13-弹性分布式数据集：Spark大厦的地基（上）

你好，我是蔡元楠。

今天我要与你分享的主题是“弹性分布式数据集”。

上一讲中提到，Spark最基本的数据抽象是弹性分布式数据集（Resilient Distributed Dataset, 下文用RDD代指）。

Spark基于RDD定义了很多数据操作，从而使得数据处理的代码十分简洁、高效。所以，要想深入学习Spark，我们必须首先理解RDD的设计思想和特性。

为什么需要新的数据抽象模型？

传统的MapReduce框架之所以运行速度缓慢，很重要的原因就是有向无环图的中间计算结果需要写入硬盘这样的稳定存储介质中来防止运行结果丢失。

而每次调用中间计算结果都需要要进行一次硬盘的读取，反复对硬盘进行读写操作以及潜在的数据复制和序列化操作大大提高了计算的延迟。

因此，很多研究人员试图提出一个新的分布式存储方案，不仅保持之前系统的稳定性、错误恢复和可扩展性，还要尽可能地减少硬盘I/O操作。

一个可行的设想就是在分布式内存中，存储中间计算的结果，因为对内存的读写操作速度远快于硬盘。而RDD就是一个基于分布式内存的数据抽象，它不仅支持基于工作集的应用，同时具有数据流模型的特点。

RDD的定义

弹性分布式数据集是英文直译的名字，乍一看这个名字相信你会不知所云。如果你去Google或者百度搜索它的定义，你会得到如下结果：

RDD表示已被分区、不可变的，并能够被并行操作的数据集合。

这个定义很不直观，我认识的很多Spark初学者在查阅了很多资料后还是对RDD一头雾水，很难理解这个抽象的概念。接下来，让我们一起来对这个晦涩的概念抽丝剥茧，见其真义。

在上述定义以及RDD的中文译名中，我们不难发现，RDD有以下基本特性：分区、不可变和并行操作。接下来让我分别讲解这些特点。

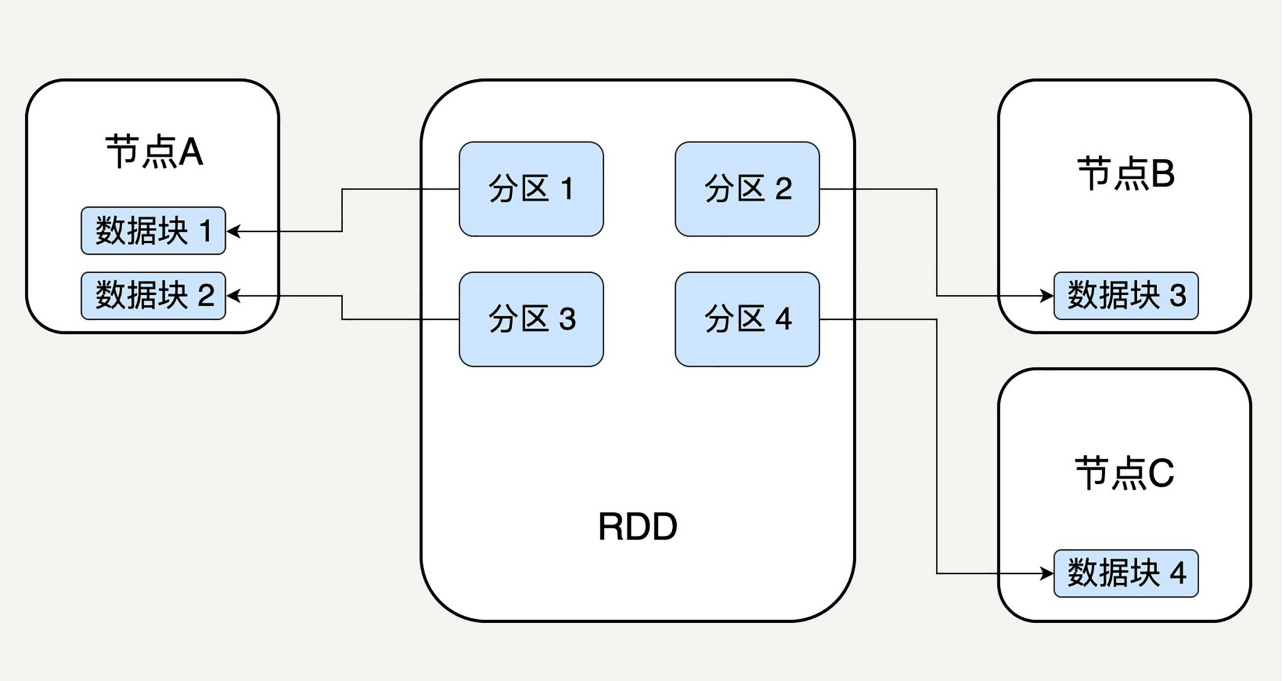
分区

顾名思义，分区代表同一个RDD包含的数据被存储在系统的不同节点中，这也是它可以被并行处理的前提。

逻辑上，我们可以认为RDD是一个大的数组。数组中的每个元素代表一个分区（Partition）。

在物理存储中，每个分区指向一个存放在内存或者硬盘中的数据块（Block），而这些数据块是独立的，它们可以被存放在系统中的不同节点。

所以，RDD只是抽象意义的数据集合，分区内部并不会存储具体的数据。下图很好地展示了RDD的分区逻辑结构：



RDD中的每个分区存有它在该RDD中的index。通过RDD的ID和分区的index可以唯一确定对应数据块的编号，从而通过底层存储层的接口中提取到数据进行处理。

在集群中，各个节点上的数据块会尽可能地存放在内存中，只有当内存没有空间时才会存入硬盘。这样可以最大化地减少硬盘读写的开销。

虽然 RDD 内部存储的数据是只读的，但是，我们可以去修改（例如通过 repartition 转换操作）并行计算单元的划分结构，也就是分区的数量。

不可变性

不可变性代表每一个RDD都是只读的，它所包含的分区信息不可以被改变。既然已有的RDD不可以被改变，我们只可以对现有的RDD进行**转换**（Transformation）操作，得到新的RDD作为中间计算的结果。从某种程度上讲，RDD与函数式编程的Collection很相似。

```
lines = sc.textFile("data.txt")
lineLengths = lines.map(lambda s: len(s))
totalLength = lineLengths.reduce(lambda a, b: a + b)
```

在上述的简单例子中，我们首先读入文本文件data.txt，创建了第一个RDD lines，它的每一个元素是一行文本。然后调用map函数去映射产生第二个RDD lineLengths，每个元素代表每一行简单文本的字数。最后调用reduce函数去得到第三个RDD totalLength，它只有一个元素，代表整个文本的总字数。

那么这样会带来什么好处呢？显然，对于代表中间结果的RDD，我们需要记录它是通过哪个RDD进行哪些转换操作得来，即**依赖关系**，而不用立刻去具体存储计算出的数据本身。

这样做有助于提升Spark的计算效率，并且使错误恢复更加容易。

试想，在一个有N步的计算模型中，如果记载第N步输出RDD的节点发生故障，数据丢失，我们可以从第N-1步的RDD出发，再次计算，而无需重复整个N步计算过程。这样的容错特性也是RDD为什么是一个“弹性”的数据集的原因之一。后边我们会提到RDD如何存储这样的依赖关系。

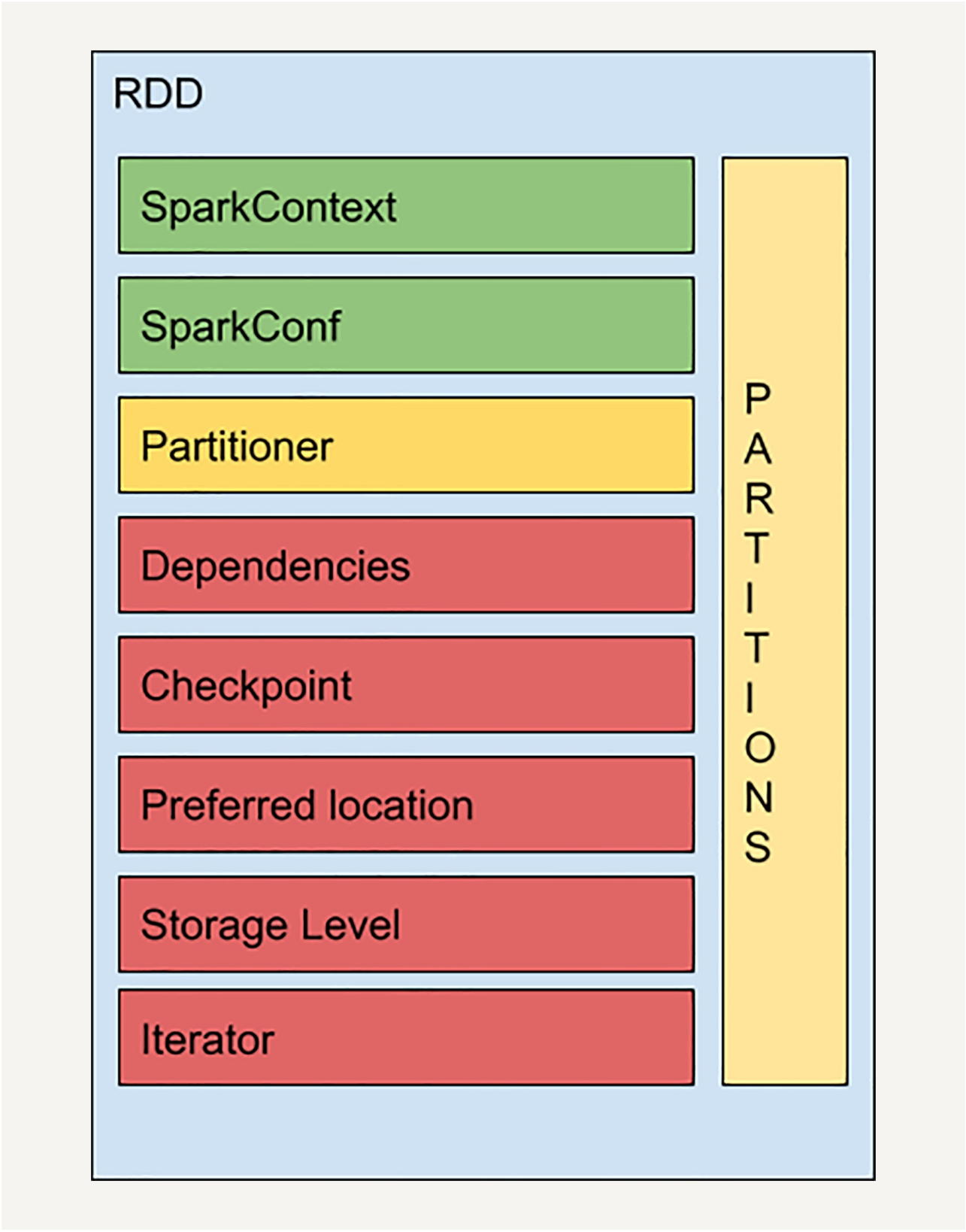
并行操作

由于单个RDD的分区特性，使得它天然支持并行操作，即不同节点上的数据可以被分别处理，然后产生一个新的RDD。

RDD的结构

通过上述讲解，我们了解了RDD的基本特性——分区、不可变和并行计算。而且，我们还提到每一个RDD里都会包括分区信息、所依赖的父RDD以及通过怎样的转换操作才能由父RDD得来等信息。

实际上RDD的结构远比你想象的要复杂，让我们来看一个RDD的简易结构示意图：



SparkContext是所有Spark功能的入口，它代表了与Spark节点的连接，可以用来创建RDD对象以及在节点中的广播变量等。一个线程只有一个SparkContext。SparkConf则是一些参数配置信息。感兴趣的同学可以去阅读官方的技术文档，一些相对不重要的概念我就不再赘述了。

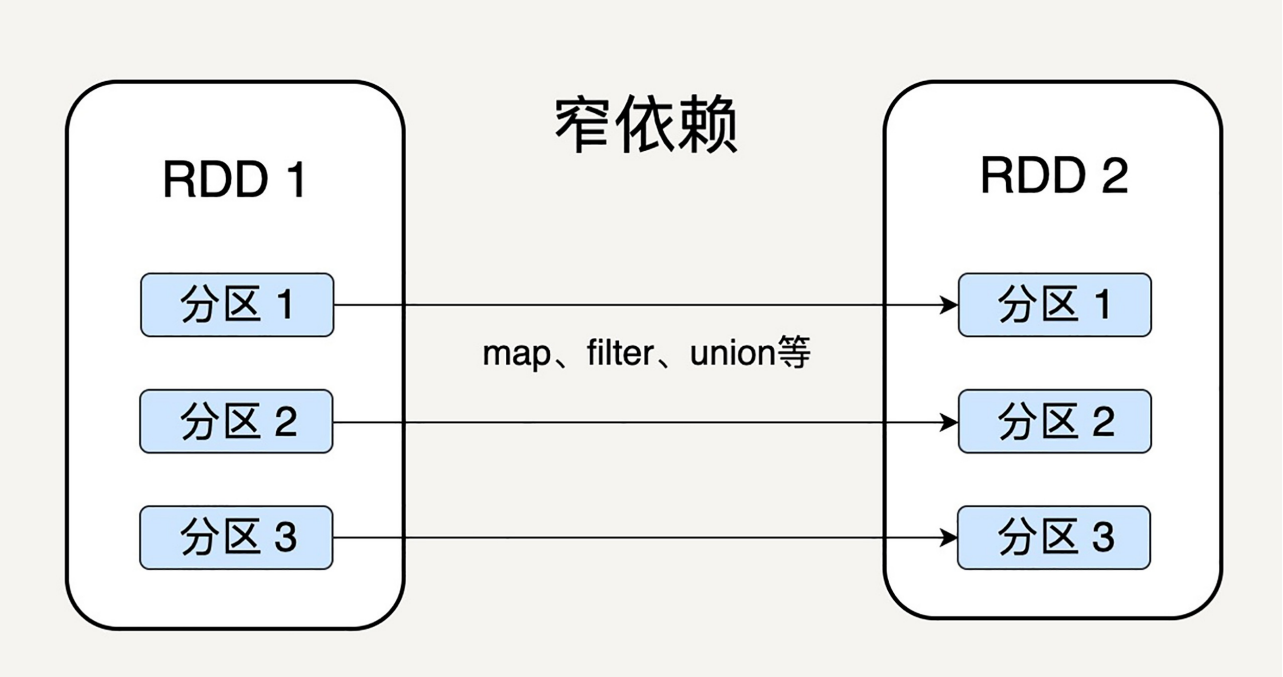
Partitions前文中我已经提到过，它代表RDD中数据的逻辑结构，每个Partition会映射到某个节点内存或硬盘的一个数据块。

Partitioner决定了RDD的分区方式，目前有两种主流的分区方式：Hash partitioner和Range partitioner。Hash，顾名思义就是对数据的Key进行散列分区，Range则是按照Key的排序进行均匀分区。此外我们还可以创建自定义的Partitioner。

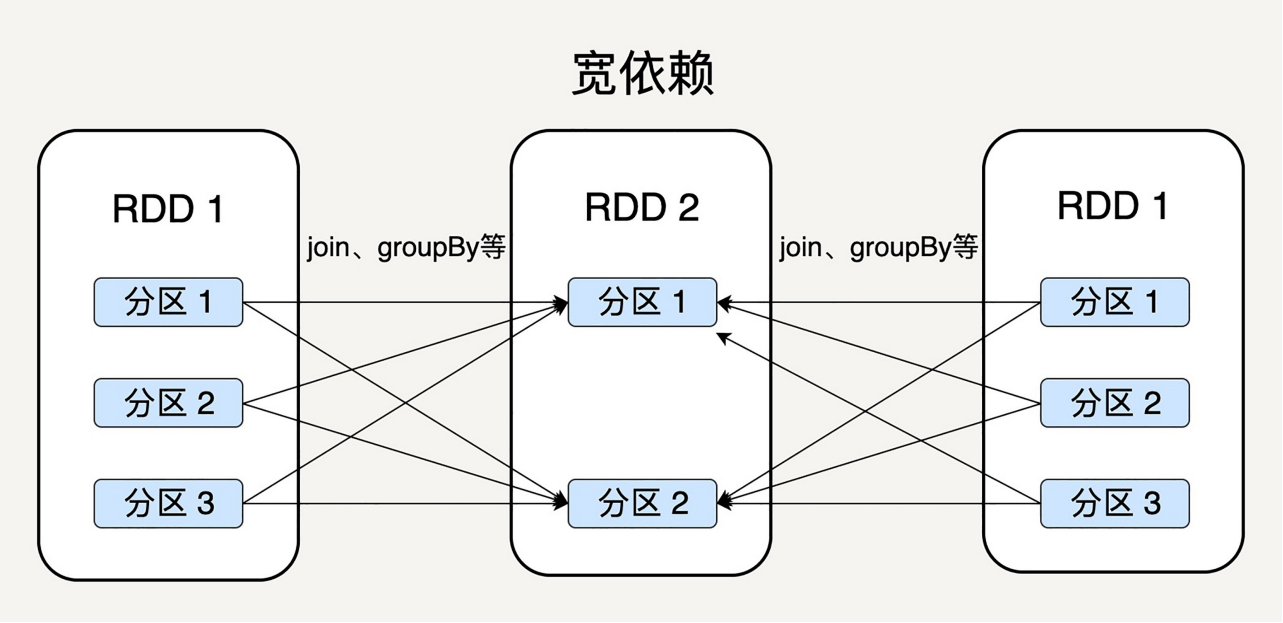
依赖关系

Dependencies是RDD中最重要的组件之一。如前文所说，Spark不需要将每个中间计算结果进行数据复制以防数据丢失，因为每一步产生的RDD里都会存储它的依赖关系，即它是通过哪个RDD经过哪个转换操作得到的。

细心的读者会问这样一个问题，父RDD的分区和子RDD的分区之间是否是一对一的对应关系呢？Spark支持两种依赖关系：窄依赖（Narrow Dependency）和宽依赖（Wide Dependency）。



窄依赖就是父RDD的分区可以一一对应到子RDD的分区，宽依赖就是父RDD的每个分区可以被多个子RDD的分区使用。



显然，窄依赖允许子RDD的每个分区可以被并行处理产生，而宽依赖则必须等父RDD的所有分区都被计算好之后才能开始处理。

如上图所示，一些转换操作如map、filter会产生窄依赖关系，而Join、groupBy则会生成宽依赖关系。

这很容易理解，因为map是将分区里的每一个元素通过计算转化为另一个元素，一个分区里的数据不会跑到两个不同的分区。而groupBy则要将拥有所有分区里有相同Key的元素放到同一个目标分区，而每一个父分区都可能包含各种Key的元素，所以它可能被任意一个子分区所依赖。

Spark之所以要区分宽依赖和窄依赖是出于以下两点考虑：

- 窄依赖可以支持在同一个节点上链式执行多条命令，例如在执行了 map 后，紧接着执行 filter。相反，宽依赖需要所有的父分区都是可用的，可能还需要调用类似 MapReduce 之类的操作进行跨节点传递。
- 从失败恢复的角度考虑，窄依赖的失败恢复更有效，因为它只需要重新计算丢失的父分区即可，而宽依赖牵涉到 RDD 各级的多个父分区。

小结

弹性分布式数据集作为Spark的基本数据抽象，相较于Hadoop/MapReduce的数据模型而言，各方面都有很大的提升。

首先，它的数据可以尽可能地存在内存中，从而大大提高的数据处理的效率；其次它是分区存储，所以天然支持并行处理；而且它还存储了每一步骤计算结果之间的依赖关系，从而大大提升了数据容错性和错误恢复的正确率，使Spark更加可靠。

下一讲，我们会继续深入研究RDD的容错机制、任务执行机制，以及Spark定义在RDD上的各种转换与动作操作。

思考题

窄依赖是指父RDD的每一个分区都可以唯一对应子RDD中的分区，那么是否意味着子RDD中的一个分区只可以对应父RDD中的一个分区呢？如果子RDD的一个分区需要由父RDD中若干个分区计算得来，是否还算窄依赖？

最后，欢迎你把对弹性分布式数据集的疑问写在留言区，与我和其他同学一起讨论。

如果你觉得有所收获，也欢迎把文章分享给你的朋友。

新版升级：点击「请朋友读」，20位好友免费读，邀请订阅更有现金奖励。

精选留言：

- Milittle 2019-05-15 09:10:22
  对于思考题：需要重点理解原文中这句话：
  窄依赖就是父 RDD 的分区可以一一对应到子 RDD 的分区，宽依赖就是父 RDD 的每个分区可以被多个子 RDD 的分区使用。
  这句话说明了，窄依赖的父RDD必须有一个对应的子RDD，也就是说父RDD的一个分区只能被子RDD一个分区使用，但是反过来子RDD的一个分区可以使用父RDD的多个分区。那就回复今天的思考题，第一个疑问窄依赖子RDD的分区不一定只对应父RDD的一个分区，只要满足被子RDD分区利用的父RDD分区不被子RDD的其他分区利用就算窄依赖。第二个疑问其实上面已经做了回答，只有当子RDD分区依赖的父RDD分区不被其他子RDD分区依赖，这样的计算就是窄依赖，否则是宽依赖。
  最后，总结以下，就是只有父RDD的分区被多个子RDD的分区利用的时候才是宽依赖，其他的情况就是窄依赖。如果有哪里理解不对的地方，请老师指正，谢谢~ [5赞]

- 一 2019-05-15 21:34:06
  老师好！能不能请老师讲讲Spark和Flink的对比呢？这二者谁在机器学习乃至深度学习中更有优势呢？ [1赞]

- 追梦 2019-05-16 19:42:19
  老师，下面这句话对吗？
  一个stage的所有task都执行完毕之后，会在各个节点本地的磁盘文件中写入计算中间结果，然后Driver就会调度运行下一个stage。下一个stage的task的输入数据就是上一个stage输出的中间结果。

  说明：spark的中间计算结果会直接落到磁盘上的？？？

- hua168 2019-05-16 17:56:56
  老师，hadoop计算框架Map/Reduce 过时了，那另一个存储框架HDFS，也过时了吗？
  现在我看很多云提供商都是对象存储实现海量需求，
  现在开源的分布式存储，能在生产环境使用的，用什么了,ceph?

- 吟游雪人 2019-05-16 10:49:58
  新的RDD不就是上一步的RDD计算出的结果么？为什么说是不保存计算结果？

- Little Spirits 2019-05-16 08:48:53
  多个父分区到一个子分区，对于任何一个父分区而言都是pipeline的所以是窄依赖，而一个父分区到多个子分区对父分区而言不是pipeline的所以是宽依赖

- 一修zz 2019-05-15 23:20:04
  请教老师一个问题，现在有个inference任务使用hadoop streaming，1000个mapper并行执行，没有reducer，需要为每个mapper载入模型 词典等文件，如果改用spark，能否实现多个RDD读取同一块内存的模型和词典，达到节省内存的效果？ 还望老师答复

- miwucc 2019-05-15 21:34:25
  子rdd依赖多个父rdd来产出结果。明显是宽依赖。因为需要等待多个父rdd结果完毕才能开始计算。宽依赖还是窄依赖关键看是否要等待更多父rdd准备完毕。

- linuxfans 2019-05-15 19:53:50
  这个没有疑问吧？既然父子分区是一一对应，当然不存在这种情况了。如果是一子对父的情况，就变成子分区要等待所以父分区的结果，并行的效率降低。

- 蒙开强 2019-05-15 13:36:32
  老师，你好，学习了你这节讲的RDD,我有个问题咨询你一下，如果我要判断这RDD是否有数据，在生产中那种方式是最优的呢。

- 锦 2019-05-15 11:16:14
  文中提到依赖关系的区分考虑基于两点：
  1、性能考虑，窄依赖可以使得每个数据节点并行计算结果，也可以支持链式计算；宽依赖需要父分区的中每个分区中的数据节点计算，只能串行计算。
  2、故障恢复考虑，窄依赖可以更快、更准的恢复数据，宽依赖则相对较慢。
  那么基于以上考虑，父rdd与子rdd是多对多的关系，则划分到宽依赖；一对一、一对多或多对一的关系都可以划分到窄依赖。

  分区方式：hash分区、rang分区，以及自定义分区
  疑问：因为分区指向某个节点中的数据块，那么分区的key是分区在RDD中的index还是其引用的数据块中的某个数据字段？我认为是后者。
  另外，hash分区和rang分区的应用场景分别是什么呢？

  RDD具有不可变性，只能通过转换来创建新的RDD，但是不代表RDD中分区指向的节点数据块也不可变，那么如何保证数据块不可变性呢？我认为可能是使用CopyOnWrite技术实现的。

  Spark优于MapReduce的地方之一在于：MapReduce的中间计算结果实时落盘，而Spark使用存储依赖关系和延迟存储中间数据来提高性能。

- 邱从贤※klion26 2019-05-15 09:42:03
  思考题中窄依赖应该是不关心一个子 rdd 的 partition 对应多少个 父rdd 的 partition，只要partition可以独立计算就行，比如 map 操作，子 rdd 只有 3 个 partition，父 rdd 有6 个 partition，那么父 rdd 肯定有两个 partition 的数据会落到子 rdd 的一个 partition 中，但是落到子 rdd 同一个 partition 的两个 partition 不需要等待就可以进行计算，因此是窄依赖

- 摇山樵客™ 2019-05-15 08:39:26
  1.窄依赖是指父 RDD 的每一个分区都可以唯一对应子 RDD 中的分区，那么是否意味着子 RDD 中的一个分区只可以对应父 RDD 中的一个分区呢？

不是的，比如coalesce这种合并分区的操作中，子rdd需要依赖父rdd的若干个分区，但它不需要全部的分区，是窄依赖

2.如果子 RDD 的一个分区需要由父 RDD 中若干个分区计算得来，是否还算窄依赖？

算。只要纪录层级没发生重新分区，全局混洗，应该都属于窄依赖吧

- CoderLean 2019-05-15 08:27:29
  算啊，父子rdd可以通过函数进行转换，对于转换因子是基本不变的，那也应该支持逆转换。，而一个子rdd是无法推导出父rdd的，因为父rdd的数据是由函数转换后拆分给多个子rdd的。另外，之前没学过spark，对于flink内部算子也没有那么深入的理论，学完这个后要可以回去看看flink的算子是怎么实现的

- 明翼 2019-05-15 08:17:42
  rdd的分区不保存数据什么意思？老师可以对rdd结构再深入讲解不？我认为子rdd在窄依赖中不会对应多个父rdd，才保障单向传递

- Rainbow 2019-05-15 07:52:42
  uion也是窄依赖

- 涵 2019-05-15 05:12:20
  我想还可以算做是窄依赖，因为子RDD分区所依赖的对个父RDD分区是互斥的，所以每个子RDD分区所依赖的多个父RDD分区可以被看做一组分区。父RDD的组分区与子分区是一一对应关系，满足窄依赖可以并行计算，而无需所以父分区都计算完毕才可以开始计算的特性。