```
TwitterUtils.createStream(...)
    .filter(_.getText.contains("Spark"))
    .countByWindow(Seconds(5))
```

Spark Streaming

databricks™

流式计算

- Streaming
- Batch 分析 vs 实时或近实时分析
  - 实时或近实时分析变得越来越关键
  - 自动驾驶汽车，电网传感器
  - 社会网络上的热门话题和主题标签
- 分析来自无限制流的数据的活动，即数据流分析
  - 可追溯到 1990 年代在斯坦福，加州理工学院和剑桥等地进行的复杂事件处理的基础研究

Spark Streaming

- 基于 Spark
- 高级库，旨在处理大多数云上的流数据
- 原理
  - 利用 Spark Core 的快速调度功能，按窗口获取流数据，将其转换为一种特殊的 RDD：Dstream
  - Dstream 进入流处理引擎，该引擎可以遵循 MapReduce 或任何 DAG 模型，完成计算
- 优点
  - 基于 RDD 的设计使为批处理分析编写的同一套应用程序代码可以在流分析中使用

Spark Streaming

- Scalable
- High-throughput
- Fault-tolerant

TCP socket

Kafka

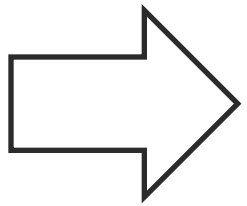
Flume

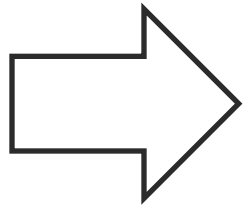
HDFS

S3

Kinesis

Twitter





HDFS / S3

Cassandra

HBase

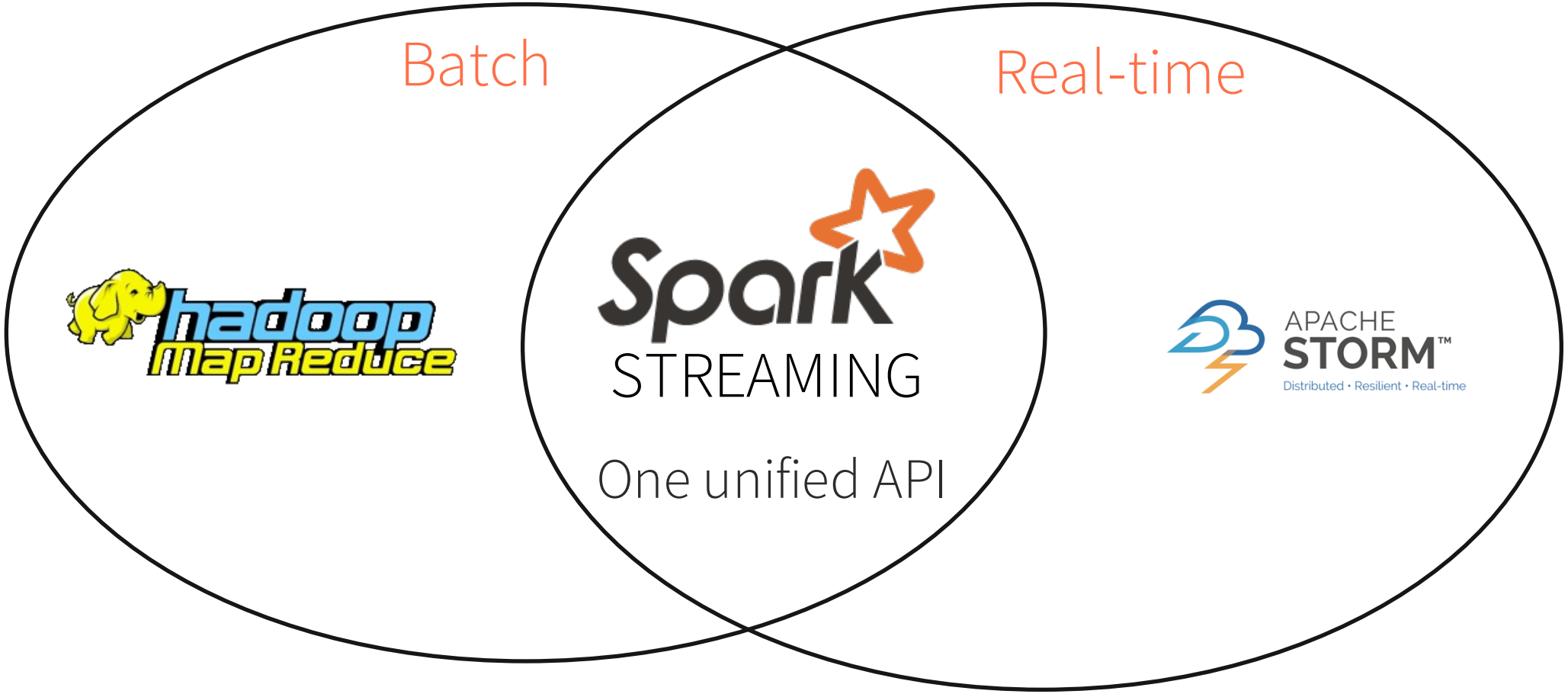
Dashboards

Databases

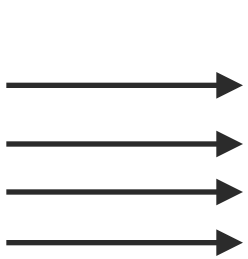
Complex algorithms can be expressed using:

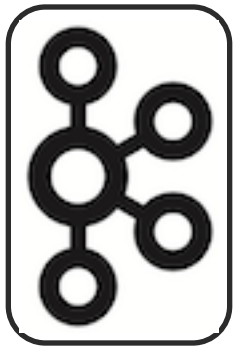
- Spark transformations: `map()`, `reduce()`, `join()`…
- MLlib + GraphX
- SQL

Batch and Real-time



Use Cases





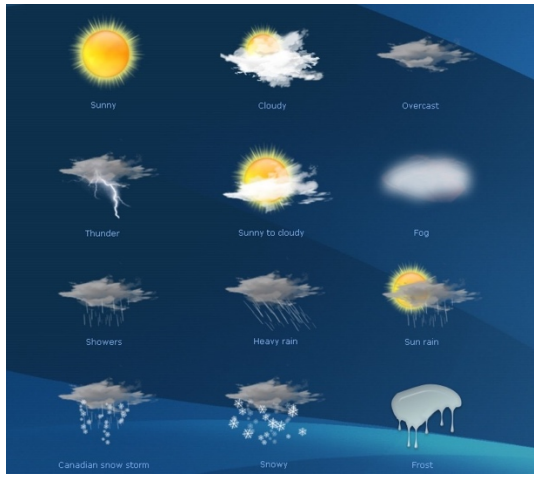
Page views

Kafka for buffering

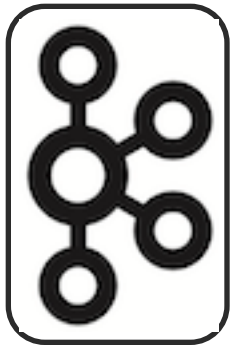
Spark for processing

Use Cases

Smart meter readings



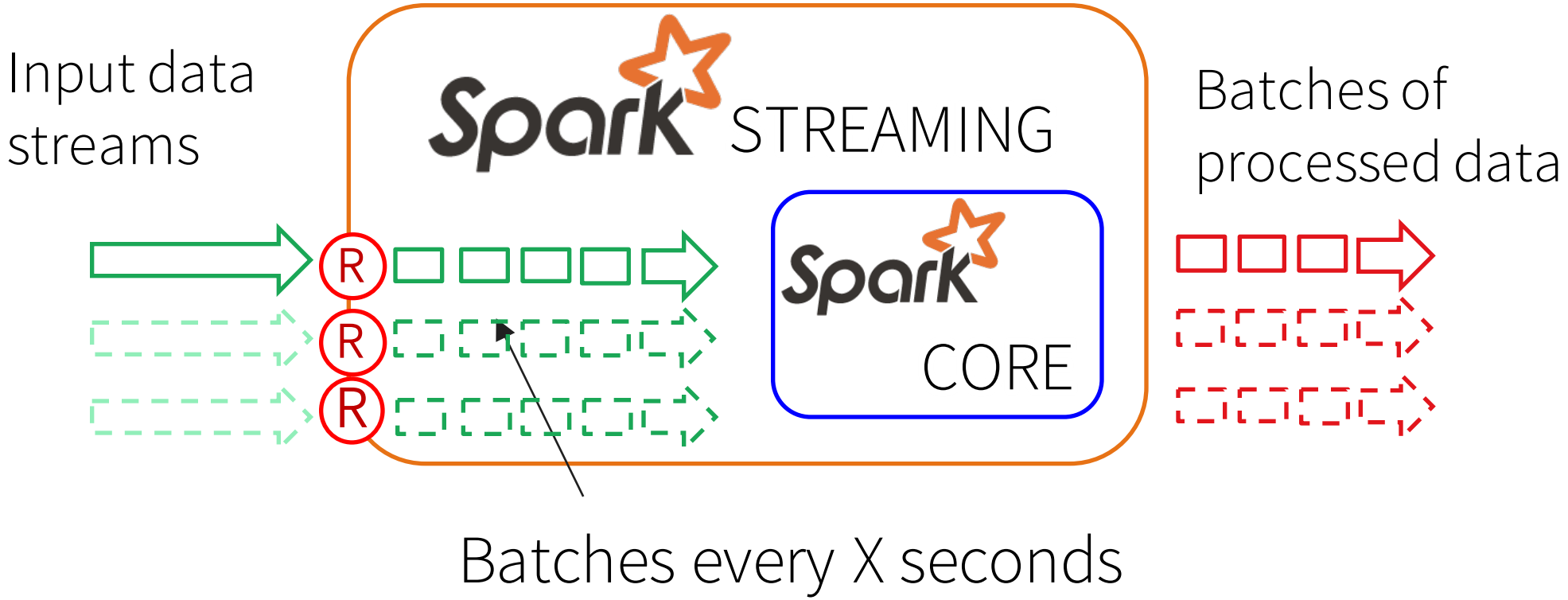
Live weather data



Join 2 live data sources

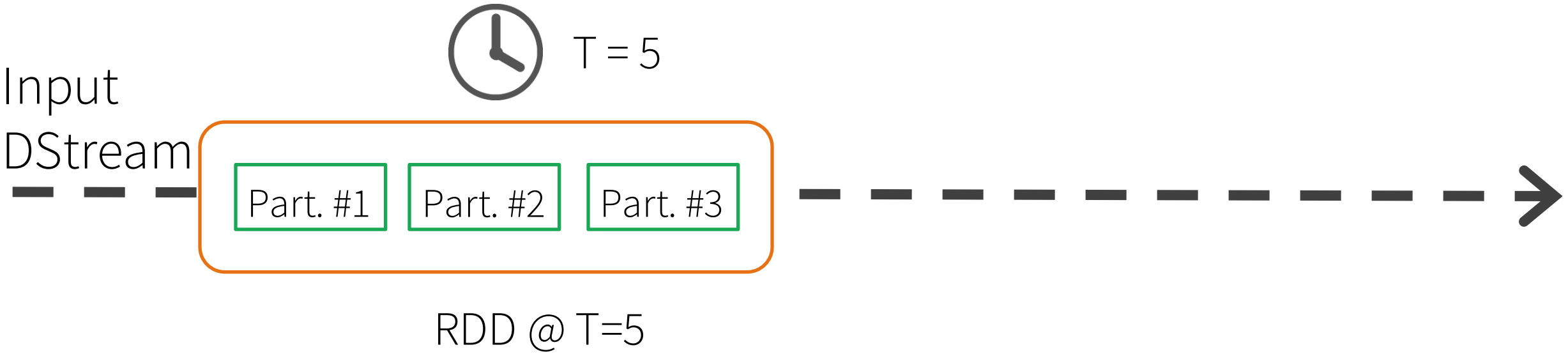
databricks

Data Model


Input data streams
Spark STREAMING
Spark CORE
R
R
R
Batches of processed data
Batches every X seconds

DStream (Discretized Stream)

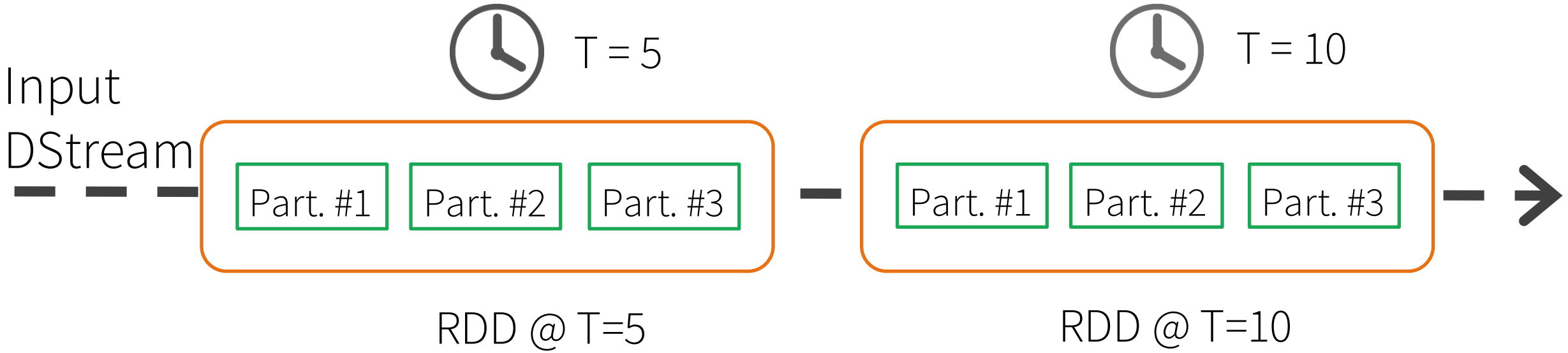
Batch interval = 5 seconds



One RDD is created every 5 seconds

DStream (Discretized Stream)

Batch interval = 5 seconds



One RDD is created every 5 seconds

Streaming 支持的 Transformation 操作

- Map
- Filter
- Repartition
- Union
- Reduce
- Join
- Transform
  - 通过对源 DStream 的每个 RDD 应用 RDD-to-RDD 函数来返回新的 DStream。这可用于在 DStream 上执行任意的 RDD 操作

部署

- 数据源
  - 可以部署为从 HDFS，Flume，Kafka，Twitter 和 ZeroMQ 源读取数据
- 可以应用于大型集群或单个引擎
  - 在大型集群的生产模式下，可使用 ZooKeeper 和 HDFS 来实现高可用性

部署

- 首先部署集群管理器，对其进行识别，分配资源
- 打包应用程序，将程序编译为 JAR
  - 如果程序使用高级资源（例如，Kafka，Flume，Twitter），程序须链接到它们
  - 比如使用 TwitterUtils 的程序必须包含 spark-streaming-twitter_2.10 及其依赖项
- 为程序配置足够内存以容纳接收到的数据。例如，如果要执行 10 分钟窗口操作，则必须至少将最后 10 分钟数据保留在内存中

部署：容错

- 部署检查点
  - 将 Hadoop API 兼容的容错存储中的目录（例如 HDFS，S3 等）配置为检查点目录。检查点信息可用于故障恢复。
- 配置重启
  - 配置程序驱动程序的自动重启，程序须监视驱动程序进程，在驱动程序失败时重新启动驱动程序
- 预写日志
  - 将接收的所有数据写入检查点目录中的预写日志，可以防止驱动程序恢复时丢失数据，从而确保零数据丢失。但会以单个接收器的接收吞吐量为代价。可通过并行运行更多接收器以提高总吞吐量来纠正此问题
  - 启用预写日志后，可以禁用 Spark 收到的数据的复制，因为该日志已经存储在复制的存储系统中

示例

以下代码用于计算滑动窗口上的推文：

```
TwitterUtils.createStream(…)
    .filter(_.getText.contains("Spark"))
    .countByWindow(Seconds(5))
```

代码讲解：3.5-Spark/code/stream/pyspark-streaming-twitter-master

参考：https://aiyanbo.gitbooks.io/spark-programming-guide-zh-cn/content/spark-streaming/index.html

Kafka

- 包含发布-订阅（Pub-Sub）消息传递和数据流处理的开源消息系统
- 在服务器群集上运行，高度可扩展性
- 其中的流是记录流，记录被分为多个主题
- 每个记录有一个键、一个值和一个时间戳
- 流可以很简单
  - 一个单流客户端使用一个或多个主题的事件
- 也可以很复杂
  - 基于组织成图的生产者和消费者的集合，称为拓扑

Google Dataflow and Apache Beam

- Google Cloud Dataflow 系统的开源版本
- 各种最新数据流分析解决方案的常用入口
- 目标是是将批处理和流处理进行统一
- 各种 Trigger（触发器）

Beam watermark（水印）

- 由 Google Dataflow 引入的概念
- 基于事件时间。当系统估计它已在给定窗口中看到所有数据时就用它来发送结果
- 基于指定水印的不同方法，你可以使用多种方法来定义触发器
- 建议参考 Google Dataflow 文档

Beam vs Spark Streaming

- 原生
  - Spark 是一个批处理系统，流模式是在此基础上附加的
  - Beam 是为了数据流处理从头开始设计的，有批处理能力
- 窗口定义
  - Spark 窗口基于 DStream 中的 RDD，不如 Beam 窗口灵活
- 乱序事件处理
  - 乱序情况下，事件发生时间和处理时间并不相同，这在分布式处理时很常见
  - Beam 能够对此处理。它对事件时间窗口、触发器和水印的介绍是它的主要贡献。因此它可以在乱序情况下仍然及时地生成近似结果