16 | Spark Streaming：Spark的实时流计算API

2019-05-24 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内容是“Spark Streaming”。

通过上一讲的内容，我们深入了解了 Spark SQL API。通过它，我们可以像查询关系型数据库一样查询 Spark 的数据，并且对原生数据做相应的转换和动作。

但是，无论是 DataFrame API 还是 DataSet API，都是基于批处理模式对静态数据进行处理的。比如，在每天某个特定的时间对一天的日志进行处理分析。

在第二章中你已经知道了，批处理和流处理是大数据处理最常见的两个场景。那么作为当下最流行的大数据处理平台之一，Spark 是否支持流处理呢？

答案是肯定的。

早在 2013 年，Spark 的流处理组件 Spark Streaming 就发布了。之后经过好几年的迭代与改进，现在的 Spark Streaming 已经非常成熟，在业界应用十分广泛。

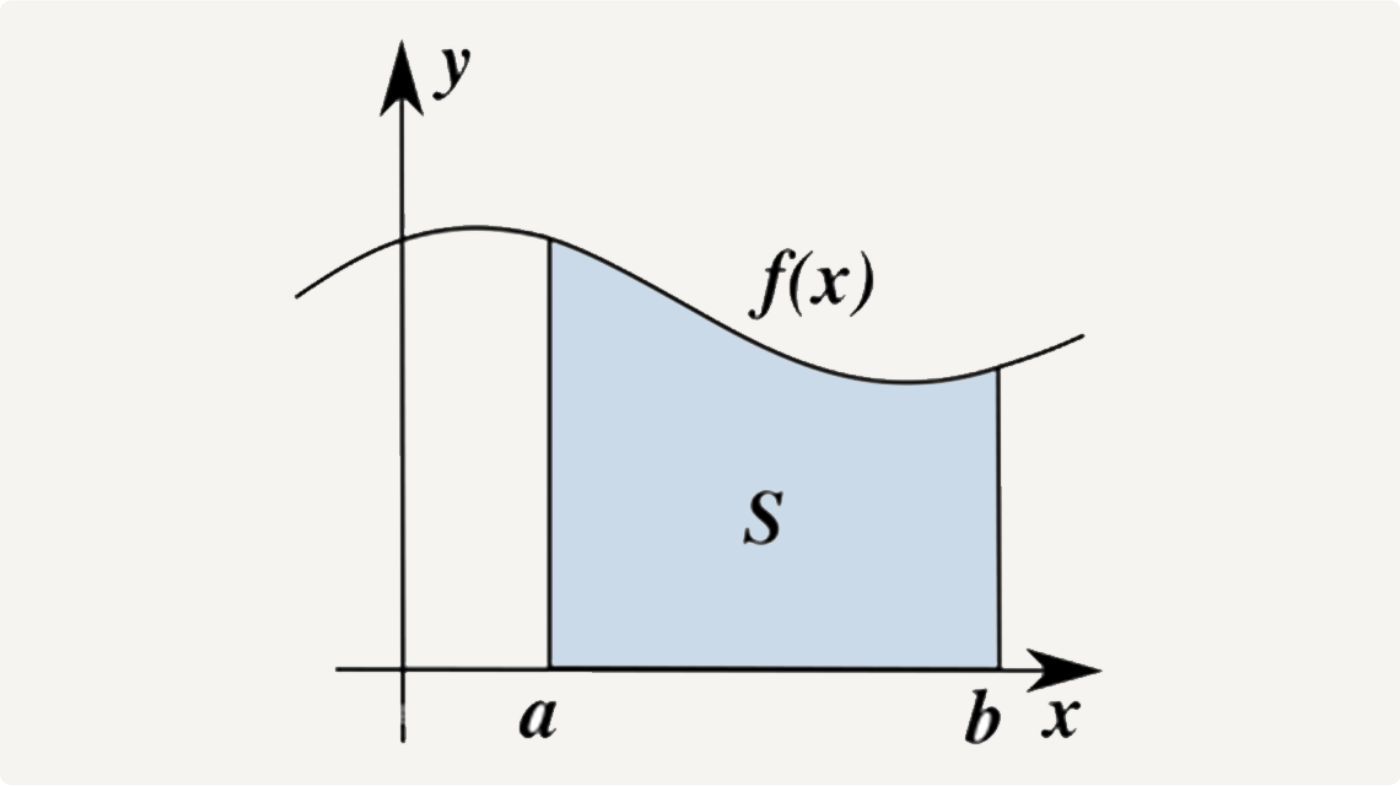
今天就让我们一起揭开 Spark Streaming 的神秘面纱，让它成为我们手中的利器。

Spark Streaming 的原理

Spark Streaming 的原理与微积分的思想很类似。

在大学的微积分课上，你的老师一定说过，微分就是无限细分，积分就是对无限细分的每一段进行求和。它本质上把一个连续的问题转换成了无限个离散的问题。

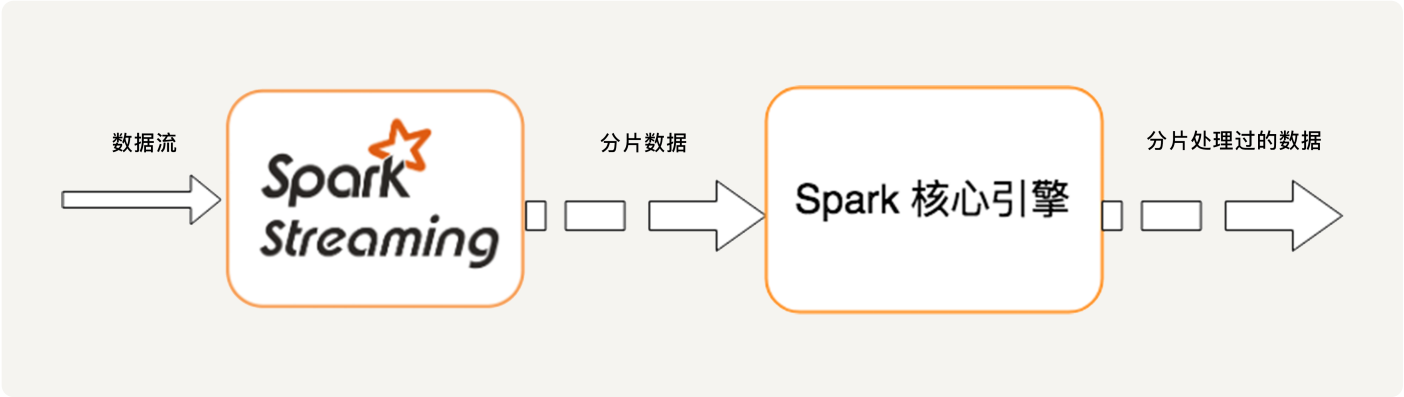
比如，用微积分思想求下图中阴影部分 S 的面积。



我们可以把 S 无限细分成无数个小矩形，因为矩形的宽足够短，所以它顶端的边近似是一个直线。这样，把容易计算的矩形面积相加，就得到不容易直接计算的不规则图形面积。

你知道，流处理的数据是一系列连续不断变化，且无边界的。我们永远无法预测下一秒的数据是什么样。Spark Streaming 用时间片拆分了无限的数据流，然后对每一个数据片用类似于批

处理的方法进行处理，输出的数据也是一块一块的。如下图所示。



Spark Streaming 提供一个对于流数据的抽象 DStream。DStream 可以由来自 Apache Kafka、Flume 或者 HDFS 的流数据生成，也可以由别的 DStream 经过各种转换操作得来。讲到这里，你是不是觉得内容似曾相识？

没错，底层 DStream 也是由很多个序列化的 RDD 构成，按时间片（比如一秒）切分成的每个数据单位都是一个 RDD。然后，Spark 核心引擎将对 DStream 的 Transformation 操作变为针对 Spark 中对 RDD 的 Transformation 操作，将 RDD 经过操作变成中间结果保存在内存中。

之前的 DataFrame 和 DataSet 也是同样基于 RDD，所以说 RDD 是 Spark 最基本的数据抽象。就像 Java 里的基本数据类型（Primitive Type）一样，所有的数据都可以用基本数据类型描述。

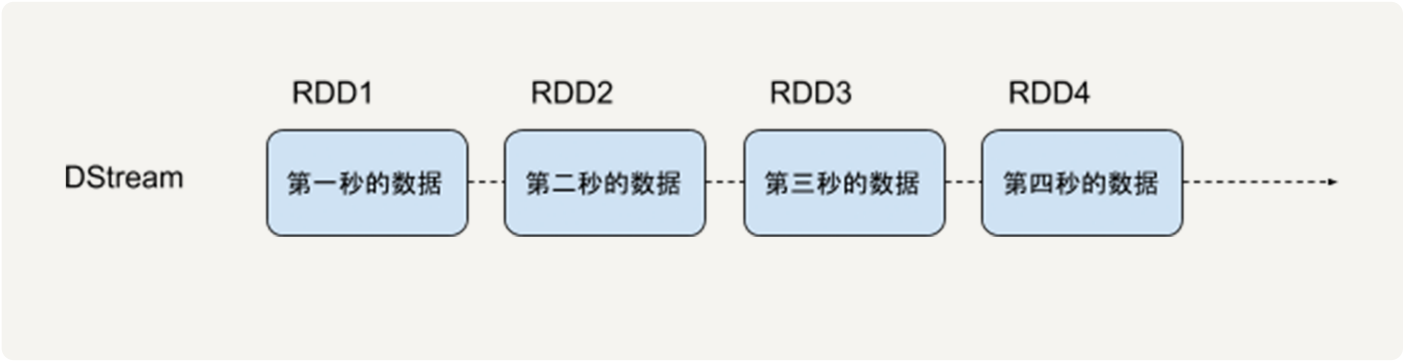
也正是因为这样，无论是 DataFrame，还是 DStream，都具有 RDD 的不可变性、分区性和容错性等特质。

所以，Spark 是一个高度统一的平台，所有的高级 API 都有相同的性质，它们之间可以很容易地相互转化。Spark 的野心就是用这一套工具统一所有数据处理的场景。

由于 Spark Streaming 将底层的细节封装起来了，所以对于开发者来说，只需要操作 DStream 就行。接下来，让我们一起学习 DStream 的结构以及它支持的转换操作。

DStream

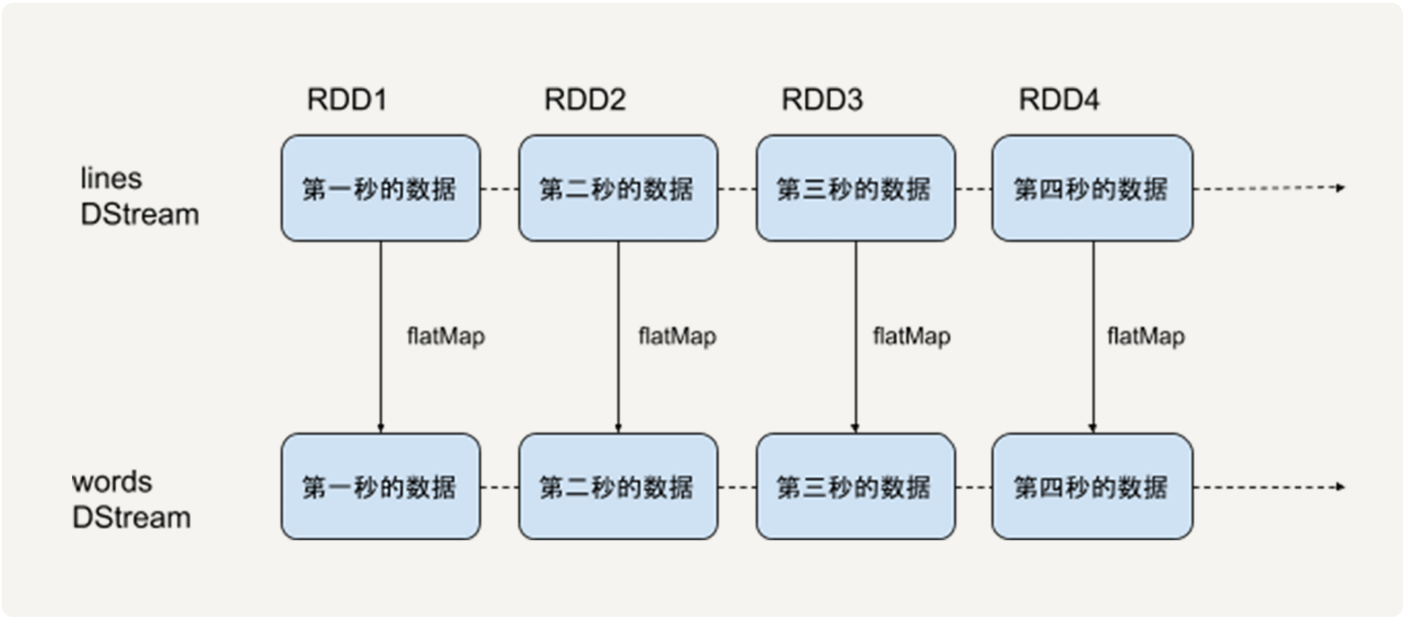
下图就是 DStream 的内部形式，即一个连续的 RDD 序列，每一个 RDD 代表一个时间窗口的输入数据流。



对 DStream 的转换操作，意味着对它包含的每一个 RDD 进行同样的转换操作。比如下边的例子。

复制代码

```
sc = SparkContext(master, appName)
ssc = StreamingContext(sc, 1)
lines = sc.socketTextStream("localhost", 9999)
words = lines.flatMap(lambda line: line.split(" "))
```



首先，我们创建了一个 lines 的 DStream，去监听来自本机 9999 端口的数据流，每一个数据代表一行文本。然后，对 lines 进行 flatMap 的转换操作，把每一个文本行拆分成词语。

本质上，对一个 DStream 进行 flatMap 操作，就是对它里边的每一个 RDD 进行 flatMap 操作，生成了一系列新的 RDD，构成了一个新的代表词语的 DStream。

正因为 DStream 和 RDD 的关系，RDD 支持的所有转换操作，DStream 都支持，比如 map、flatMap、filter、union 等。这些操作我们在前边学习 RDD 时都详细介绍过，在此不做赘述。

此外，DStream 还有一些特有操作，如滑动窗口操作，我们可以一起探讨。

滑动窗口操作

任何 Spark Streaming 的程序都要首先创建一个 **StreamingContext** 的对象，它是所有 Streaming 操作的入口。

比如，我们可以通过 StreamingContext 来创建 DStream。前边提到的例子中，lines 这个 DStream 就是由名为 sc 的 StreamingContext 创建的。

StreamingContext 中最重要的参数是批处理的**时间间隔**，即把流数据细分成数据块的粒度。

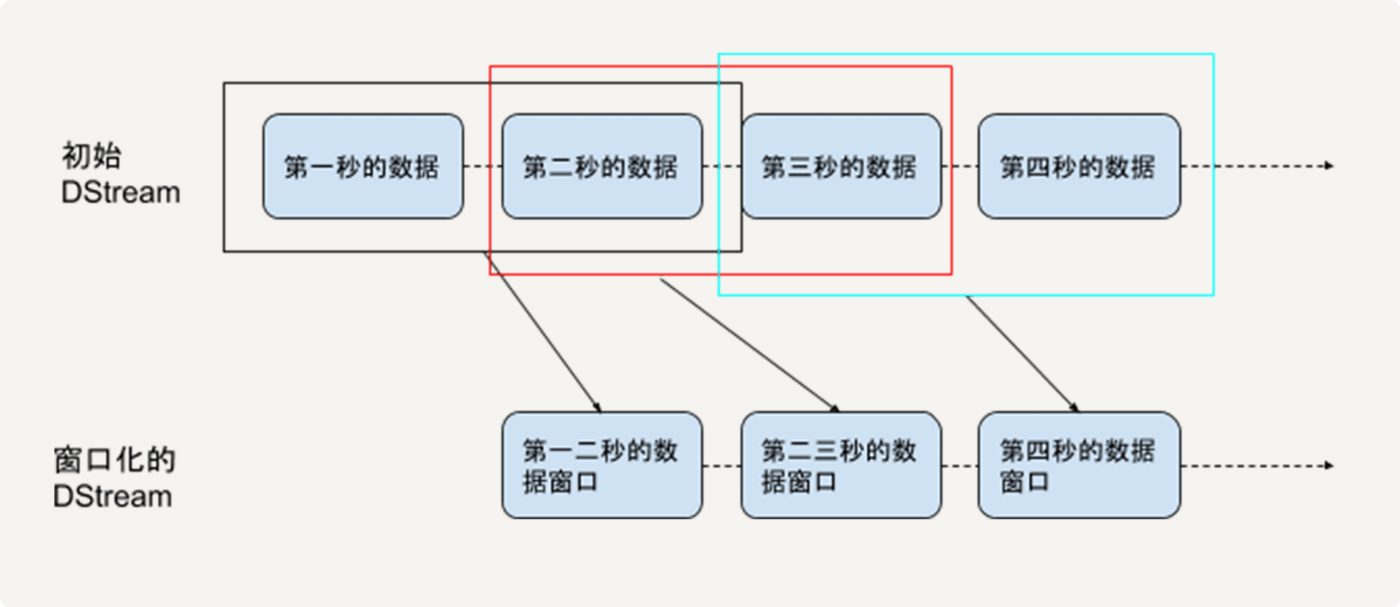
这个时间间隔决定了流处理的延迟性，所以，需要我们根据需求和资源来权衡间隔的长度。上边的例子中，我们把输入的数据流以秒为单位划分，每一秒的数据会生成一个 RDD 进行运算。

有些场景中，我们需要每隔一段时间，统计过去某个时间段内的数据。比如，对热点搜索词语进行统计，每隔 10 秒钟输出过去 60 秒内排名前十位的热点词。这是流处理的一个基本应用场景，很多流处理框架如 Apache Flink 都有原生的支持。所以，Spark 也同样支持滑动窗口操作。

从统计热点词这个例子，你可以看出滑动窗口操作有两个基本参数：

- 窗口长度（window length）：每次统计的数据的时间跨度，在例子中是 60 秒；
- 滑动间隔（sliding interval）：每次统计的时间间隔，在例子中是 10 秒。

显然，由于 Spark Streaming 流处理的最小时间单位就是 StreamingContext 的时间间隔，所以这两个参数一定是它的整数倍。



最基本的滑动窗口操作是 window，它可以返回一个新的 DStream，这个 DStream 中每个 RDD 代表一段时间窗口内的数据，如下例所示。

复制代码

```
windowed_words = words.window(60, 10)
```

windowed_words 代表的就是热词统计例子中我们所需的 DStream，即它里边每一个数据块都包含过去 60 秒内的词语，而且这样的块每 10 秒钟就会生成一个。

此外，Spark Streaming 还支持一些“进阶”窗口操作。如 countByWindow、reduceByWindow、reduceByKeyAndWindow 和 countByValueAndWindow，在此不做深入讨论。

Spark Streaming 的优缺点

讲了这么多 Spark Streaming，不管内部实现也好，支持的 API 也好，我们还并不明白它的优势是什么，相比起其他流处理框架的缺点是什么。只有明白了这些，才能帮助我们在实际工作中决定是否使用 Spark Streaming。

首先，Spark Streaming 的优点很明显，由于它的底层是基于 RDD 实现的，所以 RDD 的优良特性在它这里都有体现。

比如，数据容错性，如果 RDD 的某些分区丢失了，可以通过依赖信息重新计算恢复。

再比如运行速度，DStream 同样也能通过 persist() 方法将数据流存放在内存中。这样做的好处是遇到需要多次迭代计算的程序时，速度优势十分明显。

而且，Spark Streaming 是 Spark 生态的一部分。所以，它可以和 Spark 的核心引擎、Spark SQL、MLlib 等无缝衔接。换句话说，对实时处理出来的中间数据，我们可以立即在程序中无缝进行批处理、交互式查询等操作。这个特点大大增强了 Spark Streaming 的优势和功能，使得基于 Spark Streaming 的应用程序很容易扩展。

而 Spark Streaming 的主要缺点是实时计算延迟较高，一般在秒的级别。这是由于 Spark Streaming 不支持太小的批处理的时间间隔。

在第二章中，我们讲过准实时和实时系统，无疑 Spark Streaming 是一个准实时系统。别的流处理框架，如 Storm 的延迟性就好很多，可以做到毫秒级。

小结

Spark Streaming，作为 Spark 中的流处理组件，把连续的流数据按时间间隔划分为一个个数据块，然后对每个数据块分别进行批处理。

在内部，每个数据块就是一个 RDD，所以 Spark Streaming 有 RDD 的所有优点，处理速度快，数据容错性好，支持高度并行计算。

但是，它的实时延迟相比起别的流处理框架比较高。在实际工作中，我们还是要具体情况具体分析，选择正确的处理框架。

思考题

如果想要优化一个 Spark Streaming 程序，你会从哪些角度入手？

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 (10)

Hobbin
2019-05-24

老师，Spark团队对Spark streaming更新越来越少，Spark streaming存在使用Processing time而非 Event time，批流代码不统一等问题，而Structured streaming对这些都有一定改进。所以Structure streaming 会替代Spark streaming或者Flink，成为主流的流计算引擎吗?

> 作者回复: 你说的很对，Structured streaming是Spark流处理的未来，所以我在第17讲以及之后的实战演练才重点介绍了它。此处介绍spark streaming一来是因为它的原理很基本也很重要，二来它承接了之前介绍的RDD API。
>
> 所以我觉得Structured Streaming会替代Spark Streaming，但是很难替代Flink。Flink在流处理上的天然优势很难被Spark超越，让我们拭目以待Strucutred Streaming未来会如何发展。

共 6 条评论 >　　32

hua168
2019-05-24

批处理可以选择spark；流处理：spark stream，storm，Flink；还有现在大统一的beam
请问：这些技术都要学一遍吗？精力放在哪个技术上？
如果我是初学者，我能直接学beam其它都不学吗？

共 2 条评论 >　　17

邱从贤※klion26
2019-05-26

上一条留言没有说完。
spark streaming 需要设置 batch time 是多少，这决定时效性，以及调度的 overhead，另外要看自己需要的吞吐多大，并发是不是有特殊需求。

spark streaming 有几个点不太喜欢，修改业务逻辑后，需要删除 checkpoint 才行，这会导致从头计算；慢节点没法解决，当一个 batch 里面有一个节点很难的时候，整个 batch 都无法完成。
一个反常识的点：实时 etl 同样吞吐下，flink 比 spark streaming 更节省资源。
另外官方已经放弃 spark streaming，转向structured streaming，但是从邮件列表看又没有 commiter 在管，导致 pr 没人 review，或许这和 spark 整体的重心或者方向有关吧

14

lwenbin
2019-05-24

没用过spark streaming, 用storm比较多。
觉得流处理关键在于要在窗口内尽快地把到来的数据处理完，不要造成数据堆积，内存溢出。其中牵涉到了如何高效地接受数据，如何并行尽快地处理数据。
觉得优化可以从：接受输入，处理算法，处理单元数量，GC调优等方面入手吧。
有个问题，对于RDD如果transform链很长，感觉是否会对性能造成一定影响，特别是流式或者图形计算？老师能否解答一下。
谢谢！

作者回复: 如果transform链很长，在流处理中确实会影响处理的实时性，你的想法是对的。如果只有一条很长的链，在Spark的框架中，也很难去优化。

共 2 条评论>

9

Fiery
2020-03-11

既然DStream底层还是RDD，那我认为针对RDD的一些优化策略对DStream也有效。比如平衡RDD分区减少数据倾斜，在tranformation链中优先使用filter/select/first减少数据量，尽量串接窄依赖函数方便实现节点间并行计算和单节点链式计算优化，join时优化分区或使用broadcast减少stage间shuffle。

另外专门针对流数据的处理，个人经验上主要是要根据业务需求微调window length和sliding interval以达到吞吐量和延时之间的一个最优平衡，时间窗口越大，一个RDD可以一次批处理的数据就越多，Spark的优势就可以发挥出来，吞吐量就上去了。而滑动间隔越大，windowed DStream在固定时间内的RDD就越少，系统的任务队列里同时需要处理的计算当然就越少，不过这两个调整都会加大数据更新延迟和牺牲数据实时性，所以说要根据业务真实需求谨慎调整。

不过个人理解RDD里面用来避免重复计算的cache和persist无法用来减少窗口滑动产生的重复

计算，因为窗口每滑动一次，都产生一个新的RDD，而persist只针对其中某个RDD进行缓存，在RDD这种low level api里面，应该是无法知道下个窗口中的RDD和现在的RDD到底有多少数据是重叠的。对于这点理解是否正确望老师解答！

7

Ming
2019-05-24

优化的定义很广，不知道在这个领域大家提到这个词主要指的是什么？望解答

不过，对具体实现细节不了解的情况下有几个猜测：
我会改变时间粒度，来减少RDD本身带来的开销，上文的例子里时间粒度如果设置成10秒应该逻辑上也是可行的。
另外，我大概会考虑多使用persist来减少因为窗口滑动产生的重复计算。

共 1 条评论 >

3

不记年
2020-03-30

在满足需求的情况下尽可能的使用更宽的窗口长度，减少rdd的处理链

1

王翔宇
2019-07-12

sc和ssc的区别是什么？我理解ssc才是那个streamingContext吧，如果是这样，那么又出现错误了。

渡码
2019-05-28

稳定性：对于7*24小时的流式任务至关重要
低延迟高吞吐量

1

邱从贤※klion26
2019-05-26

spark streaming 批次的大小设置多少合适，从官方宣传来看已经被 structure steeaming 替代。