O'REILLY®
Strata
CONFERENCE
Making Data Work

Feb. 26 – 28, 2013
SANTA CLARA, CA

strataconf.com
#strataconf

Spark Streaming

Large-scale near-real-time stream processing

Tathagata Das (TD)
UC Berkeley

amplab
UC BERKELEY

Requirements

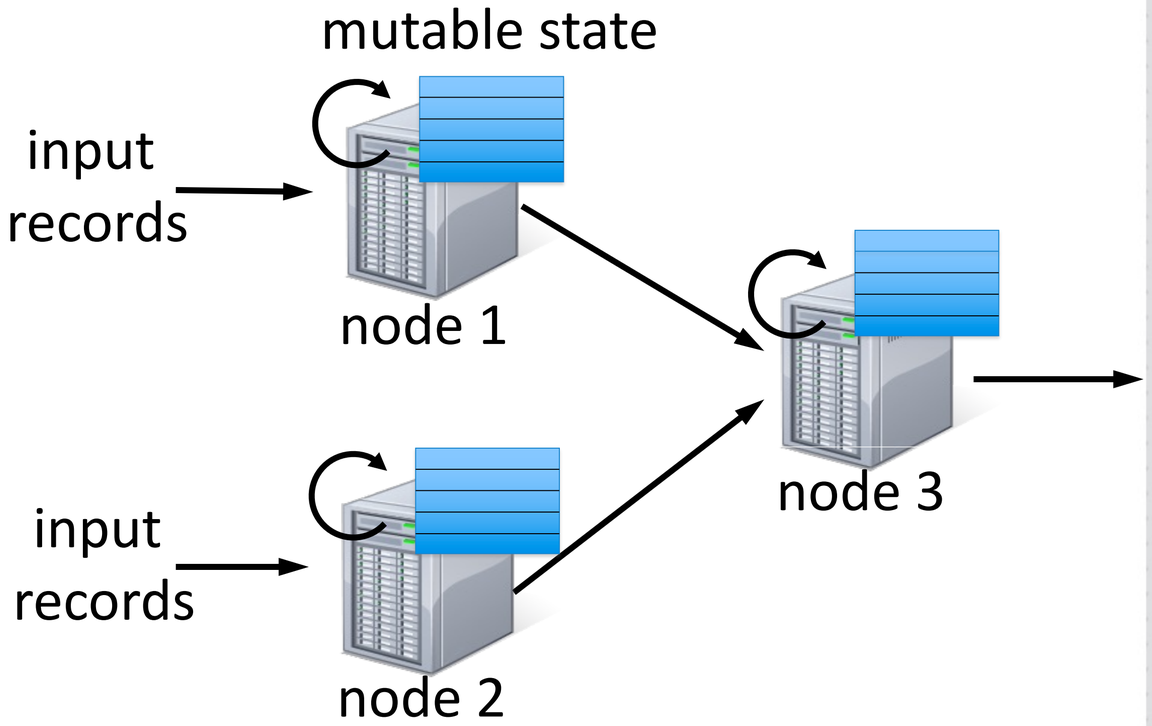
- **Scalable** to large clusters
- **Second-scale** latencies
- **Simple** programming model
- **Integrated** with batch & interactive processing

Stateful Stream Processing

- Traditional streaming systems have a event-driven **record-at-a-time** processing model
  - Each node has mutable state
  - For each record, update state & send new records

- State is lost if node dies!

- Making stateful stream processing be fault-tolerant is challenging



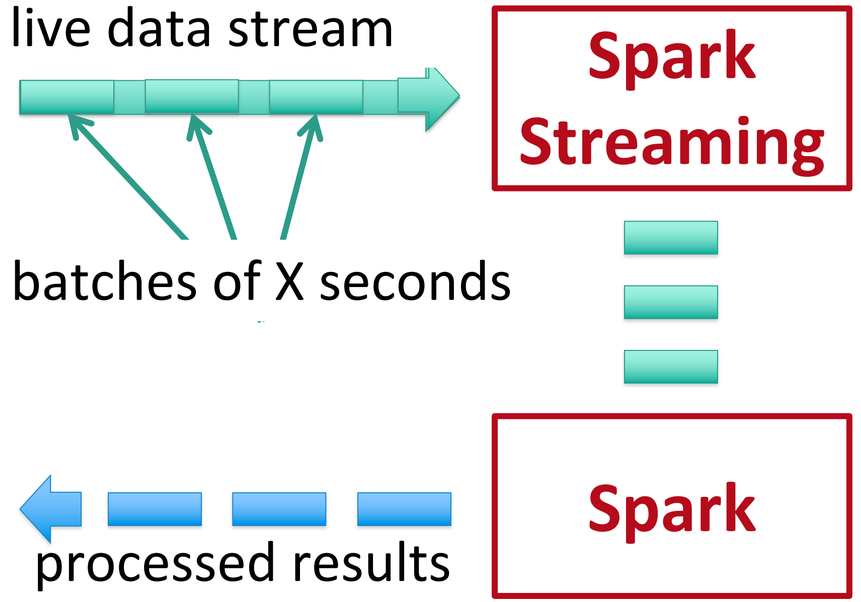
Requirements

- **Scalable** to large clusters
- **Second-scale** latencies
- **Simple** programming model
- **Integrated** with batch & interactive processing
- **Efficient fault-tolerance** in stateful computations

Discretized Stream Processing

Run a streaming computation as a **series of very small, deterministic batch jobs**

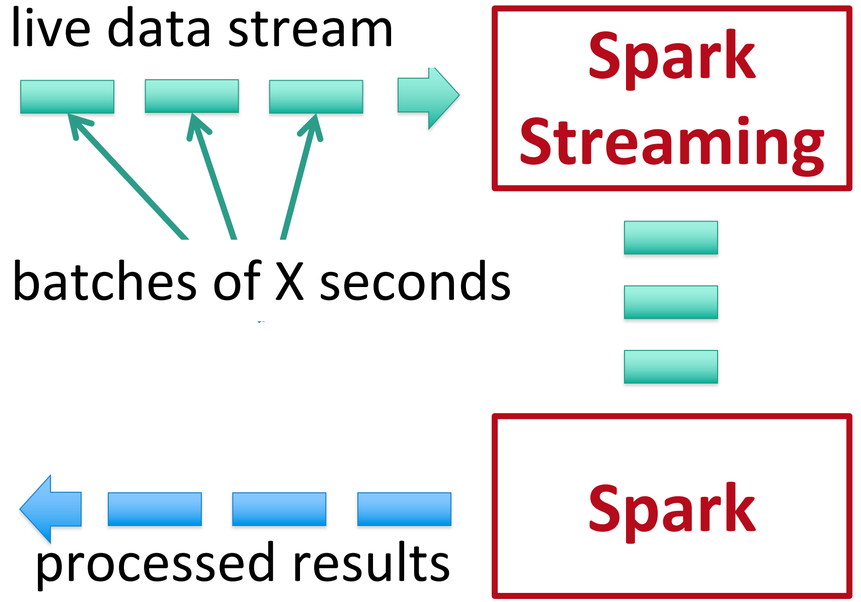
- Chop up the live stream into batches of X seconds
- Spark treats each batch of data as RDDs and processes them using RDD operations
- Finally, the processed results of the RDD operations are returned in batches



Discretized Stream Processing

Run a streaming computation as a **series of very small, deterministic batch jobs**

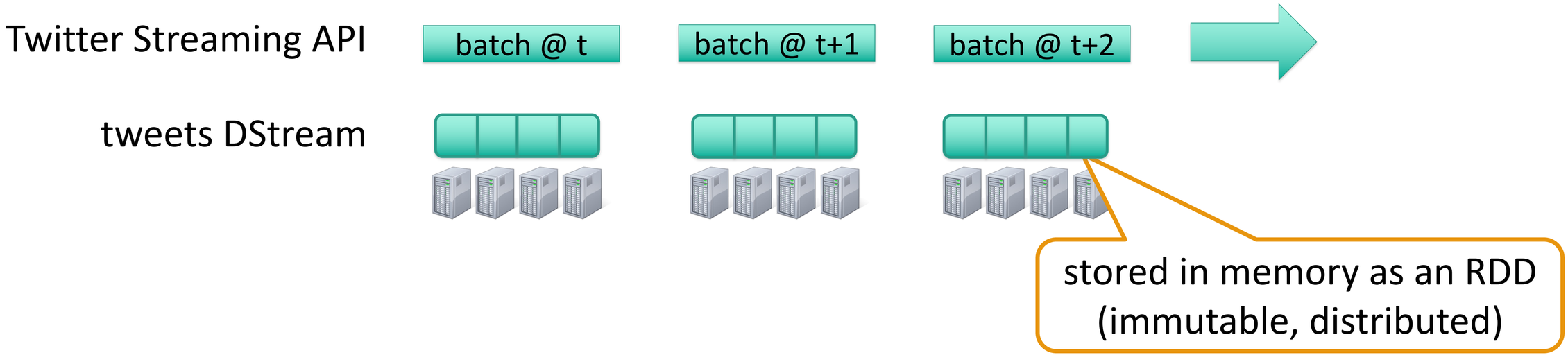
- Batch sizes as low as ½ second, latency ~ 1 second
- Potential for combining batch processing and streaming processing in the same system



Example 1 – Get hashtags from Twitter

```
val tweets = ssc.twitterStream(<Twitter username>, <Twitter password>)
```

DStream: a sequence of RDD representing a stream of data

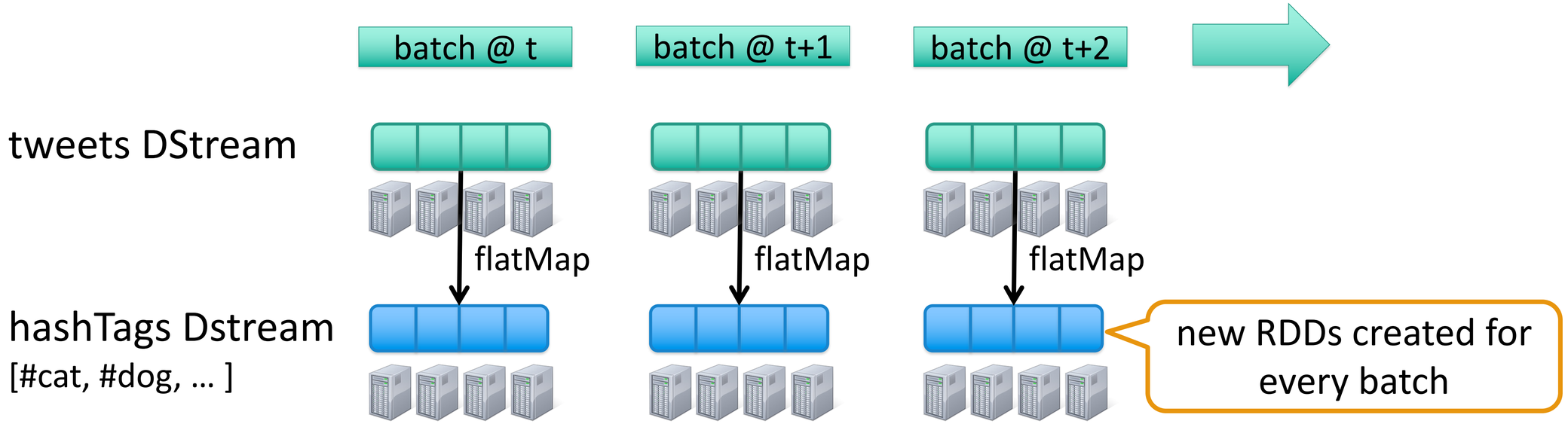


Example 1 – Get hashtags from Twitter

```
val tweets = ssc.twitterStream(<Twitter username>, <Twitter password>)
val hashTags = tweets.flatMap (status => getTags(status))
```

new DStream

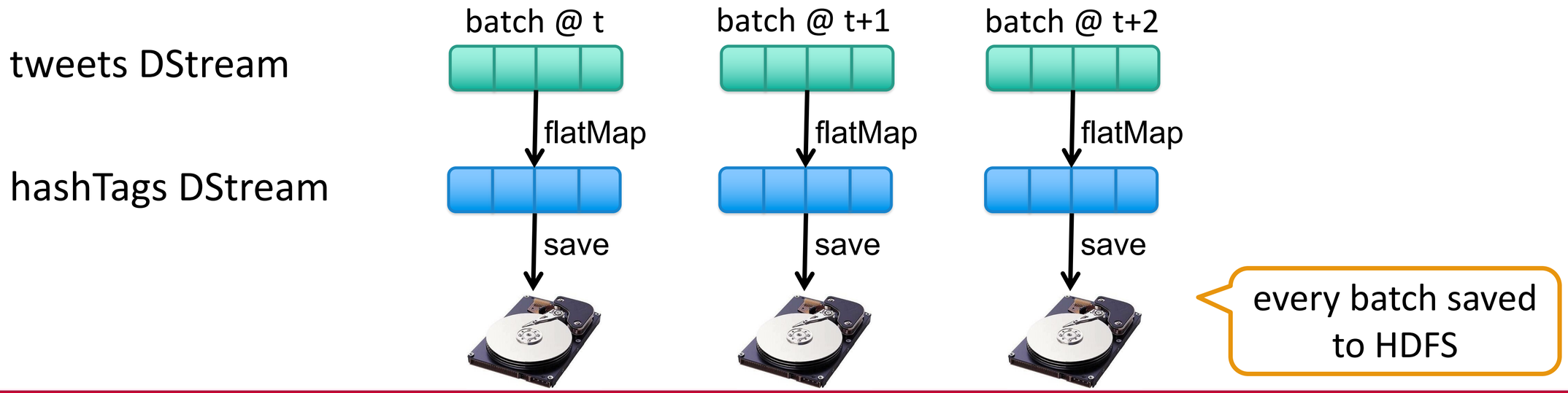
transformation: modify data in one Dstream to create another DStream



Example 1 – Get hashtags from Twitter

```
val tweets = ssc.twitterStream(<Twitter username>, <Twitter password>)
val hashTags = tweets.flatMap (status => getTags(status))
hashTags.saveAsHadoopFiles("hdfs://...")
```

output operation: to push data to external storage



Java Example

Scala

```
val tweets = ssc.twitterStream(<Twitter username>, <Twitter password>)
val hashTags = tweets.flatMap (status => getTags(status))
hashTags.saveAsHadoopFiles("hdfs://...")
```

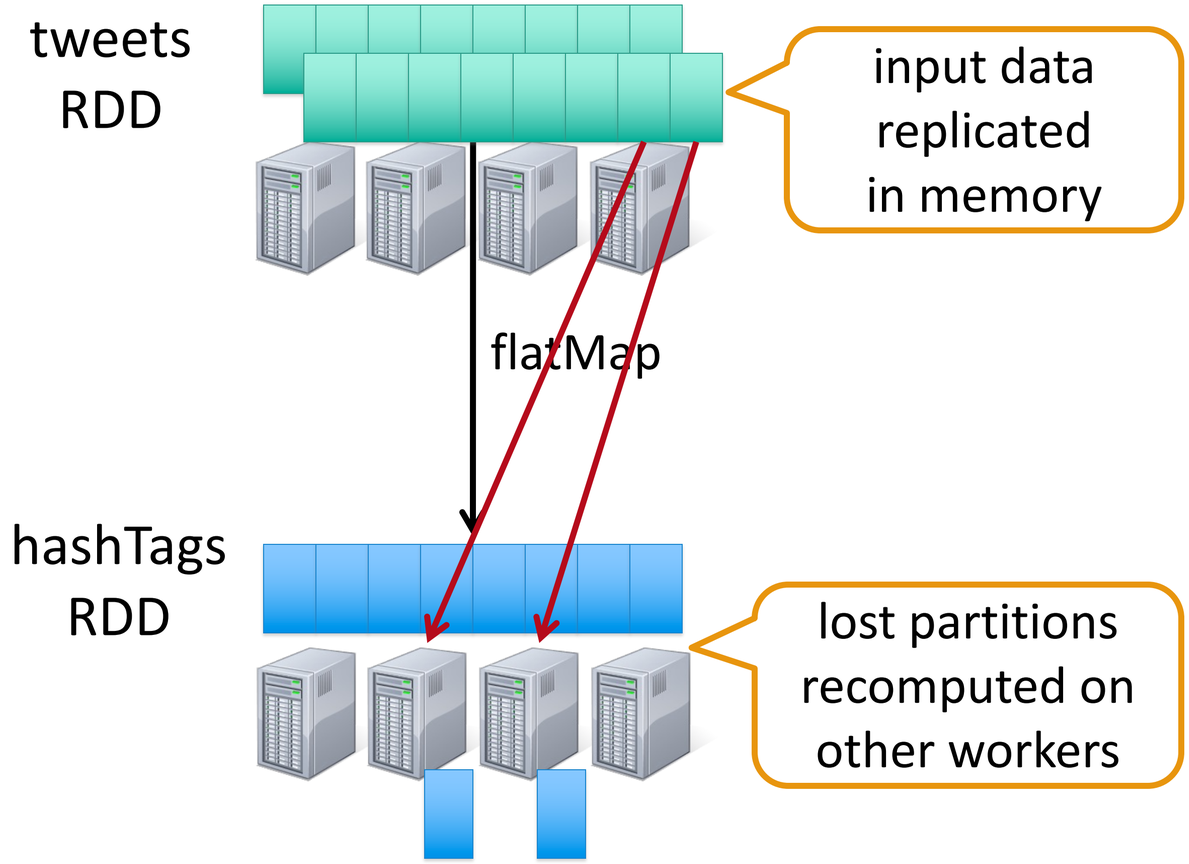
Java

```
JavaDStream<Status> tweets = ssc.twitterStream(<Twitter username>, <Twitter password>)
JavaDstream<String> hashTags = tweets.flatMap(new Function<...> {  })
hashTags.saveAsHadoopFiles("hdfs://...")
```

Function object to define the transformation

Fault-tolerance

- RDDs are remember the sequence of operations that created it from the original fault-tolerant input data
- Batches of input data are replicated in memory of multiple worker nodes, therefore fault-tolerant
- Data lost due to worker failure, can be recomputed from input data



Key concepts

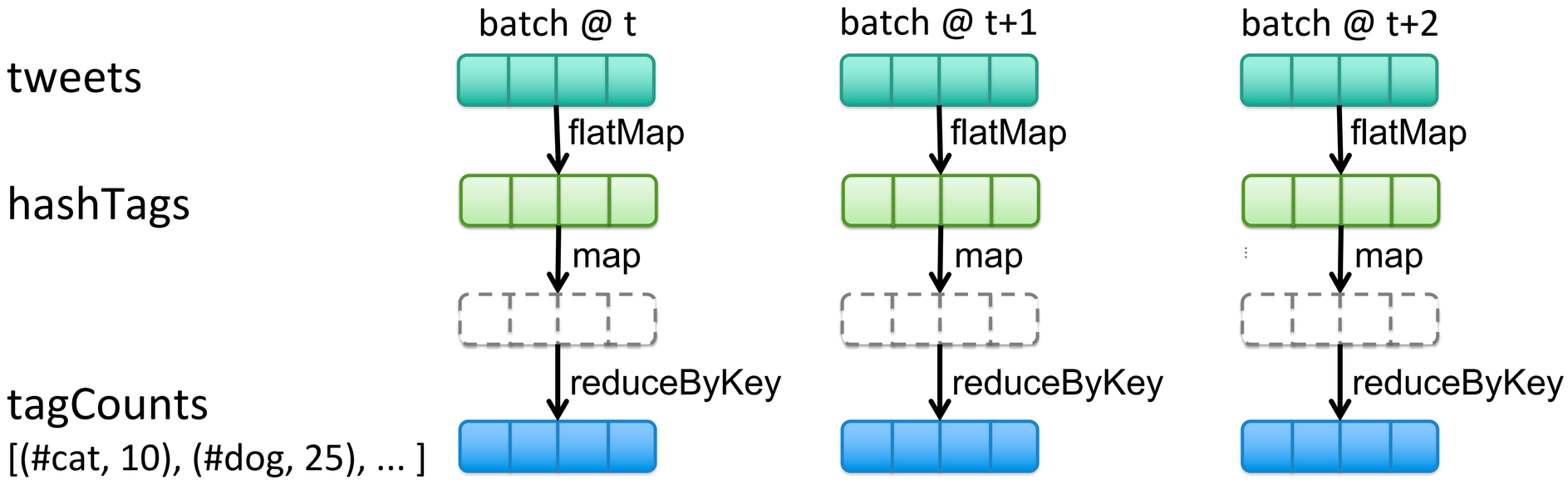
- **DStream** – sequence of RDDs representing a stream of data
  - Twitter, HDFS, Kafka, Flume, ZeroMQ, Akka Actor, TCP sockets

- **Transformations** – modify data from on DStream to another
  - Standard RDD operations – map, countByValue, reduce, join, …
  - Stateful operations – window, countByValueAndWindow, …

- **Output Operations – send data to external entity**
  - saveAsHadoopFiles – saves to HDFS
  - foreach – do anything with each batch of results

Example 2 – Count the hashtags

```
val tweets = ssc.twitterStream(<Twitter username>, <Twitter password>)
val hashTags = tweets.flatMap (status => getTags(status))
val tagCounts = hashTags.countByValue()
```



Example 3 – Count the hashtags over last 10 mins

```
val tweets = ssc.twitterStream(<Twitter username>, <Twitter password>)
val hashTags = tweets.flatMap (status => getTags(status))
val tagCounts = hashTags.window(Minutes(10), Seconds(1)).countByValue()
```

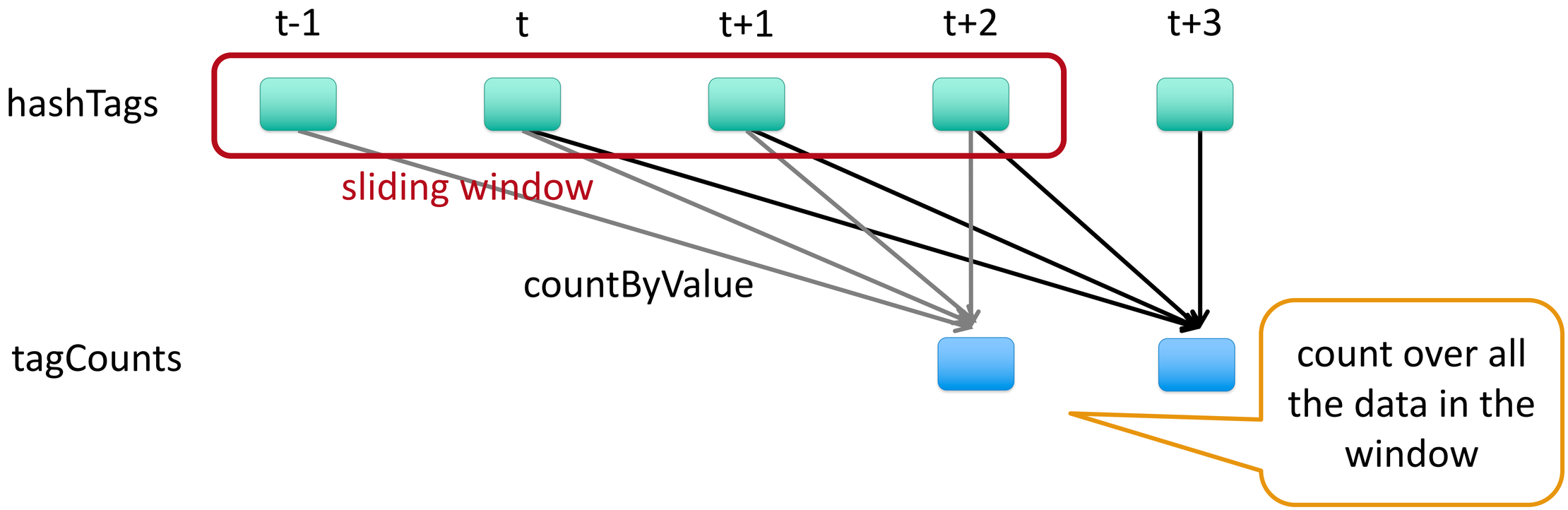
sliding window operation

window length

sliding interval

Example 3 – Counting the hashtags over last 10 mins

```
val tagCounts = hashTags.window(Minutes(10), Seconds(1)).countByValue()
```



Smart window-based countByValue

```
val tagCounts = hashtags.countByValueAndWindow(Minutes(10), Seconds(1))
```