用 Broadcast Joins 的策略。因此，我们在做数据关联的时候，还要先预估一张表在内存中的存储大小。一种精确的预估方法是先把 DataFrame 缓存，然后读取执行计划的统计数据。

其次，用 API 强制广播有两种方法，分别是设置 Join Hints 和用 broadcast 函数。设置 Join Hints 的方法就是在 SQL 结构化查询语句里面加上一句 `/*+ broadcast(某表) */` 的提示就可以了，这里的 broadcast 关键字也可以换成 broadcastjoin 或者 mapjoin。另外，你也可以在 DataFrame 的 DSL 语法中使用调用 hint 方法，指定 broadcast 关键字，来达到同样的效果。设置 broadcast 函数的方法非常简单，只要用 broadcast 函数封装需要广播的数据表就可以了。

总的来说，不管是设置配置项还是用 API 强制广播都有各自的优缺点，所以，**以广播阈值配置为主、强制广播为辅**，往往是一个不错的选择。

最后，不过，我们也要注意，广播变量不是银弹，它并不能解决所有的数据关联问题，所以在日常的开发工作中，你要注意避免滥用广播。

每日一练

1. 除了 broadcast 关键字外，在 Spark 3.0 版本中，Join Hints 还支持哪些关联类型和关键字？
2. DataFrame 可以用 sparkContext.broadcast 函数来广播吗？它和 org.apache.spark.sql.functions.broadcast 函数之间的区别是什么？

期待在留言区看到你的思考和答案，我们下一讲见！

提建议

更多课程推荐

Kafka 核心技术与实战

全面提升你的 Kafka 实战能力

胡夕

Apache Kafka Committer
老虎证券技术总监

涨价倒计时 🕒

今日订阅 **¥89**，4月29日涨价至 **¥199**

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 12 | 广播变量（一）：克制Shuffle，如何一招制胜！

下一篇 14 | CPU视角：如何高效地利用CPU？

精选留言 (5)

写留言



kingcall

2021-04-12

第一题：可以参考JoinStrategyHint.scala
BROADCAST,
SHUFFLE_MERGE,
SHUFFLE_HASH,
SHUFFLE_REPLICATE_NL...

展开 ∨

作者回复: Perfect x 2! 两道题都是满分 🌹~

不过，第一题，我再追问一句，当然，这么追问有点过分，哈哈，毕竟咱们这节课还没有讲不同Join的实现方式（26讲会展开）。追问的问题是：

SHUFFLE_MERGE,
SHUFFLE_HASH,
SHUFFLE_REPLICATE_NL

这几个hints的作用和效果，分别是什么？他们分别适合哪些场景？不妨提前思考一下，等到了26讲，咱们再细聊哈~

1

9



YJ

2021-04-16

老师，我有一个问题。

```
bigTableA.Join(broadcast(smallTable), ...);
```

```
bigTableB.Join(broadcast(smallTable), ...);
```

```
bigTableA.Join(bigTableB, ...);
```

这里 广播了的smallTable 会被第二个join重用吗？ 还是说会被广播两次？

展开

作者回复: 好问题， 这种写法， Spark不会复用先前的广播变量，所以第二次的Broadcast会重复计算。

复用广播最保险的方式，是这种写法：

```
val bcSmallTable = sparkContext.broadcast(smallTable)
```

```
bigTableA.Join(bcSmallTable.value, ...);
```

```
bigTableB.Join(bcSmallTable.value, ...);
```

2

3



耳东

2021-04-20

在左连接（Left Outer Join）中，我们只能广播右表；在右连接（Right Outer Join）中，我们只能广播左表。这段的意思是指在 left outer join时 大表放左边，小表放右边吗？为什么？

展开

作者回复: 先说join形式，left join，一定是大表放左边，小表放右边。right join，反过来，一定是小表放左边，大表放右边~ 这个是关联形式决定的，不然干嘛叫“左”、“右”连接呢？

再说Broadcast Joins，参与关联的两张表，要广播的话，一定是广播小表，原因虽然我们本讲没有直接说，不过结合上一讲Broadcast Join和Shuffle Joins的计算过程和原理，其实可以自行推

导出来哈~

不妨反向思维，如果你强行广播大表，会发生什么？广播大表的收益是什么？广播大表的收益，是否大于原Shuffle Joins的执行性能？不妨想一想哈~



Jefitar
2021-04-19

老师，有个问题，字符串“abcd”只需要消耗 4 个字节，为什么JVM 在堆内存存储这 4 个字符串总共需要消耗 48 个字节？

展开

作者回复: 具体细节可以参考这里哈：<https://databricks.com/blog/2015/04/28/project-tungsten-bringing-spark-closer-to-bare-metal.html>

java.lang.String object internals:
OFFSET SIZE TYPE DESCRIPTION VALUE
0 4 (object header) ...
4 4 (object header) ...
8 4 (object header) ...
12 4 char[] String.value []
16 4 int String.hash 0
20 4 int String.hash32 0
Instance size: 24 bytes (reported by Instrumentation API)



辰
2021-04-16

Hint好像是spark2.4版本之后才会有的吧，我公司版本是2.2，但是我之前使用broadcast不管用，然后试了一下hint,居然生效了

展开

作者回复: 3.0之前只支持Broadcast hint，3.0之后支持的就多了：
BROADCAST
MERGE
SHUFFLE_HASH
SHUFFLE_REPLICATE_NL

