dogggggggggggggggggggggggggggggggggie的专栏

关注大数据处理技术、数据挖掘、算法、编程语言相关以及好看的电影、

三 目录视图

₩ 摘要视图

RSS 订阅





译文: 1篇

博客专栏



大数据畅谈 文章: 10篇 阅读: 32604

评论: 20条

文章搜索

文章分类 论文 (2) 其他 (1) Scala (8) Spark (9) 资料存档 (1) ZooKeeper (1)

文章存档
2017年04月 (2)
2017年03月 (1)
2016年12月 (2)
2016年11月 (1)
2014年07月 (1)



SparkSQL的3种Join实现

标签: 大数据 spark sparksql join 分布式

2016-12-12 23:06 5792人阅读 评论 "

≦ 分类: Spark (8) ▼

■版权声明:本文为博主原创文章,转载请联系微信doggie_wang。

目录(?) [+]

引言

Join是SQL语句中的常用操作,良好的表结构能够将数据分散在不同的表中,使其符合某种范式,减少表冗余、更新容错等。而建立表和表之间关系的最佳方式就是Join操作。

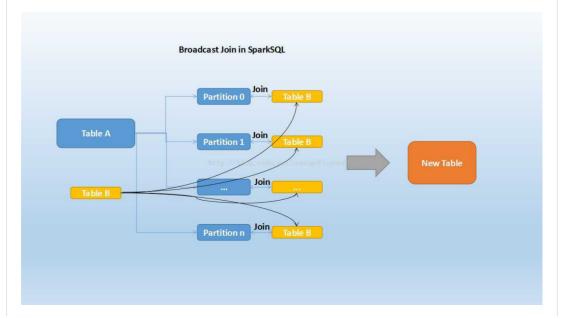
SparkSQL作为大数据领域的SQL实现,自然也对Join操作做了不少优化,今天主要看一下在SparkSQL中对于Join,常见的3种实现。

SparkSQL的3种Join实现 Broadcast Join

大家知道,在数据库的常见模型中(比如星型模型或者雪花模型),表一般分为两种:事实表和维度表。维度表一般指固定的、变动较少的表,例如联系人、物品种类等,一般数据有限。而事实表一般记录流水,比如销售清单等,通常随着时间的增长不断膨胀。

因为Join操作是对两个表中key值相同的记录进行连接,在SparkSQL中,对两个表做Join最直接的方式是先根据key 分区,再在每个分区中把key值相同的记录拿出来做连接操作。但这样就不可避免地涉及到shuffle,而shuffle在 Spark中是比较耗时的操作,我们应该尽可能的设计Spark应用使其避免大量的shuffle。

当维度表和事实表进行Join操作时,为了避免shuffle,我们可以将大小有限的维度表的全部数据分发到每个节点上,供事实表使用。executor存储维度表的全部数据,一定程度上牺牲了空间,换取shuffle操作大量的耗时,这在SparkSQL中称作Broadcast Join,如下图所示:



阅读排行

Scala从零开始: 使用Inte

(53193) Scala从零开始: 使用Sca

Scala M 令 开始 · 使用 Sca

(27138) Spark大师之路:广播变! (7522)

Scala从零开始: 函数参数 (7495)

Spark大师之路: 使用ma (7101)

SparkSQL的3种Join实现 (5784)

Spark大师之路: Spark的 (5455)

Efficient Graph-Based In (4911)

Coursera公开课Function (3876)

选择Scala的理由? (3014)

评论排行

Coursera公开课Function	(4)
Spark大师之路: 使用ma	(4)
Scala从零开始: 函数参数	(3)
Scala从零开始: 使用Sca	(3)
Efficient Graph-Based In	(2)
Spark 1.0.0版本发布	(2)
Scala从零开始: 使用Inte	(2)
SparkSQL的3种Join实现	(1)
在CSDN博客中用latex写	(0)

推荐文章

- * CSDN日报20170828——《4 个方法快速打造你的阅读清单》
- * Android检查更新下载安装

Curator在大数据集群可靠

(0)

- * 动手打造史上最简单的 Recycleview 侧滑菜单
- * TCP网络通讯如何解决分包粘 包问题
- * SDCC 2017之区块链技术实战 线上峰会
- * 快速集成一个视频直播功能

最新评论

SparkSQL的3种Join实现 钱曙光: 已支付宝。

Scala从零开始: 使用Intellij IDE/bimeiqiao: 您好,我在create project时,配置Scala Home 为D:\Program Fil...

Scala从零开始: 使用Intellij IDE/ Nouma: 谢谢

Coursera公开课Functional Progi 狗叔: @qq_24719251:链接: http://pan.baidu.com/s/1dDyHXQh 密码...

Scala从零开始:函数参数的传名 狗叔:@sinat_29290997:如果走truc分支的概率较大,则使用传名方式可以避免很多不必要的提前...

Scala从零开始:使用Scala IDE¹ 狗叔:@u012598327:和java一样的,只要添加了scala的类库,它就可以把scala源码编译成...

Scala从零开始: 函数参数的传名 sinat_29290997: 最后的例子没看明白,用不用传名都一样好吧。

Scala从零开始: 使用Scala IDE

Table B是较小的表,黑色表示将其广播到每个executor节点上,Table A的每个partition会通过block manager取到 Table A的数据。根据每条记录的Join Key取到Table B中相对应的记录,根据Join Type进行操作。这个过程比较简单,不做赘述。

Broadcast Join的条件有以下几个:

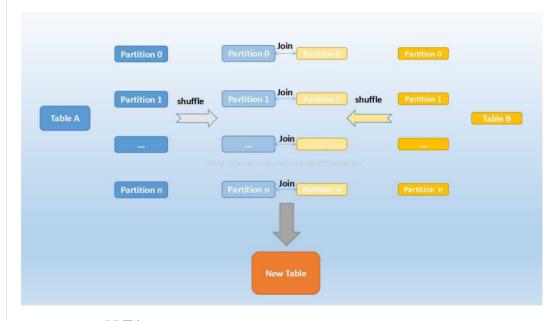
- 1. 被广播的表需要小于spark.sql.autoBroadcastJoinThreshold所配置的值,默认是10M (或者加了hint)
- 2. 基表不能被广播,比如left outer join时,只能广播右表

看起来广播是一个比较理想的方案,但它有没有缺点呢?也很明显。这个方案只能用于广播较小的 冗余传输就远大于shuffle的开销;另外,广播时需要将被广播的表现collect到driver端,当频繁有广挂 driver的内存也是一个考验。

Shuffle Hash Join

当一侧的表比较小时,我们选择将其广播出去以避免shuffle,提高性能。但因为被广播的表首先被collect到driver 段,然后被冗余分发到每个executor上,所以当表比较大时,采用broadcast join会对driver端和executor端造成较大的压力。

但由于Spark是一个分布式的计算引擎,可以通过分区的形式将大批量的数据划分成n份较小的数据集进行并行计算。这种思想应用到Join上便是Shuffle Hash Join了。利用key相同必然分区相同的这个原理,SparkSQL将较大表的join分而治之,先将表划分成n个分区,再对两个表中相对应分区的数据分别进行Hash Join,这样即在一定程度上减少了driver广播一侧表的压力,也减少了executor端取整张被广播表的内存消耗。其原理如下图:



Shuffle Hash Join分为两步:

- 1. 对两张表分别按照join keys进行重分区,即shuffle,目的是为了让有相同join keys值的记录分到对应的分区中
- 2. 对对应分区中的数据进行join,此处先将小表分区构造为一张hash表,然后根据大表分区中记录的join keys值拿出来进行匹配

Shuffle Hash Join的条件有以下几个:

- 1. 分区的平均大小不超过spark.sql.autoBroadcastJoinThreshold所配置的值,默认是10M
- 2. 基表不能被广播, 比如left outer join时, 只能广播右表
- 3. 一侧的表要明显小于另外一侧,小的一侧将被广播(明显小于的定义为3倍小,此处为经验值)

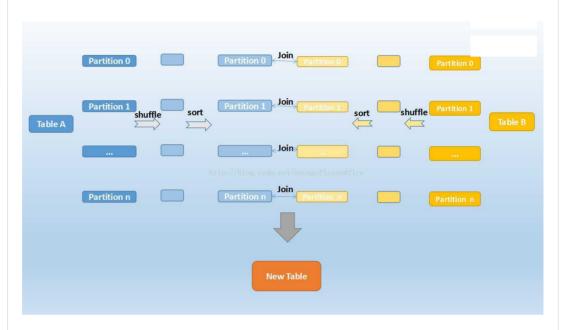


我们可以看到,在一定大小的表中,SparkSQL从时空结合的角度来看,将两个表进行重新分区,并且对小表中的分区进行hash化,从而完成join。在保持一定复杂度的基础上,尽量减少driver和executor的内存压力,提升了计算时的稳定性。

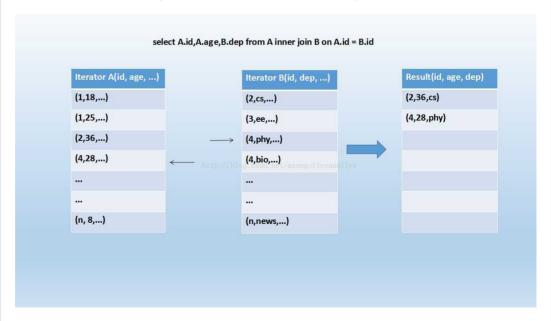
Sort Merge Join

上面介绍的两种实现对于一定大小的表比较适用,但当两个表都非常大时,显然无论适用哪种都会对计算内存造成很大压力。这是因为join时两者采取的都是hash join,是将一侧的数据完全加载到内存中,使用hash code取ioin keys值相等的记录进行连接。

当两个表都非常大时,SparkSQL采用了一种全新的方案来对表进行Join,即Sort Merge Join。这和一侧数据全部加载后再进星hash join,但需要在join前将数据排序,如下图所示:



可以看到,首先将两张表按照join keys进行了重新shuffle,保证join keys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,如下图示:

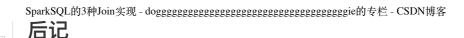


看着很眼熟吧?也很简单,因为两个序列都是有序的,从头遍历,碰到key相同的就输出;如果不同,左边小就继续取左边,反之取右边。

可以看出,无论分区有多大,Sort Merge Join都不用把某一侧的数据全部加载到内存中,而是即用即取即丢,从而大大提升了大数据量下sql join的稳定性。

石梅山庄怎么样

京都民宿



本文介绍了SparkSQL中的3中Join实现,其实这也不是什么新鲜玩意儿。传统DB也有这也的玩法儿,SparkSQL只是将其做成分布式的实现。

本文仅仅从大的理论方面介绍了这几种实现,具体到每个join type是怎么遍历、没有join keys时应该怎么做、这些实现对join keys有什么具体的需求,这些细节都没有展现出来。感兴趣的话,可以去翻翻源码。

声明: 本文为原创,版权归本人所有,禁止用于任何商业目的,转载请注明出处: http://blog.csdn.net/asongoficeandfire/article/details/53574034

如果你觉得我文章写的不错,欢迎扫支付宝二维码打赏我:)



顶 踩

上一篇 我是怎么在Spark中踩到Jetty的坑的

下一篇 SparkSQL中的Sort实现(一)

相关文章推荐

- SparkSQL JOIN 相关的自己看的笔记;
- 大数据技术实战线上峰会--董西成
- spark sql 之join等函数用法
- 30天系统掌握机器学习--唐宇迪
- Spark map-side-join 关联优化
- ORACLE数据库学习资料集锦
- 友元类
- PHP从零开始实战篇

- Spark Streaming:缓存与持久化机制
- 玩转微信小程序第一篇
- Spark-Sql创建多数据源Join实例——涉及关系库...
- 深度学习案例分享—人脸检测
- SparkSQL的3种Join实现
- SparkSQL的3种Join实现
- spark-spark-SparkSQL的3种Join实现(转)
- scala实战之SparkSQL应用实例(单表count和gro...

loft公寓出租 二级人力资源管理师 石梅山庄怎么样 石梅山庄 网上花店 平面设计周末班 二手车估价计算器 二手宝马x5价格 spark hadoop 整牙年龄 cluster 大兴二手房出售 运动服专卖店 礼品册 宝马7系二手车 hadoop spark 宝马x5 价产 杳看评论

1楼 钱曙光 2017-03-27 17:57发表



已支付宝。