7.5 转换操作

7.5.1 DStream无状态转换操作

7.5.2 DStream有状态转换操作

7.5.1 DStream无状态转换操作



- map(func)：对源DStream的每个元素，采用func函数进行转换，得到一个新的Dstream
- flatMap(func)：与map相似，但是每个输入项可用被映射为0个或者多个输出项
- filter(func)：返回一个新的DStream，仅包含源DStream中满足函数func的项
- repartition(numPartitions)：通过创建更多或者更少的分区改变DStream的并行程度
- reduce(func)：利用函数func聚集源DStream中每个RDD的元素，返回一个包含单元素RDDs的新DStream
- count()：统计源DStream中每个RDD的元素数量
- union(otherStream)：返回一个新的DStream，包含源DStream和其他DStream的元素

7.5.1 DStream无状态转换操作



- countByValue()：应用于元素类型为K的DStream上，返回一个（K, V）键值对类型的新DStream，每个键的值是在原DStream的每个RDD中的出现次数
- reduceByKey(func, [numTasks])：当在一个由(K,V)键值对组成的DStream上执行该操作时，返回一个新的由(K,V)键值对组成的DStream，每一个key的值均由给定的recuce函数（func）聚集起来
- join(otherStream, [numTasks])：当应用于两个DStream（一个包含（K,V）键值对,一个包含(K,W)键值对），返回一个包含(K, (V, W))键值对的新Dstream
- cogroup(otherStream, [numTasks])：当应用于两个DStream（一个包含（K,V）键值对,一个包含(K,W)键值对），返回一个包含(K, Seq[V], Seq[W])的元组
- transform(func)：通过对源DStream的每个RDD应用RDD-to-RDD函数，创建一个新的DStream。支持在新的DStream中做任何RDD操作

7.5.1 DStream无状态转换操作



无状态转换操作实例：
之前“套接字流”部分介绍的词频统计，就是采用无状态转换，每次统计，都是只统计当前批次到达的单词的词频，和之前批次无关，不会进行累计

7.5.2 DStream有状态转换操作

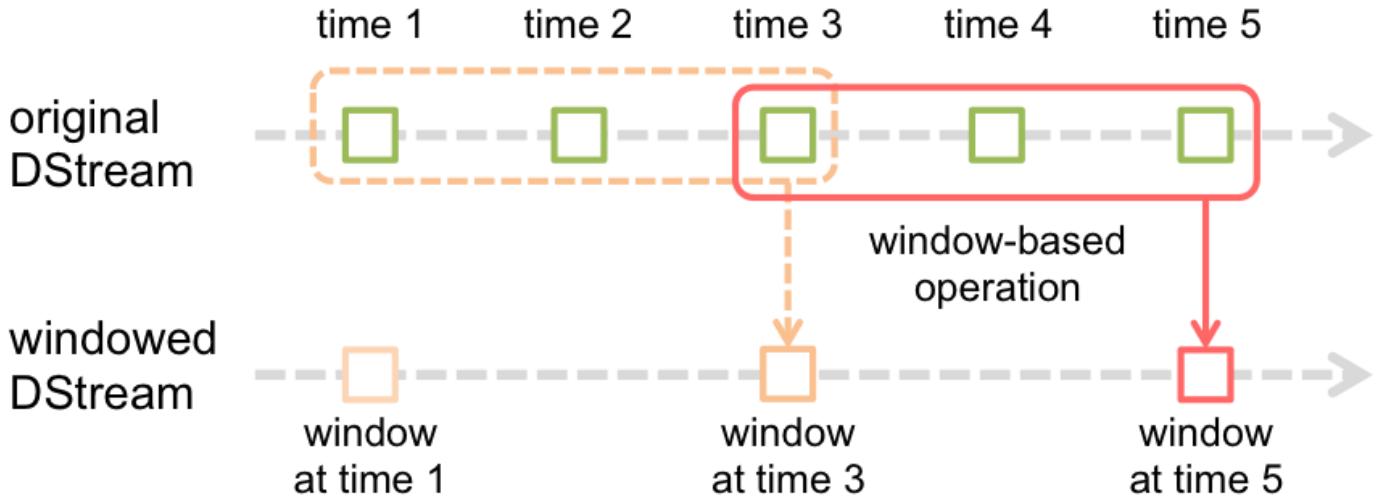
1. 滑动窗口转换操作
2. updateStateByKey操作

7.5.2 DStream有状态转换操作



1. 滑动窗口转换操作

- 事先设定一个滑动窗口的长度（也就是窗口的持续时间）
- 设定滑动窗口的时间间隔（每隔多长时间执行一次计算），让窗口按照指定时间间隔在源DStream上滑动
- 每次窗口停放的位置上，都会有一部分Dstream（或者一部分RDD）被框入窗口内，形成一个小段的Dstream
- 可以启动对这个小段DStream的计算



7.5.2 DStream有状态转换操作



一些窗口转换操作的含义：

•window(windowLength, slideInterval) 基于源DStream产生的窗口化的批数据，计算得到一个新的Dstream

•countByWindow(windowLength, slideInterval) 返回流中元素的一个滑动窗口数

•reduceByWindow(func, windowLength, slideInterval) 返回一个单元素流。利用函数func聚集滑动时间间隔的流的元素创建这个单元素流。函数func必须满足结合律，从而可以支持并行计算

•countByValueAndWindow(windowLength, slideInterval, [numTasks]) 当应用到一个(K,V)键值对组成的DStream上，返回一个由(K,V)键值对组成的新的DStream。每个key的值都是它们在滑动窗口中出现的频率

7.5.2 DStream有状态转换操作



一些窗口转换操作的含义：

•reduceByKeyAndWindow(func, windowLength, slideInterval, [numTasks]) 应用到一个(K,V)键值对组成的DStream上时，会返回一个由(K,V)键值对组成的新的DStream。每一个key的值均由给定的reduce函数(func函数)进行聚合计算。注意：在默认情况下，这个算子利用了Spark默认的并发任务数去分组。可以通过numTasks参数的设置来指定不同的任务数

•reduceByKeyAndWindow(func, invFunc, windowLength, slideInterval, [numTasks]) 更加高效的reduceByKeyAndWindow，每个窗口的reduce值，是基于先前窗口的reduce值进行增量计算得到的；它会对进入滑动窗口的新数据进行reduce操作，并对离开窗口的老数据进行“逆向reduce”操作。但是，只能用于“可逆reduce函数”，即那些reduce函数都有一个对应的“逆向reduce函数”（以InvFunc参数传入）

7.5.2 DStream有状态转换操作



WindowedNetworkWordCount.py

```
# coding:utf8
from __future__ import print_function
import sys
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

if __name__ == "__main__":
    sc = SparkContext(appName="PythonStreamingWindowedNetworkWordCount")
    ssc = StreamingContext(sc, 10)
    ssc.checkpoint("file:///usr/local/spark/mycode/streaming/socket/checkpoint")
    lines = ssc.socketTextStream("hadoop01",9999)
    counts = lines.flatMap(lambda line: line.split(" "))₩
                  .map(lambda word: (word, 1))₩
                  . reduceByKeyAndWindow(lambda x, y: x + y, lambda x, y: x - y, 30, 10)
    counts.pprint()
    ssc.start()
    ssc.awaitTermination()
```

7.5.2 DStream有状态转换操作



```
reduceByKeyAndWindow(lambda x, y: x + y, lambda x, y: x - y, 30, 10)
```

这行代码就使用了滑动窗口转换操作reduceByKeyAndWindow()，为了便于理解，表7-3列出了该函数的各个参数的取值及含义。

参数	值	含义
func	lambda x, y: x + y	词频加操作
invFunc	lambda x, y: x -y	词频减操作
windowLength	30	滑动窗口大小为30秒
slideInterval	10	每隔10秒滑动一次

7.5.2 DStream有状态转换操作



```
reduceByKeyAndWindow(lambda x, y: x + y, lambda x, y: x - y, 30, 10)
```

实现增量计算

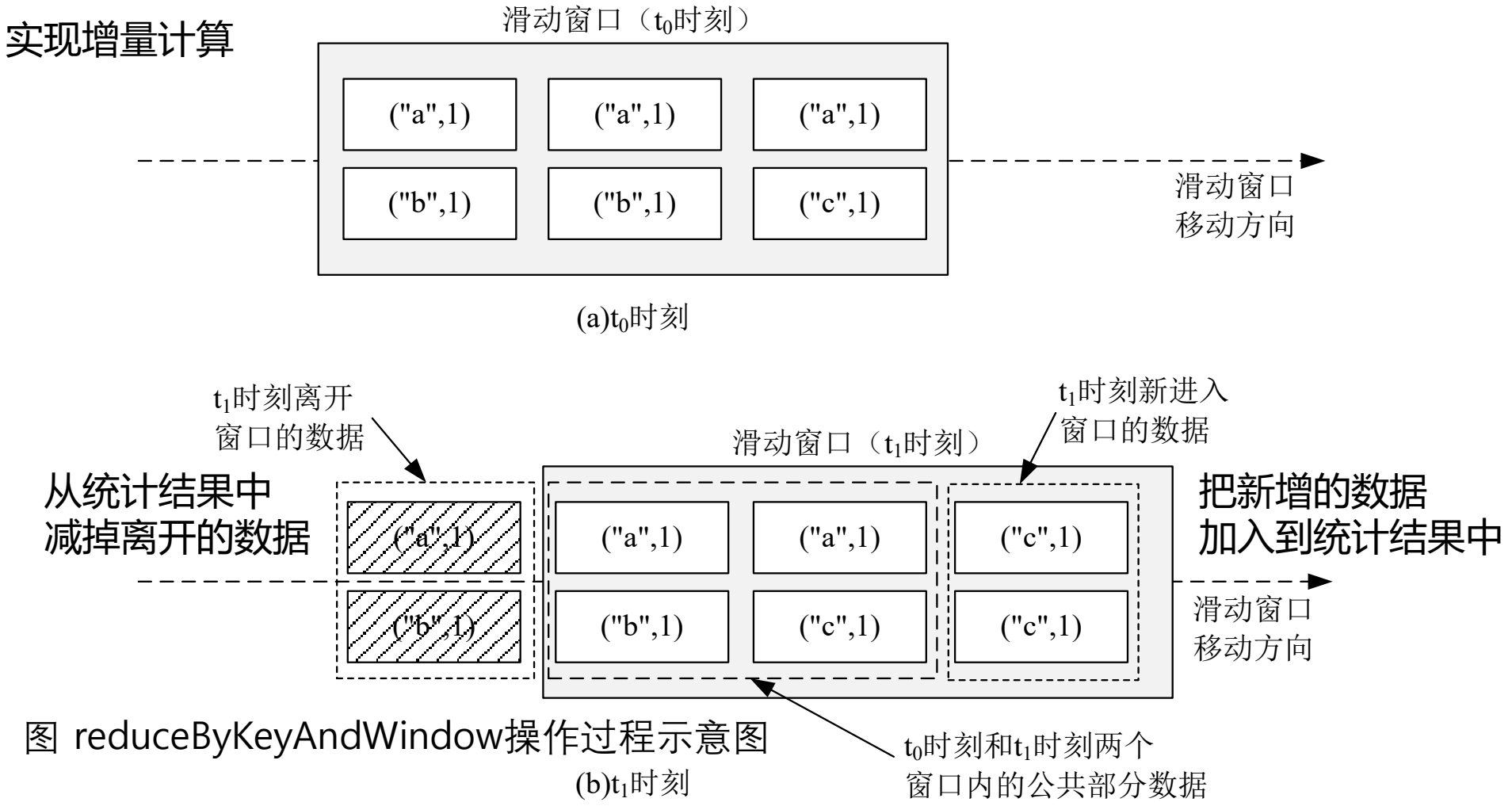


图 reduceByKeyAndWindow操作过程示意图

7.5.2 DStream有状态转换操作



为了测试程序的运行效果，首先新建一个终端（记作“数据源终端”），执行如下命令运行nc程序：

```
$ cd /usr/local/spark/mycode/streaming/socket/
$ nc -lk 9999
```

再新建一个终端（记作“流计算终端”），运行客户端程序WindowedNetworkWordCount.py，命令如下：

```
$ cd /usr/local/spark/mycode/streaming/socket/
$ /usr/local/spark/bin/spark-submit  WindowedNetworkWordCount.py
```

在数据源终端内，用键盘连续敲入10个“hadoop”，每个hadoop单独占一行（即输入一个hadoop就回车），再用键盘连续敲入10个“spark”，每个spark单独占一行。这时，可以查看流计算终端内显示的词频动态统计结果，可以看到，随着时间的流逝，词频统计结果会发生动态变化。

7.5.2 DStream有状态转换操作



2. updateStateByKey操作

需要在跨批次之间维护状态时，就必须使用updateStateByKey操作

词频统计实例：

对于有状态转换操作而言，本批次的词频统计，会在之前批次的词频统计结果的基础上进行不断累加，所以，最终统计得到的词频，是所有批次的单词的总的词频统计结果

7.5.2 DStream有状态转换操作



NetworkWordCountStateful.py

```
#coding:utf8

from __future__ import print_function
import sys
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

if __name__ == "__main__":
    sc = SparkContext(appName="PythonStreamingStatefulNetworkWordCount")
    ssc = StreamingContext(sc, 1)
    ssc.checkpoint("file:///usr/local/spark/mycode/streaming/stateful/")
    # RDD with initial state (key, value) pairs
    initialStateRDD = sc.parallelize([(u'hello', 1), (u'world', 1)])

    def updateFunc(new_values, last_sum):
        return sum(new_values) + (last_sum or 0)

    lines = ssc.socketTextStream("hadoop01", 9999)
    running_counts = lines.flatMap(lambda line: line.split(" "))₩
                          .map(lambda word: (word, 1))₩
                          .updateStateByKey(updateFunc, initialRDD=initialStateRDD)
    running_counts.pprint()
    ssc.start()
    ssc.awaitTermination()
```

7.5.2 DStream有状态转换操作



新建一个终端（记作"数据源终端"），执行如下命令启动nc程序：

```
$ nc  -lk  9999
```

新建一个Linux终端（记作"流计算终端"），执行如下命令提交运行程序：

```
$ cd /usr/local/spark/mycode/streaming/stateful
$ /usr/local/spark/bin/spark-submit NetworkWordCountStateful.py
```

在数据源终端内手动输入一些单词并回车，再切换到流计算终端，可以看到已经输出了类似如下的词频统计信息：

```
-------------------------------------------
Time: 2023-09-17 15:38:07
-------------------------------------------
('hadoop', 1)
('world', 1)
('hello', 1)
('spark', 1)
```