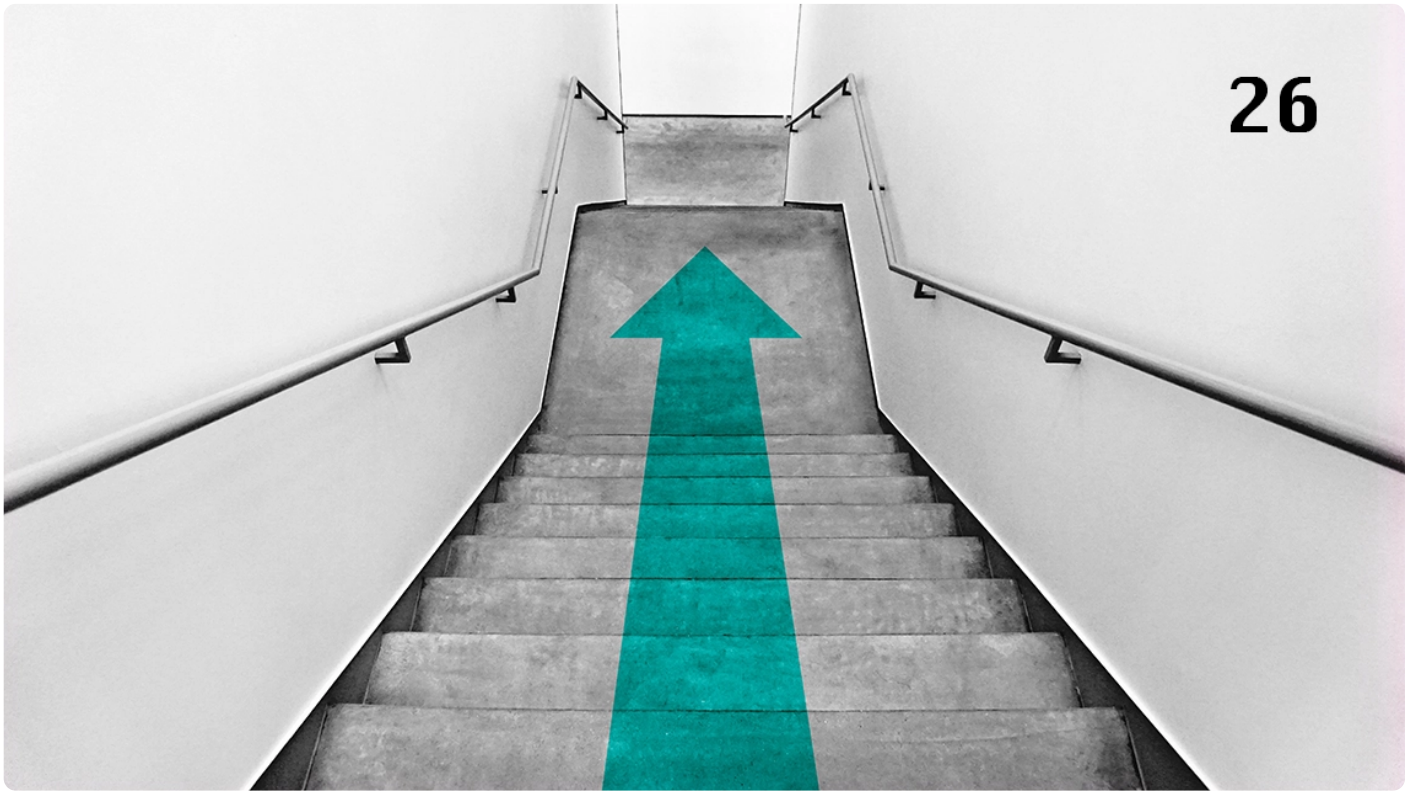
26 | Pipeline：Beam如何抽象多步骤的数据流水线?

2019-06-21 蔡元楠 来自北京

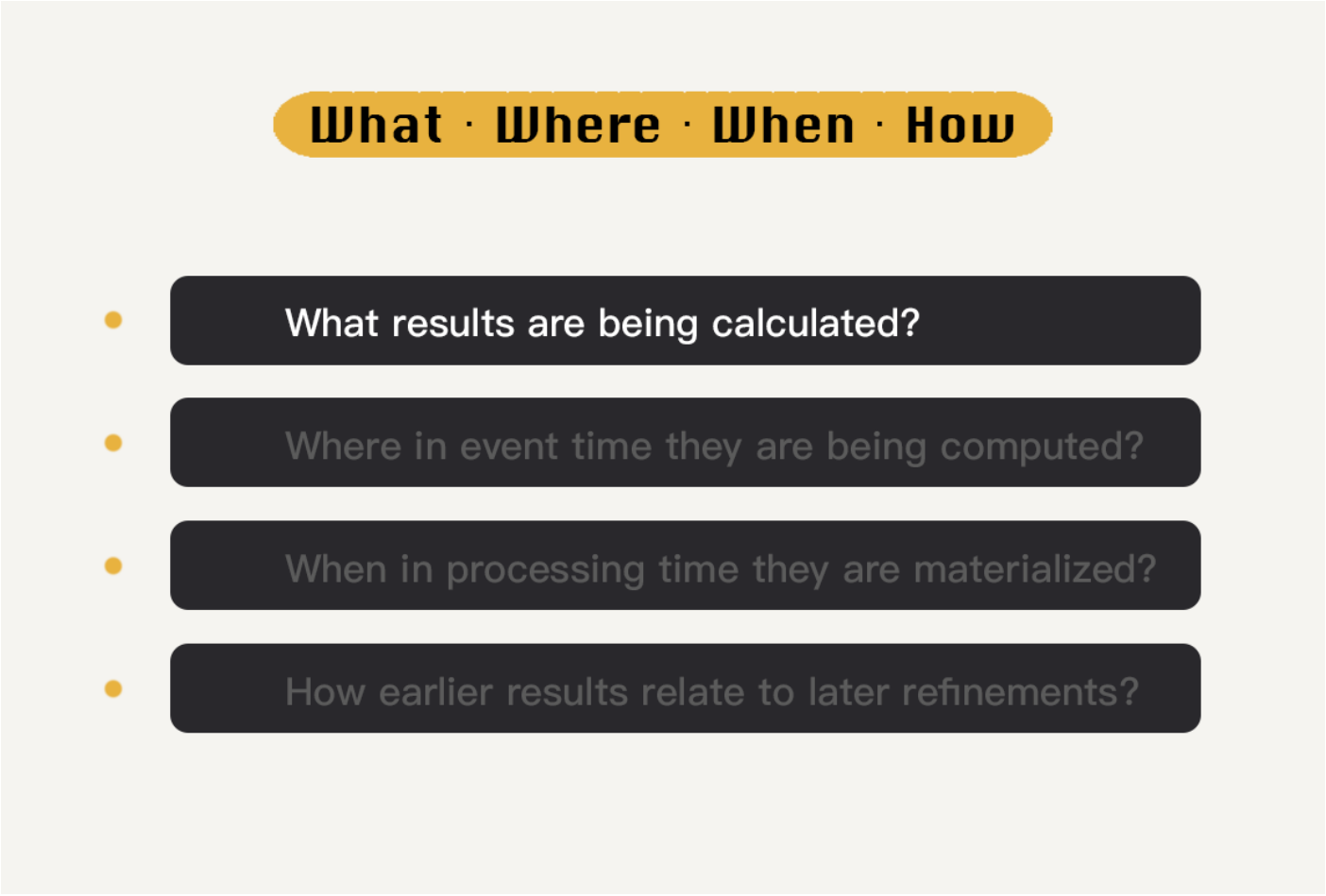
《大规模数据处理实战》



你好，我是蔡元楠。

今天我要与你分享的主题是“Pipeline：Beam 如何抽象多步骤的数据流水线”。

在上两讲中，我们一起学习了 Beam 是如何抽象封装数据，以及如何抽象对于数据集的转换操作的。在掌握了这两个基本概念后，我们就可以很好地回答 Beam 编程模型里的 4 个维度 What、Where、When、How 中的第一个问题——What 了。也就是，我们要做什么计算？想得到什么样的结果？



这个时候你可能已经跃跃欲试，开始想用 PCollection 和 Transform 解决我们平常经常会使用到的批处理任务了。没有问题，那我们就先抛开 Where、When 和 How 这三个问题，由简至繁地讲起。

现在假设我们的数据处理逻辑只需要处理有边界数据集，在这个情况下，让我们一起来看看 Beam 是如何运行一套批处理任务的。

数据流水线

在 Beam 的世界里，所有的数据处理逻辑都会被抽象成**数据流水线（Pipeline）**来运行。那么什么是数据流水线呢？

Beam 的数据流水线是对于数据处理逻辑的一个封装，它包括了从**读取数据集**，**将数据集转换成想要的结果**和**输出结果数据集**这样的一整套流程。

所以，如果我们想要跑自己的数据处理逻辑，就必须在程序中创建一个 Beam 数据流水线出来，比较常见的做法是在 main() 函数中直接创建。

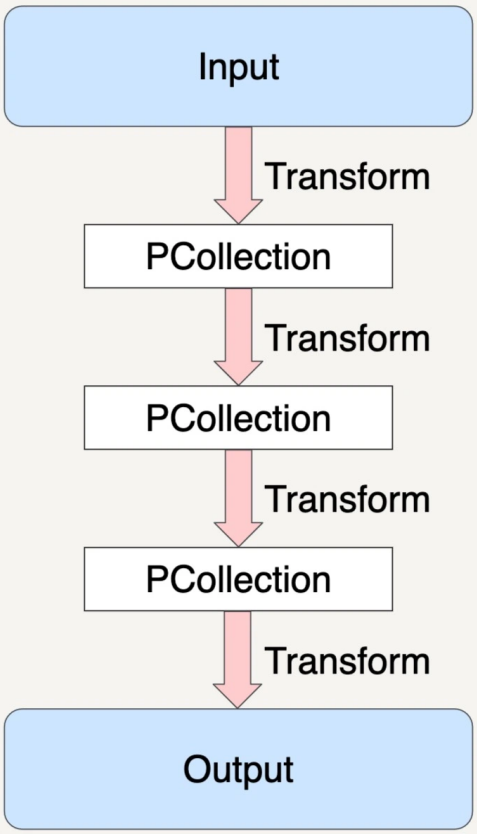
Java

```
PipelineOptions options = PipelineOptionsFactory.create();
Pipeline p = Pipeline.create(options);
```

在创建 Beam 数据流水线的同时，我们必须给这个流水线定义一个**选项**（Options）。这个选项会告诉 Beam，用户的 Pipeline 应该如何运行。例如，是在本地的内存上运行，还是在 Apache Flink 上运行？关于具体 Beam 选项的解释，我会在第 30 讲中展开讲解。

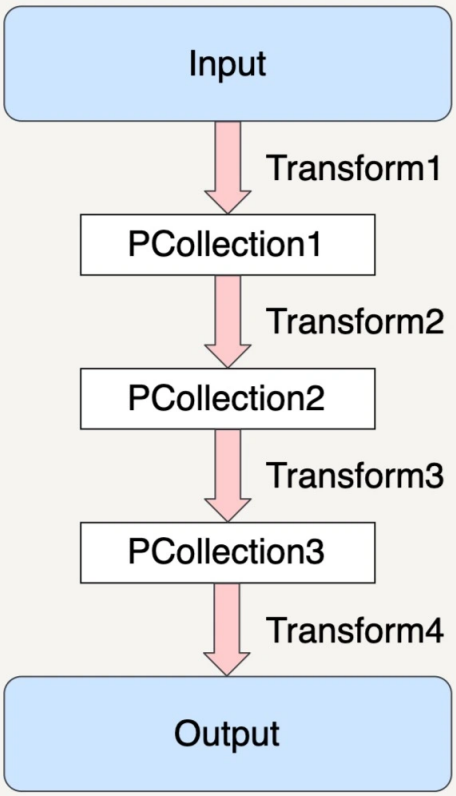
Beam 数据流水线的应用

有了数据流水线这个抽象概念之后，我们就可以将 PCollection 和 Transform 应用在这个流水线里面了。



上图就是一个 Beam 的数据流水线，整个数据流水线包括了从读取数据，到经过了 N 个 Transform 之后输出数据的整个过程。

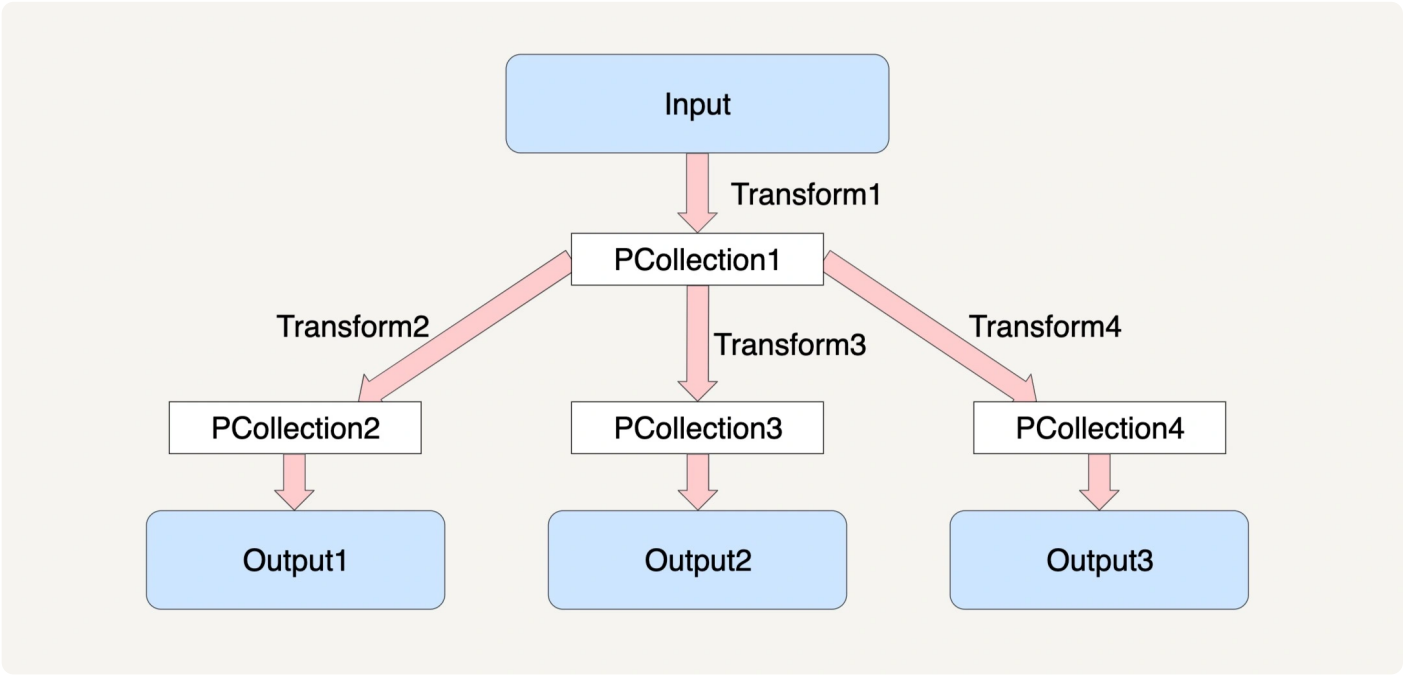
在第 24 讲中我们学习过 PCollection 的不可变性。也就是说，一个 PCollection 一经生成，我们就不能够再增加或者删除它里面的元素了。所以，在 Beam 的数据流水线中，每次 PCollection 经过一个 Transform 之后，流水线都会新创建一个 PCollection 出来。而这个新的 PCollection 又将成为下一个 Transform 的新输入。



在上图的示例中，Beam 数据流水线在经过 Transform1 读取了输入数据集之后，会创建出一个新的 PCollection1，而经过了 Transform2 之后，数据流水线又会创建出新的 PCollection2 出来，同时 PCollection1 不会有任何改变。也就是说，在上面的例子中，除去最终的输出结果，数据流水线一共创建了 3 个不同的 PCollection 出来。

这种特性可以让我们在编写数据处理逻辑的时候，对同一个 PCollection 应用多种不同的 Transfrom。

例如下图所示，对于 PCollection1，我们可以使三个不同的 Transform 应用在它之上，从而再产生出三个不同的 PCollection2、PCollection3 和 PCollection4 出来。



Beam 数据流水线的处理模型

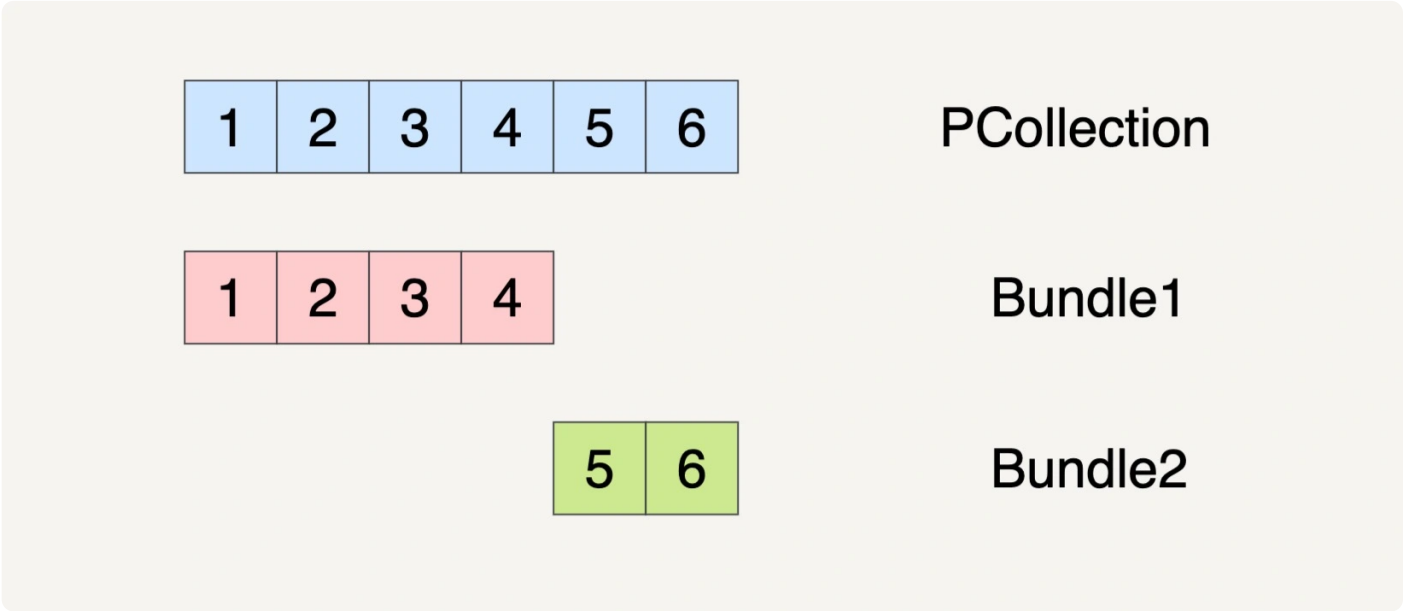
在了解完 Beam 数据流水线高度抽象的概念后，紧接着，我想和你介绍一下 Beam 数据流水线的处理模型，也就是数据流水线在运行起来之后，会发生些什么，它是如何处理我们定义好的 PCollection 和 Transform 的。

Beam 数据流水线的底层思想其实还是动用了 MapReduce 的原理，在分布式环境下，整个数据流水线会启动 N 个 Workers 来同时处理 PCollection。而在具体处理某一个特定 Transform 的时候，数据流水线会将这个 Transform 的输入数据集 PCollection 里面的元素分割成不同的 Bundle，将这些 Bundle 分发给不同的 Worker 来处理。

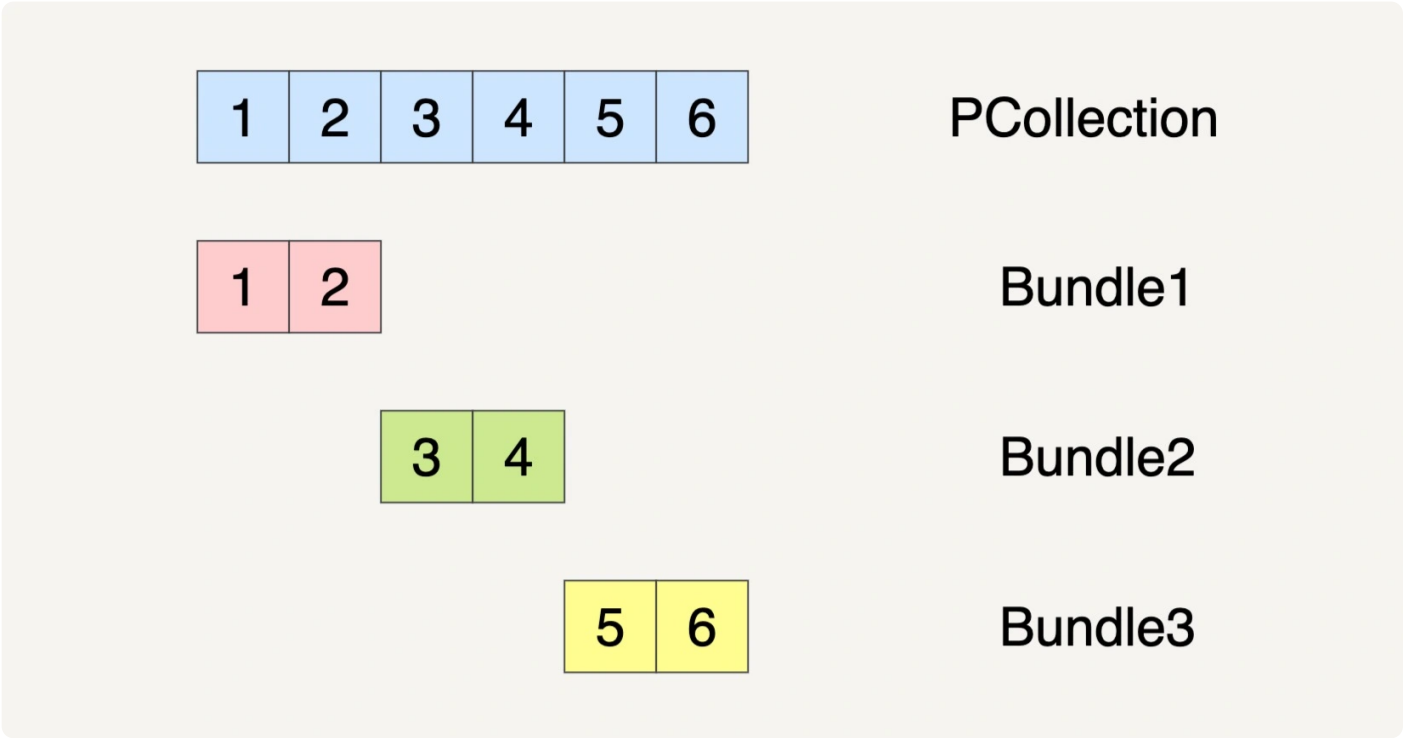
Beam 数据流水线具体会分配多少个 Worker，以及将一个 PCollection 分割成多少个 Bundle 都是随机的。但 Beam 数据流水线会尽可能地让整个处理流程达到**完美并行**（Embarrassingly Parallel）。

我想举个几个例子让你更好地来理解这个概念。

假设在数据流水线的一个 Transform 里面，它的输入数据集 PCollection 是 1、2、3、4、5、6 这个 6 个元素。数据流水线可能会将这个 PCollection 按下图的方式将它分割成两个 Bundles。

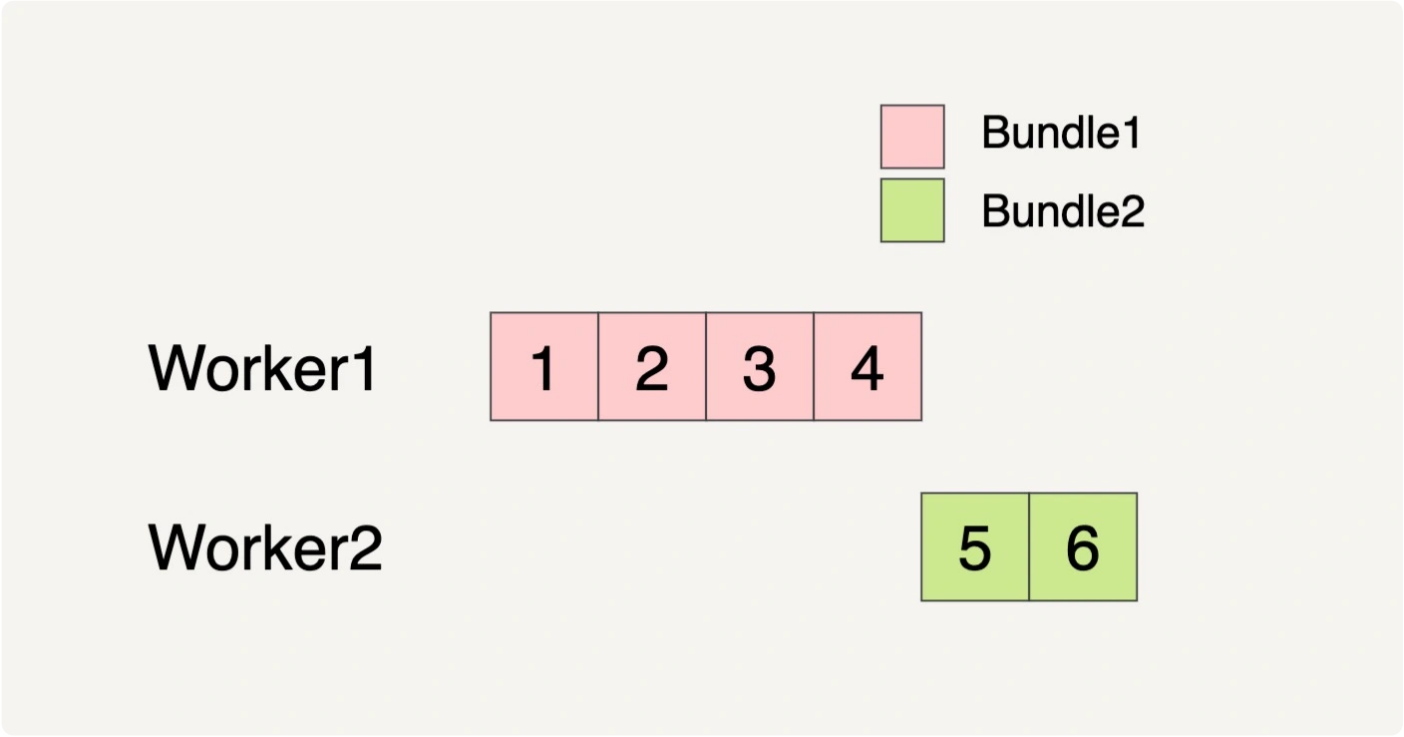


当然，PCollection 也有可能会被分割成三个 Bundles。

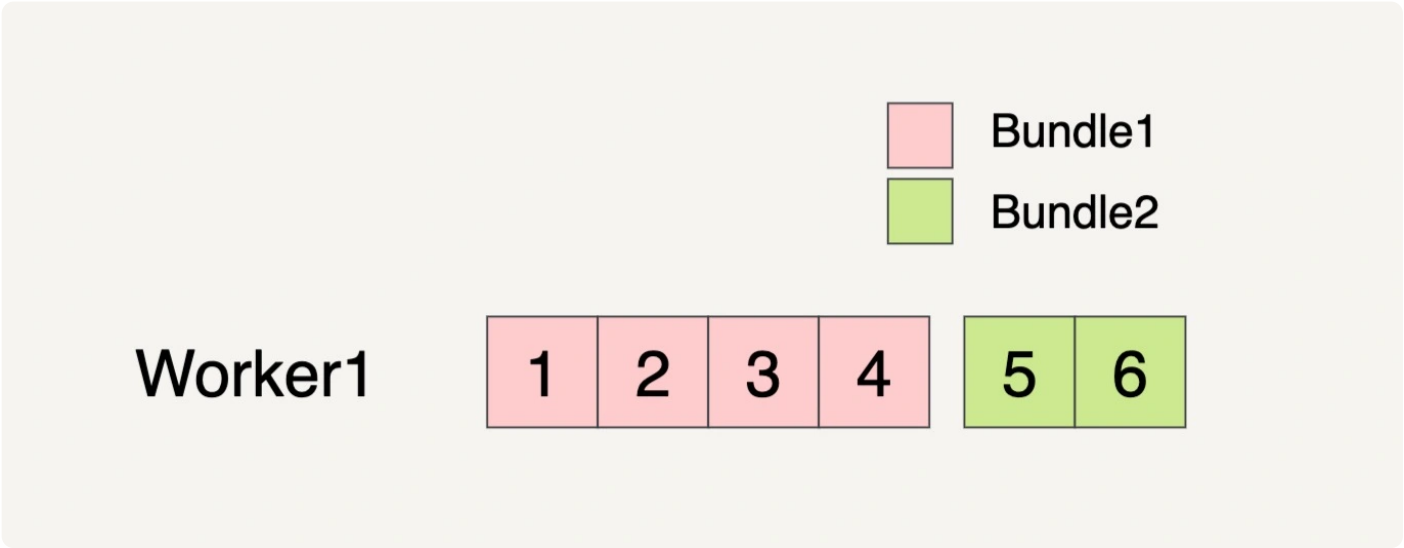


那数据流水线会启用多少个 Worker 来处理这些 Bundle 呢？这也是任意的。还是以刚刚的 PCollection 输入数据集作为例子，如果 PCollection 被分割成了两个 Bundles，数据流水线

有可能会分配两个 Worker 来处理这两个 Bundles。



甚至有可能只分配一个 Worker 来处理这两个 Bundles。

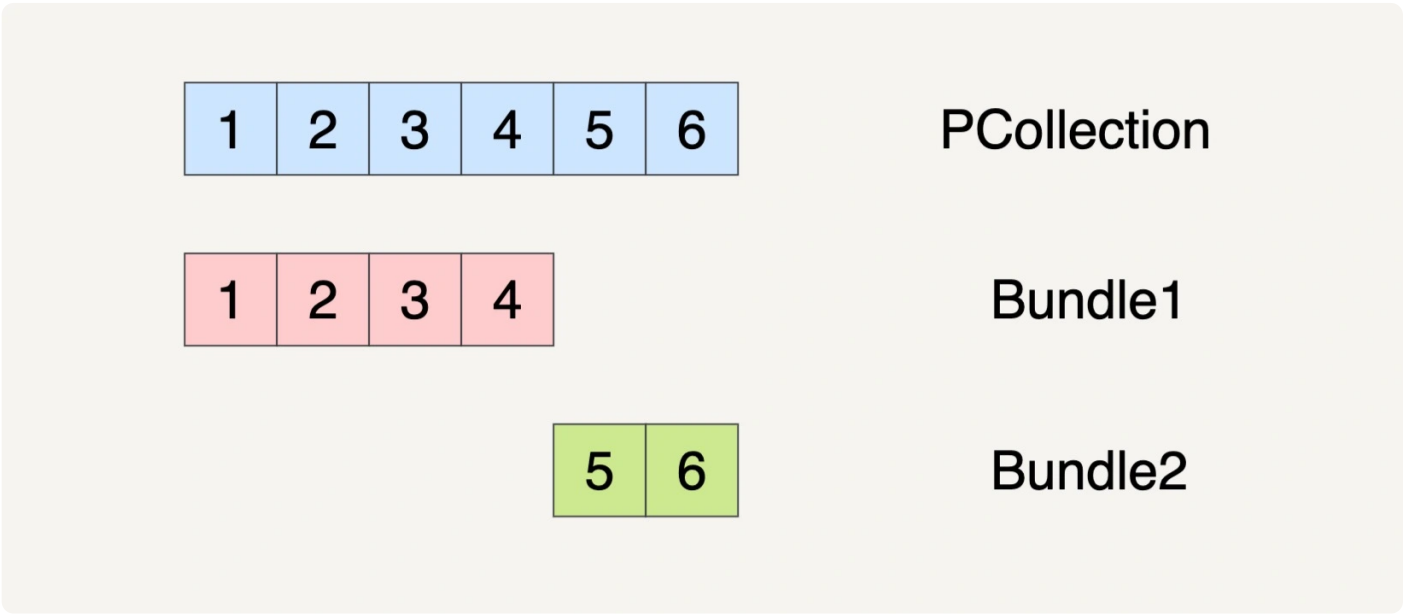


在多步骤的 Transforms 中，一个 Bundle 通过一个 Transform 产生出来的结果会作为下一个 Transform 的输入。

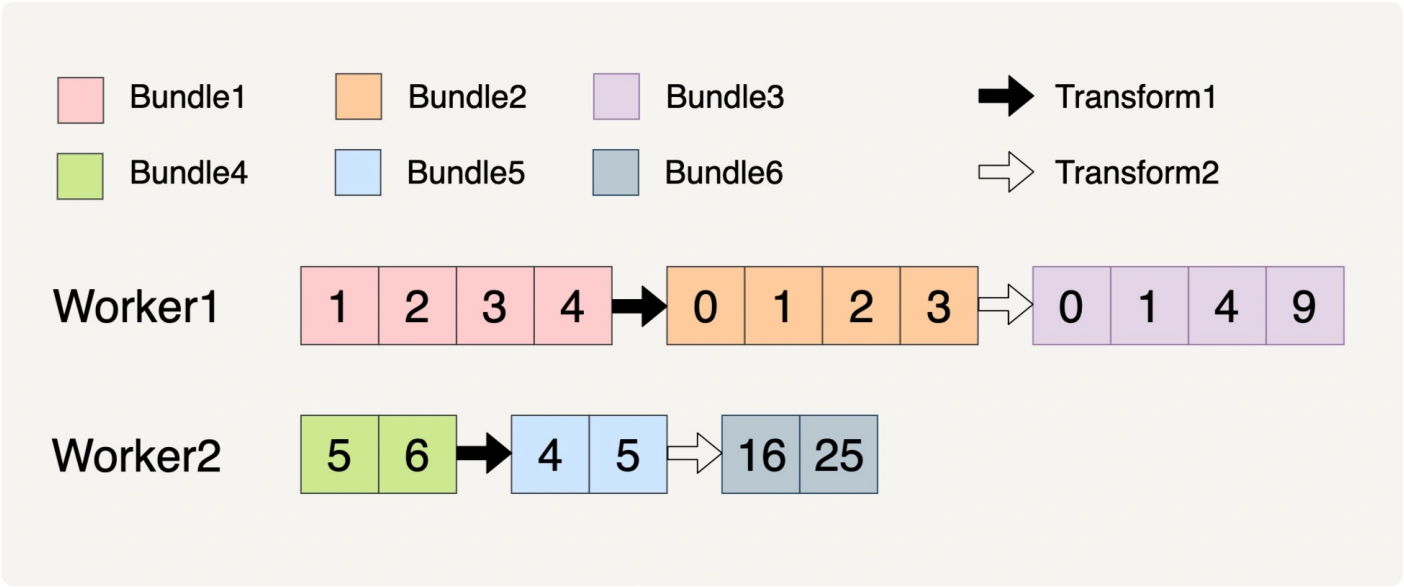
之前刚刚讲过，在 Beam 数据流水线中，抽象出来的 PCollection 经过一个 Transform 之后，流水线都会新创建一个 PCollection 出来。同样的，Beam 在真正运行的时候，每一个

Bundle 在一个 Worker 机器里经过 Transform 逻辑后，也会产生出来一个新的 Bundle，它们也是具有不可变性的。像这种具有关联性的 Bundle，必须在同一个 Worker 上面处理。

我现在来举例说明一下上面的概念。现在假设输入数据集如下图所示，它被分成了两个 Bundles。



我们现在需要做两个 Transforms。第一个 Transform 会将元素的数值减一；第二个 Transform 会对元素的数值求平方。整个过程被分配到了两个 Workers 上完成。



过程就如上图所示，总共产生了 6 个不可变的 Bundle 出来，从 Bundle1 到 Bundle3 的整个过程都必须放在 Worker1 上完成，因为它们都具有关联性。同样的，从 Bundle4 到 Bundle6 的整个过程也都必须放在 Worker2 上完成。

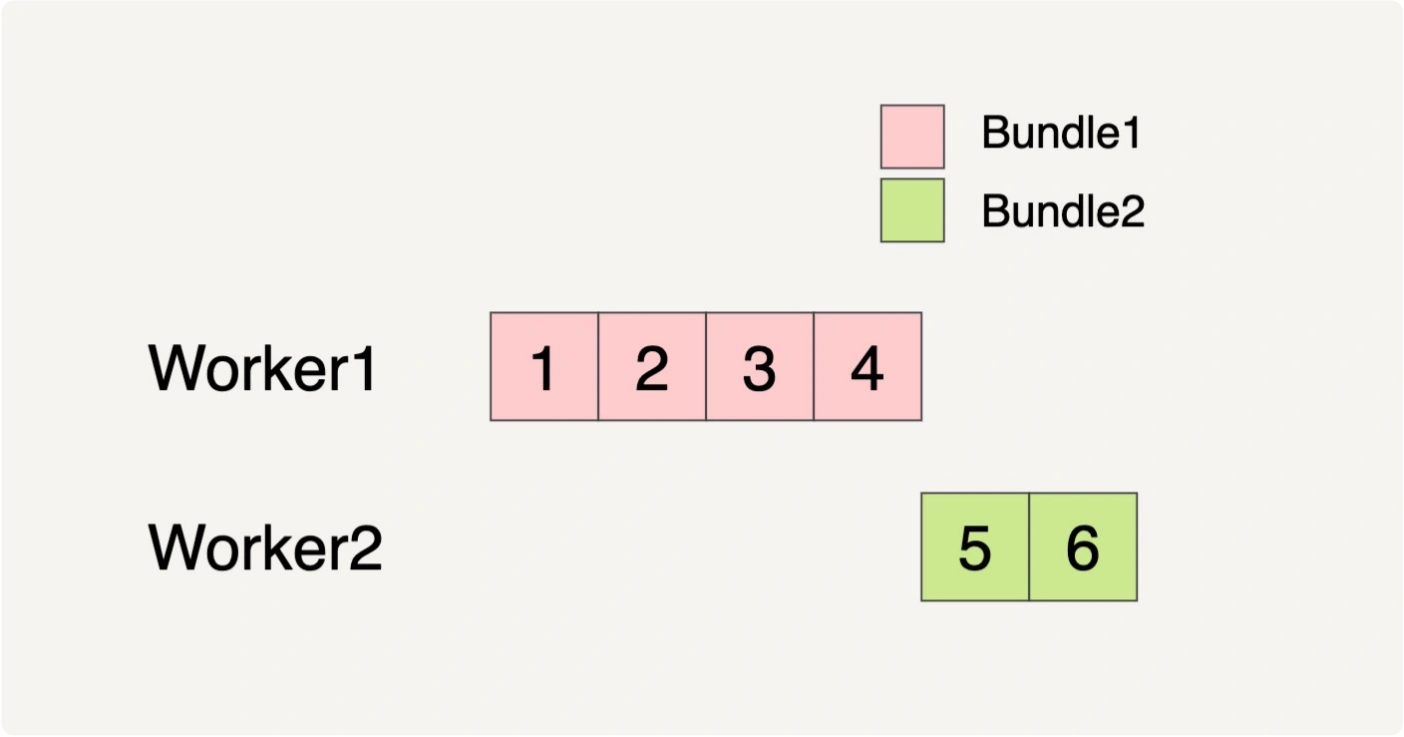
Beam 数据流水线的错误处理

在学习完 Beam 数据流水线底层的处理模型之后，你可能会有个疑问：既然 Bundle 都是放在分布式环境下处理的，要是其中一个步骤出错了，那数据流水线会做什么样的处理？接下来我会给你讲解一下 Beam 数据流水线的错误处理机制。

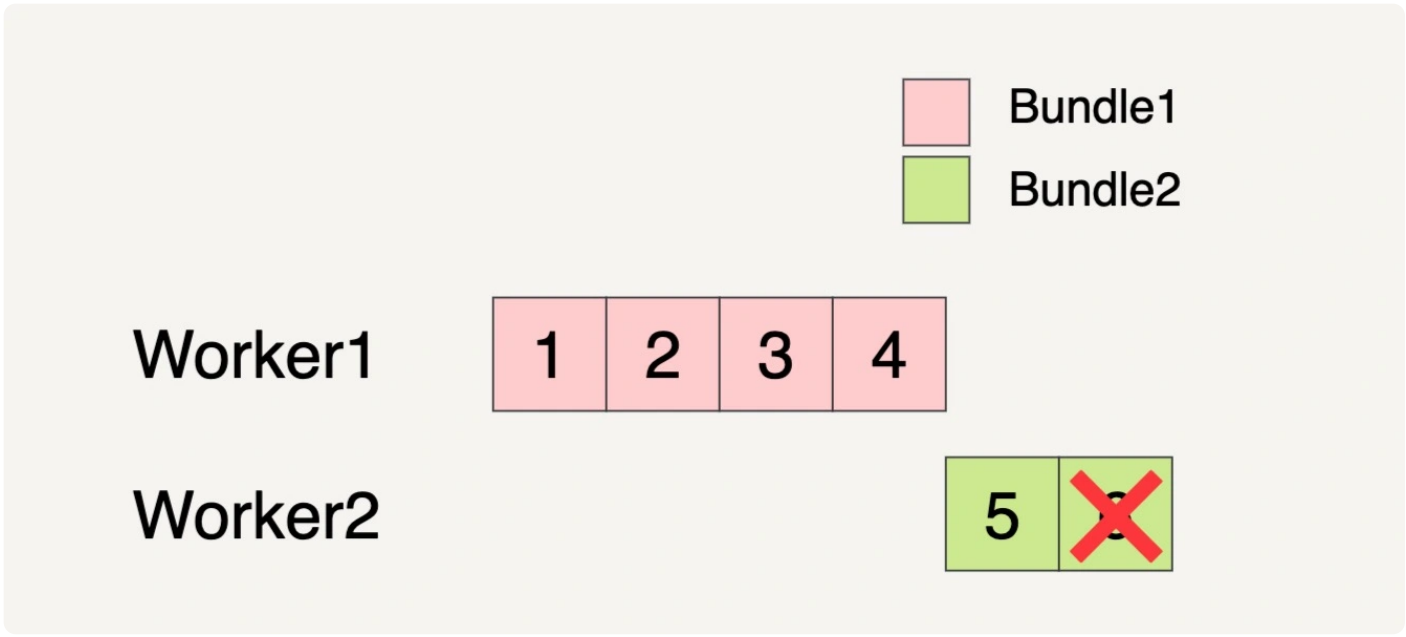
单个 Transform 上的错误处理

我们还是以单个 Transform 开始讲解。在一个 Transform 里面，如果某一个 Bundle 里面的元素因为任意原因导致处理失败了，则这整个 Bundle 里的元素都必须重新处理。

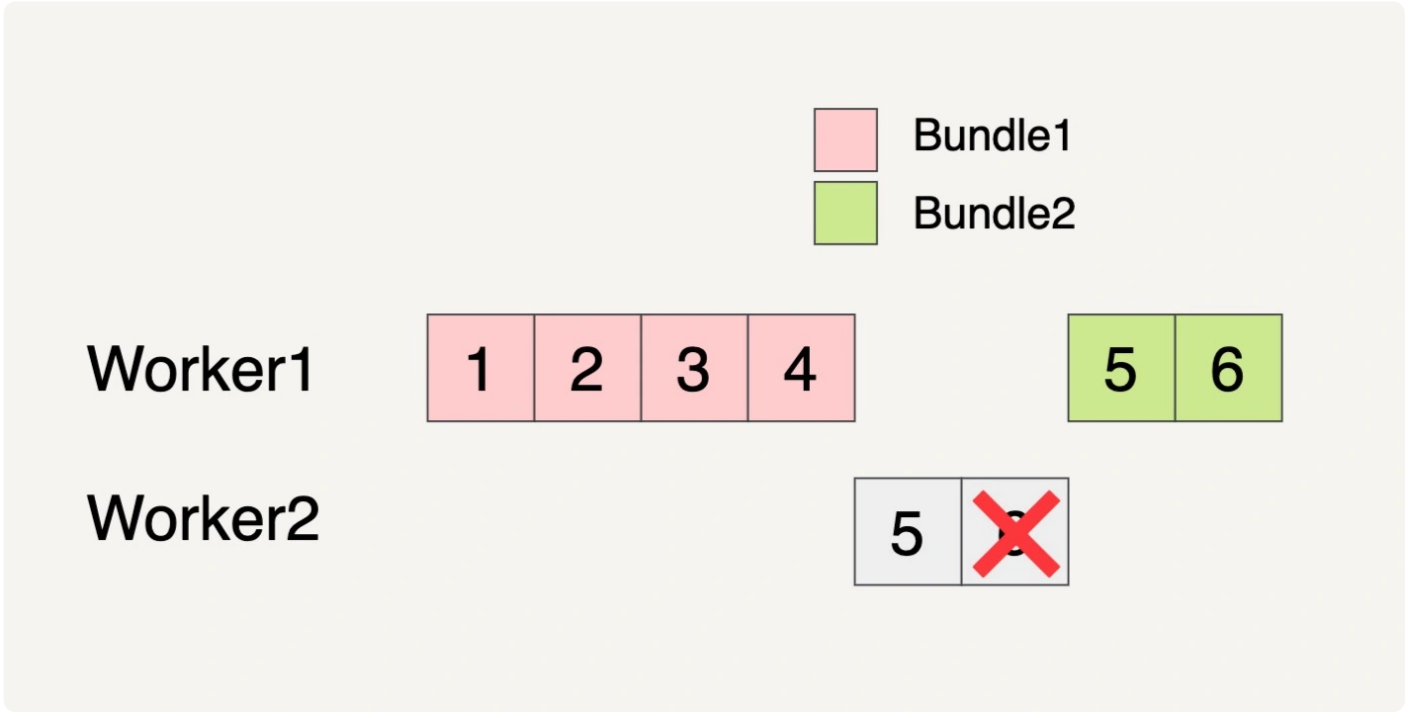
还是假设输入数据集如下图所示，被分成了两个 Bundles。



Beam 数据流水线分配了两个 Worker 来处理这两个 Bundles。我们看到下图中，在 Worker2 处理 Bundle2 的时候，最后一个元素 6 处理失败了。



这个时候，即便 Bundle2 的元素 5 已经完成了处理，但是因为同一个 Bundle 里面的元素处理失败，所以整个 Bundle2 都必须拿来重新处理。



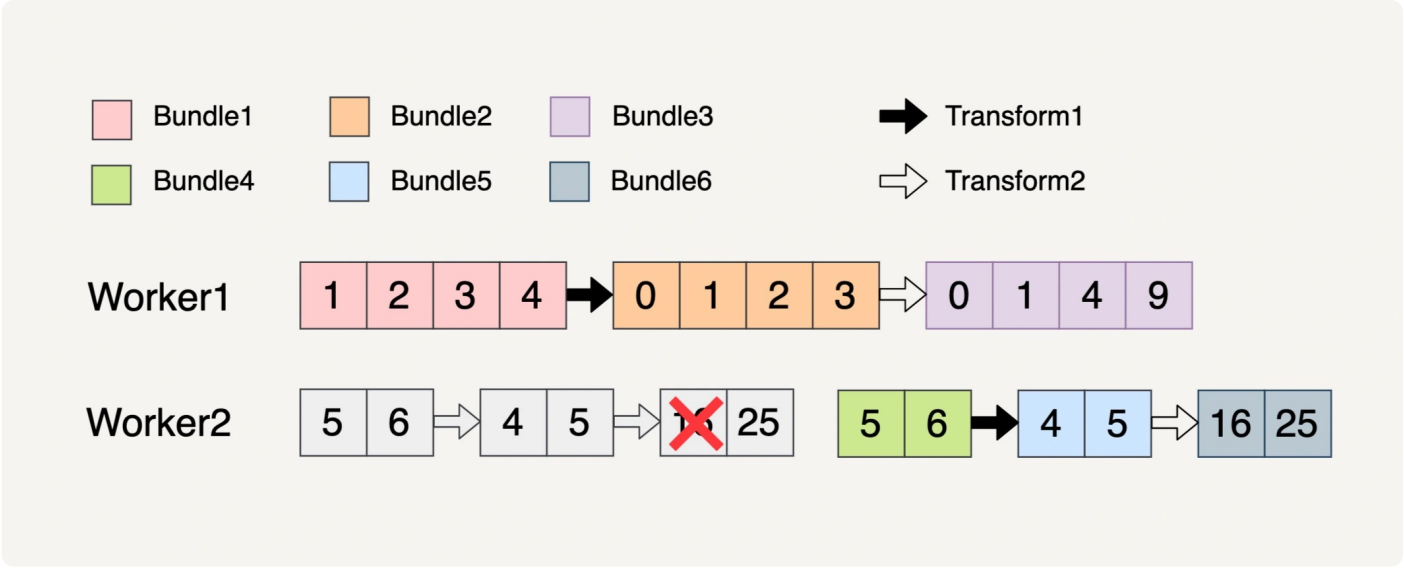
重新处理的 Bundle 也不一定要在原来的 Worker 里面被处理，有可能会被转移到另外的 Worker 里面处理。如上图所示，需要重新被处理的 Bundle2 就被转移到 Worker1 上面处理了。

多步骤 Transform 上的错误处理

学习完单个 Transform 上的错误处理机制，我们再来看看在多步骤的 Transform 上发生错误时是如何处理的。

在多步骤的 Transform 上，如果处理的一个 Bundle 元素发生错误了，则这个元素所在的整个 Bundle 以及与这个 Bundle 有关联的所有 Bundle 都必须重新处理。

我们还是用上面的多步骤 Transform 来讲解这个例子。



你可以看到，在 Worker2 中，处理 Transform2 逻辑的时候生成 Bundle6 里面的第一个元素失败了。因为 Bundle4、Bundle5 和 Bundle6 都是相关联的，所以这三个 Bundle 都会被重新处理。

小结

今天我们一起学习了 Beam 里对于数据处理逻辑的高度抽象数据流水线，以及它的底层处理模型。数据流水线是构建数据处理的基础，掌握了它，我们就可以根据自身的应用需求，构建出一套数据流水线来处理数据。

思考题

你能根据自己的理解重述一下在 Beam 的数据流水线中，当处理的元素发生错误时流水线的错误处理机制吗？

欢迎你把答案写在留言区，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。

精选留言 (18)

espzest
2019-06-21

bundle怎么聚合成pcollection？ 一个bundle处理失败，为什么需要重做前面的bundle?

作者回复: 谢谢你的提问！其实我们在第24讲中知道，PCollection是具有无序性的，所以最简单的做法bundle在处理完成之后可以直接append到结果PCollection中。

至于为什么需要重做前面的bundle，这其实也是错误处理机制的一个trade-off了。Beam希望尽可能减少persistence cost，也就是不希望将中间结果保持在某一个worker上。可以这么想，如果我们想要不重新处理前面的bundle，我们必须要将很多中间结果转换成硬盘数据，这样一方面增加很大的时间开销，另一方面因为数据持久化了在具体一台机器上，我们也没有办法再重新动态分配bundle到不同的机器上去了。

16

cricket1981
2019-06-21

bundle随机分配会不会产生数据倾斜？完美并行背后的机制是？beam应该也有类似spark的persist方法缓存转换中间结果，防止出错恢复链太长吧?

作者回复: 谢谢你的提问！其实文章中所讲到的随机分配并不是说像分配随机数那样将bundle随机分配出去给workers，只是说根据runner的不同，bundle的分配方式也会不一样了，但最终还是还是希望能最大化并行度。

至于完美并行的背后机制，Beam会在真正处理数据前先计算优化出执行的一个有向无环图，希望保持并行处理数据的同时，能够减少每个worker之间的联系。

Beam据我所知是有的，BEAM-7131 issue就有反应这个问题。

共 2 条评论 >

9

YZJ
2019-06-21

老师请教个问题：PCollectionA transform PCollectionB, 假如PCollectionB 要比PCollectionA大很多倍，比如transform 是把PCollectionA 中每个字符串重复1000次，那PCollectionB 就要大1000倍，worker会不会有内存溢出问题? spark中可以配置executor 的core和memery来控制每个task内存用量，beam有类似机制吗?不然怎样让资源利用最优化呢?

作者回复: 谢谢你的提问！这个问题很好啊，运行大型的Beam pipeline遇到OOM也是有可能的。要配置底层资源的话要看Runner支不支持，像如果将Google Cloud Dataflow作为Runner的话，我们可以通过配置PipelineOption来达到目的。底层使用Spark的话我个人还没有使用过，不过应该是可以用SparkContextOptions来配置的。

8

aof
2019-06-26

Beam的错误处理和RDD的真的很像，因为transformation都是lazy的，只有action才会触发计算，中间的转换过程都是被记录在DAG中的，这就导致中间某个transformation失败之后，需要往上追溯之前的转换，可以理解为是寻找父transformation，然后父transformation还要往上寻找父父transformation，直到没有父transformation为止，就像是类加载机制一样。但是如果能把中间结果保存在内存中，在失败重新计算时，就能提高计算的效率。

作者回复: 的确如此

6

沈洪彬
2019-06-23

在 Beam 的数据流水线中，当处理的元素发生错误时流水线的的错误处理机制分两种情况

1.单个Transform上的错误处理
如果某个Bundle里元素处理失败，则整个Bundle里元素都必须重新处理

2.多步骤Transform上的错误处理
如果某个Bundle里元素处理失败，则整个Bundle里元素及与之关联的所有Bundle都必须重新处理

作者回复: 谢谢留言！总结得不错！

4

onepieceJT2018
2019-06-21

老师 想到一个问题啊 如果有个计算是 需要worker1 和 worker2 都算完的结果再计算 发生worker1 一直错误没通过 这时候worker2会一直傻傻等待嘛

作者回复: 谢谢你的提问！这个依赖worker1和worker2计算结果的Transform会一直等待。但是与此同时，worker2可以做其它的计算，甚至有可能worker1的计算如果一直出错，Beam会将这个bundle重新分配给worker2来计算。

4

常超
2019-06-21

<在多步骤的 Transform 上，如果处理的一个 Bundle 元素发生错误了，则这个元素所在的整个 Bundle 以及与这个 Bundle 有关联的所有 Bundle 都必须重新处理。
如果upstream transform里状态有更新操作，重新处理已经成功的bundle会出现数据重复，可能导致状态更新不正确吧?

作者回复: 谢谢你的提问！这个问题非常好啊，如果你所说的是stateful processing的话，那它的错误处理机制和stateful-less会不太一样。Stateful processing里的ParDo在处理每一个元素的时候会将它的state持久化，也就是保存到外部的storage中，下次需要用到这个元素的时候会再读取这个元素的state。如果是发生了错误的话，会有机制reclaim这些states或者是invalidate它们。

3

Alpha
2019-06-21

上一期讲到，PCollection 是有向图中的边，而 Transform 是有向图里的节点。这一期的图咋又变了呢

作者回复: 谢谢留言！哈哈，需要表达的内容不一样了。

1

哇哈哈

2022-03-13

流式数据也会分bundle吗？那不是变成spark streaming的微型批处理形式了？

weiming

2022-01-12

1. PCollection会被划分成多个Bundle（分配多少个是随机的），Bundle会被分配到Worker中处理（分配也是随机的），最终机制保障最大程度的完美并行。
2. 错误处理中有关联Bundle的概念（因为是同一个Worker处理），如果关联Bundle中的一个Bundle失败了，所有关联的Bundle全部重做，主要是考虑到数据持久化的成本。（通过重做消除持久化）
3. 注意有PCollection不可变性，引申到Bundle中的不可变性。

旭东(Frank)

2021-01-14

分而治之

冯杰

2020-04-11

从您的描述中可以看出，数据的实际计算和容错都是以分区来进行的，原因在于ParDo模式下同一个Pc下不同的数据记录之间不存在依赖关系即可以完成计算。　　在实际计算时，我们处理玩Trasform1得到Pc1，然后在接着计算transform2，那为什么不能以单条数据来并行呢？
即分区内的每一条数据独立完成所有的计算链，而不是要等同一个Pc下的数据都就绪后在执行下一个计算。
关于容错不以单条数据来设计，我倒是能理解，因为要这样做的话，我们必然需要为每条数据都记录他的计算关系，追溯它具体是从上游的哪一条数据来的，这会增加存储的压力。 而以分区来实现容错，我们只需要记录血缘即可，血缘关系太长，可以像Spark那样做一些持久化的操作。

jimyth
2019-07-26

老师你好，既然PCollection 是无序的，请问一下怎么处理数据流中的先后依赖的问题，本节例子的 bound中的数据都是有序的分配的，实际计算过程中是不是会出现 1,3,5出现在一个 bound ;2,4,6 出现在一个 bound
您在 23 讲的例子中，ParDo 是针对单个元素的处理，怎么实现计算2 个元素的累加的呢?
例如下面是一组速度数据
时间　　速度
2019-07-26 00:00:00　10
2019-07-26 00:00:01　15
2019-07-26 00:00:02　20
2019-07-26 00:00:03　40
2019-07-26 00:00:04　70
我需要大概怎么计算加速度，

 　　  1

chief
2019-07-04

老师您好，bundle经过Transform会产生新的bundle，那么是同时保留前后bundle数据还是在新生成的bundle中保留血缘关系?

作者回复: 谢谢你的留言！前后bundle保不保留这个还要看你的执行DAG需不需要还用到这个bundle。至于保留前后关系的话主要是用于在发生错误的情况下重新trace back到最开始的那个transform，这个信息从DAG中就可以找到了。

fy
2019-06-23

老师，有编程语言基础。我也去Beam看了看教程，请问这个可以直接学吧。还需要其他基础么，比如操作系统，计算机组成原理等

作者回复: 谢谢你的留言！确实是可以直接学习，不过我也建议如果平时有时间的话，可以看看这些计算机的基础。像操作系统的话里面的一些调度算法或许可以给你平时的实际应用有一些启发。

TJ
2019-06-21

能否说一下Beam和底层执行系统的边界在哪里？那些功能由Beam提供,那些由底层如Spark提供?
如果底层是spark，是否PCollection就是RDD?

作者回复: 谢谢你的留言！如果我没有理解错你的问题的话，我想你说的"边界"指的就是在第23讲中讲到的Beam的统一模型层了。

其实Beam提供的是在Dataflow Model论文里面的一种批流统一处理的思想，数据处理引擎如果能够按照这个思想提供出相应的APIs的话那这个数据处理引擎就可以成为Beam的底层Runner。

最后一问的话是的，PCollection抽象就是Spark中的RDD。


JohnT3e
2019-06-21

由于beam优化器，是不是实际产生的bundle要少于逻辑上的个数?

作者回复: 谢谢你的留言！你这样的理解其实也没有错，如果Beam优化器能优化合并掉一些步骤的话，那确实实际产生出来的bundle会比理论上可以产生出来的bundle要少。


dancer
2019-06-21

想问老师，一个bundle的数据必须要全部处理完之后才能进行第二个transform吗？如果部分数据经过transform1后就可以继续执行transform2，这样数据并行度会更高吧，为什么没有采用这种机制呢?

作者回复: 谢谢你的提问！这个问题还是要看情况吧，如果多步骤的Transform都是ParDo的话，那确实可以按照你说的做法去做。不过当Transform涉及到Combine或者Flatten这种Transform的话， 那就必须等到这一阶段所有的Transform完成了之后才能够进行下一步的Transform了。



1