复习课（八）| Resilient Distributed Datasets

2021-12-13 黄金

《大数据经典论文解读》 课程介绍 >

讲述：王惠

时长 06:53 大小 6.31M

你好，我是黄金。今天我们要来一起回顾复习的论文内容，是开源系统 Spark 的这篇引入了 RDD 概念的论文。

RDD 介绍

RDD 的全称是弹性分布式数据集，它允许开发人员在大规模集群上，以容错的方式执行内存计算。而 Spark 就是实现了 RDD 的分布式计算框架。

在 Spark 出现之前，并没有通用的分布式计算框架，可以高效地运行迭代算法。MapReduce 是通用的分布式计算框架，但不管是 Mapper 任务还是 Reducer 任务，它们的执行结果都需要写入硬盘。这样一来，由多个 MapReduce 组合而成的迭代算法程序，运行起来就不够高效。因此，**如何有效地利用分布式内存**，就成为了研究的重点。

而在 MapReduce 之后，也出现了一些可以利用分布式内存的计算框架，它们把运算的中间结果保存在内存当中。这些计算框架确实提升了执行效率，但是不够通用，只能服务于特定的算法。

直到 Spark 的出现，才有了既高效又通用的分布式内存计算框架。

容错的分布式内存数据集

在设计 RDD 的时候，主要的挑战就是**如何定义编程接口**，才能让 RDD 具备有效的故障恢复能力。

我们先来看看 MapReduce 的结果集是如何容错的。Mapper 任务把执行结果写入本地文件，服务器即使宕机，重启后依然可以读取结果。对于不能恢复的服务器，只需要把它负责的任务交给其他服务器，重新执行一遍即可。而 Reducer 任务是把执行结果写入 HDFS，由 HDFS 提供容错支持。

那么，RDD 保存在内存中，又要如何避免服务器宕机带来的影响呢？

它的方式是利用上游的 RDD，重新执行一遍任务，来生成丢失的数据。RDD 以分布式的方式存在，也就是由多个数据分片构成，一个数据分片损坏了，只需要重新生成这一个分片的数据就好。所以 RDD 在编程接口上支持的更新操作，是**粗粒度**的操作。

所谓的粗粒度，是指 RDD 需要在整个数据集上执行完更新才可见，更新到一半的数据集不可见。RDD 是只读数据集，更新操作会让一个 RDD 变成另一个 RDD，不存在中间状态，这就让恢复数据变得容易了。

RDD 定义和 Spark 编程接口

我们再来看看 RDD 的定义，论文中讲 RDD 是只读的、已分区的记录集合，RDD 只能通过明确的操作，以及通过两种数据创建：稳定存储系统中的数据；其他 RDD。 这个明确的操作，是指 map、filter 和 join 这样的操作。

相信你现在应该就能明白，为什么 RDD 需要定义成只读的、经过明确操作来创建，其实这都是**为了支持有效容错**。

Spark 为 RDD 提供的编程接口有两种，分别是转换操作 Transformation 和行动操作 Action。转换操作就是 map、filter 和 join 之类的操作，行动操作就是 count、reduce 和 collect 之类的操作。

Spark 使用惰性求值，直到调用行动操作，它才开始执行之前定义的一系列转换操作和当前这个行动操作。这些操作共同构成了一个**拓扑图**，在论文中被称为 Lineage graph。

窄依赖和宽依赖

那么，在这个拓扑图中，我们可以把 RDD 之间的依赖关系分成两种，一种是窄依赖，一种是宽依赖：

窄依赖是指，上游 RDD 的一个分区，只会影响到下游 RDD 的一个分区；

宽依赖是指，上游 RDD 的一个分区，会影响到下游 RDD 的多个分区。
对于 map 这样的转换操作，会让上下游的 RDD 形成窄依赖，对于 groupByKey 这样的转换操作，会让上下游的 RDD 形成宽依赖。

而区分这两种依赖到底有什么作用呢？

对于窄依赖而言，我们可以在 RDD 的数据分片所在的节点上，执行转换操作，让计算靠近存储，避免数据在网络上传输。而如果多个连续的转换操作构成窄依赖，那它们都可以在同一个节点，并且是数据分片所在的节点上执行。另外，窄依赖的下游 RDD 如果损坏了，只需要根据上游 RDD 重新计算，来恢复丢失的分区，由于它依赖的节点少，所以恢复的速度会很快。

对于宽依赖而言，我们需要从多个上游 RDD 获取数据。下游 RDD 损坏时，也可以通过多个上游 RDD 恢复。不过恢复宽依赖的下游 RDD，消耗的网络带宽和计算资源，会比窄依赖的下游 RDD 大得多。这个时候，我们其实可以持久化下游 RDD，当出现故障时就从磁盘中恢复，来加快恢复的速度。

性能表现

在论文的第 6 节 Evaluation，作者还对比了逻辑回归、K-means 两种迭代机器学习应用，在三种系统上运行的性能表现，这三种系统分别是：Hadoop、HadoopBinMem、Spark，其中 HadoopBinMem 是使用内存存储数据的 Hadoop。

令人惊讶的是，Spark 比使用内存存储数据的 HadoopBinMem 还要快 20 倍。这是什么原因呢？

首先是 Hadoop 软件栈的最小开销，开始时的初始化工作，结束后的清理工作，都要花不少时间；其次是处理数据时 Hadoop 的开销，处理每个 Block 都需要执行多次内存拷贝，计算校验和；最后是二进制和 Java 对象之间的转换开销，HadoopBinMem 的内存数据使用二进制表示，Spark 的内存数据直接用 Java 对象表示，这就省掉了转换的开销。

小结

最后，让我们回到 RDD 的全称，弹性分布式数据集，来理解下“弹性”的含义。徐老师说这个弹性体现在两个方面：

第一是**数据存储**上。数据不再是存放在硬盘上的，而是可以缓存在内存中。只有当内存不足的时候，才会把它们换出到硬盘上。

第二个是**选择把什么数据输出到硬盘**上。对于经过大量网络传输和计算得到的 RDD，需要保存在硬盘上。

我认为除了这两点，弹性还体现在 **RDD 的故障恢复**上，它能根据上游节点重新计算出结果。那么你有没有其他的思考或感悟呢？欢迎留言，咱们一起交流讨论。

上一篇 复习课（七）| Dremel

下一篇 25 | 从S4到Storm（一）：当分布式遇上实时计算

精选留言

写留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。