Spark Streaming Tutorial

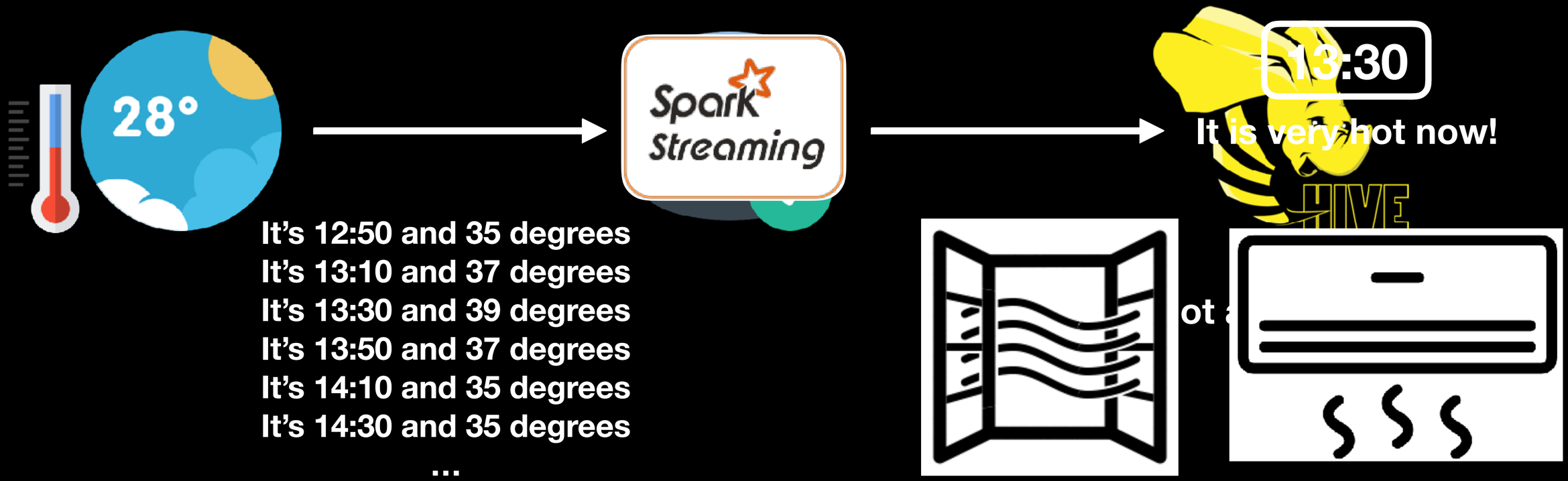
Alex

Agenda

- Why Need Streaming
- Spark Streaming Overview
- DStream and Key Concepts
- A Quick Example
- State and Window Operation
- Demo

Why Need Streaming?

- “Big data” never stop!
- Analyze data streams in real time (results in near-real-time)



Why Need Streaming?

Example use cases are:

- Fraud detection in bank transaction
- Advertising clicks / Mobile device
- Anomalies in sensor data
- Website monitoring / Services monitoring

Why Need Streaming?

Stuff with timestamps can be streaming data:

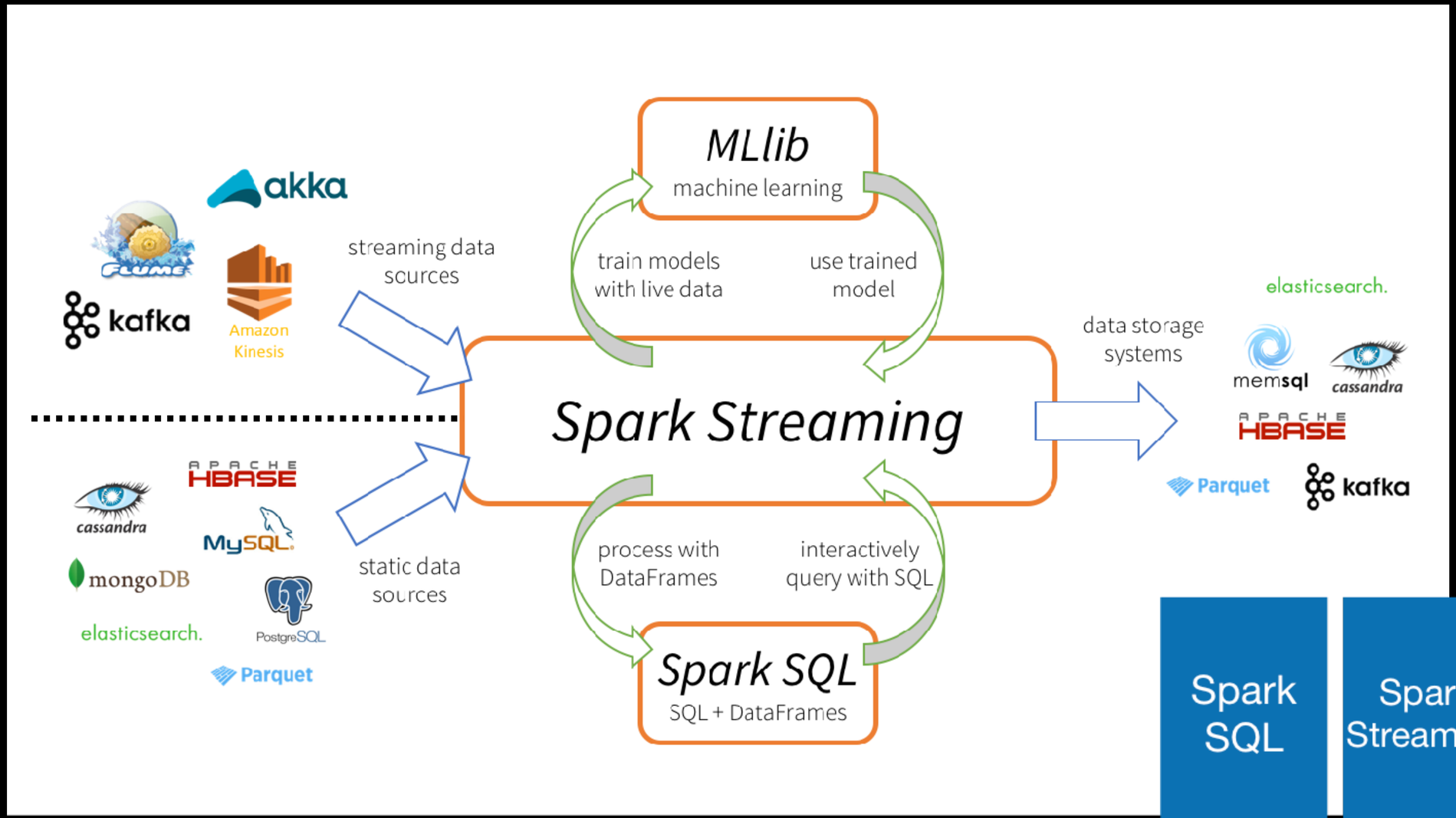
- Fraud detection in bank transactions (credit card transactions)
- Advertising clicks / Mobile device (user behavior log)
- Anomalies in sensor data (sensor logs)
- Website monitoring / Services monitoring (log files)

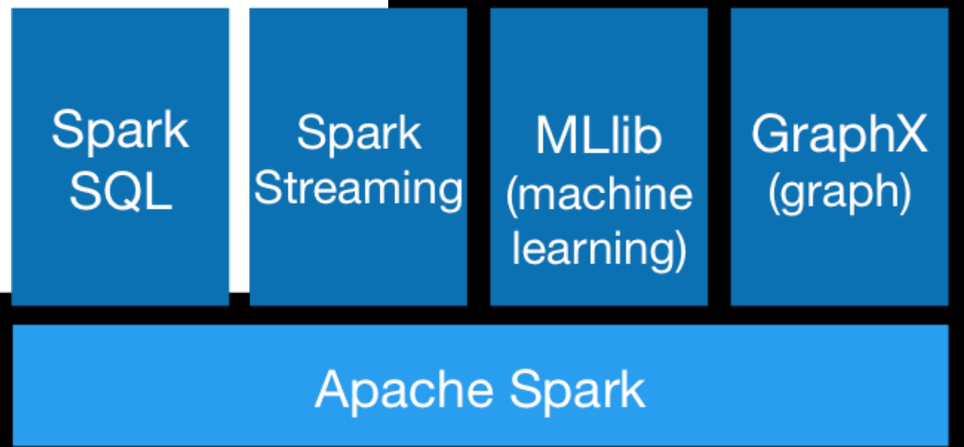
Something +

Agenda

- Why Need Streaming
- Spark Streaming Overview
- DStream and Key Concepts
- A Quick Example
- State and Window Operation
- Demo

Spark Streaming: Overview



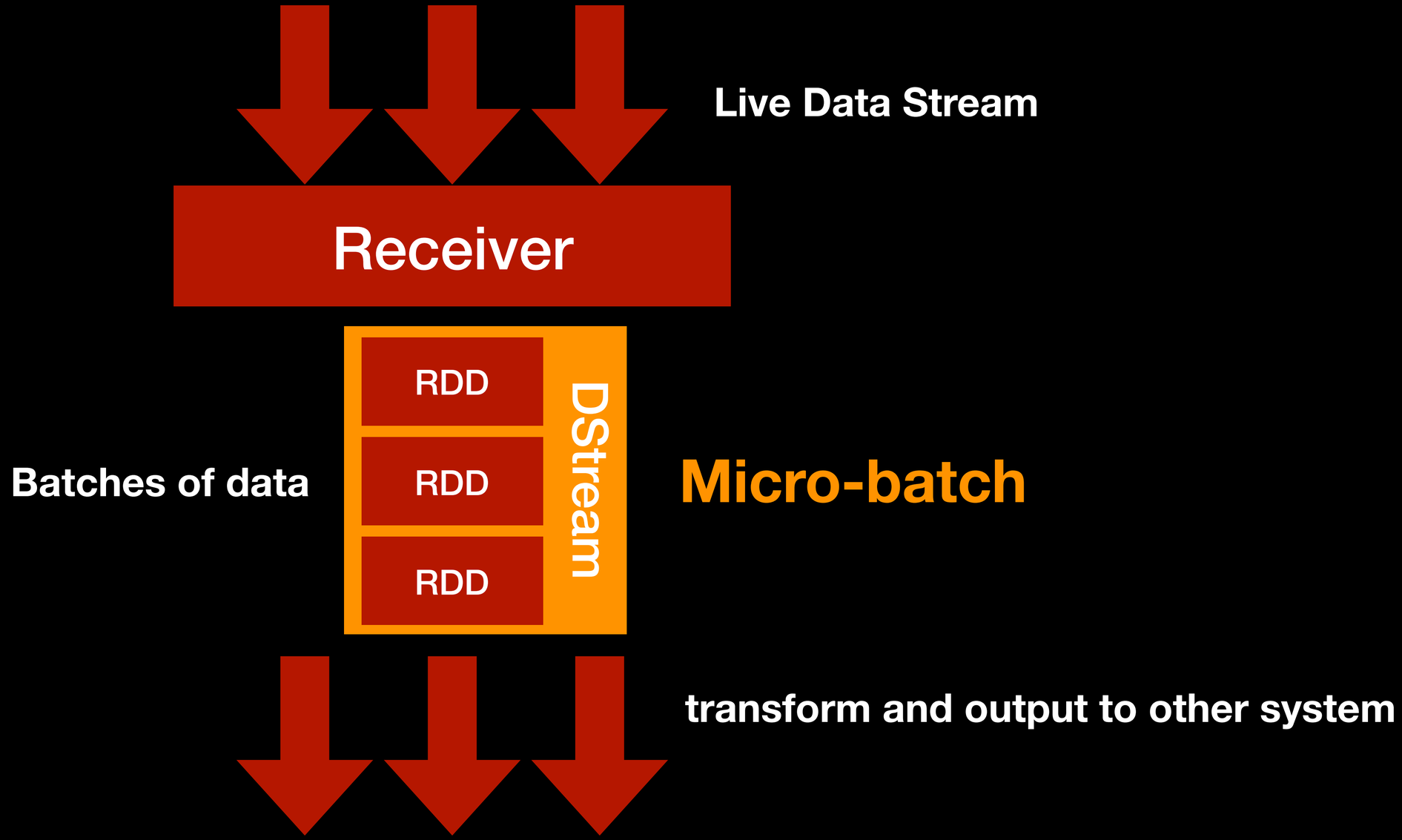


Spark Streaming: Overview

Features:

- **Scaling** : Spark Streaming can easily scale to hundreds of nodes.
- **Speed** : Second-scale latencies (0.5sec. ~ 2sec.)
- **High-Throughput** : 6GB records/sec. on EC2 spark cluster (100 nodes/4core)
- **Fault-Tolerance** : Spark has the ability to efficiently recover from failures.

Spark Streaming: High Level

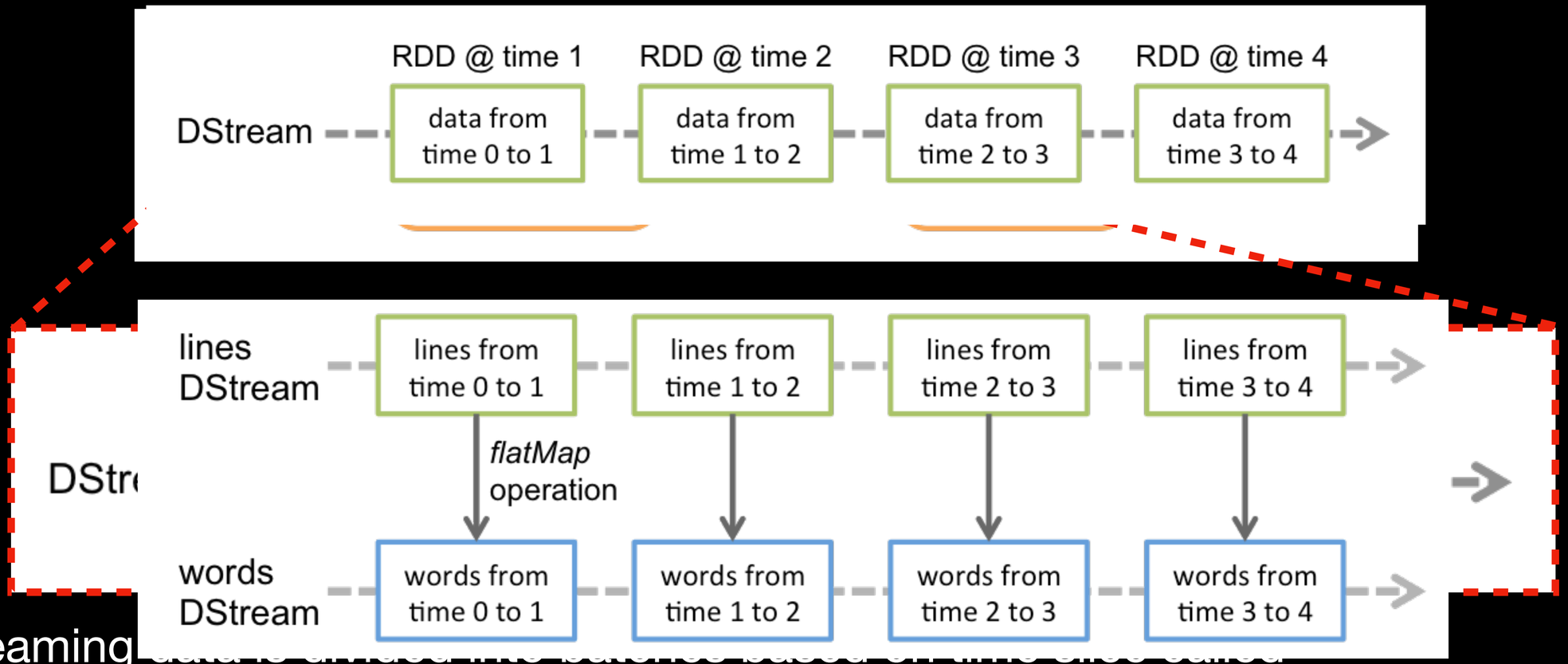


Agenda

- Why Need Streaming
- Spark Streaming Overview
- DStream and Key Concepts
- A Quick Example
- State and Window Operation
- Demo

Spark Streaming: DStream

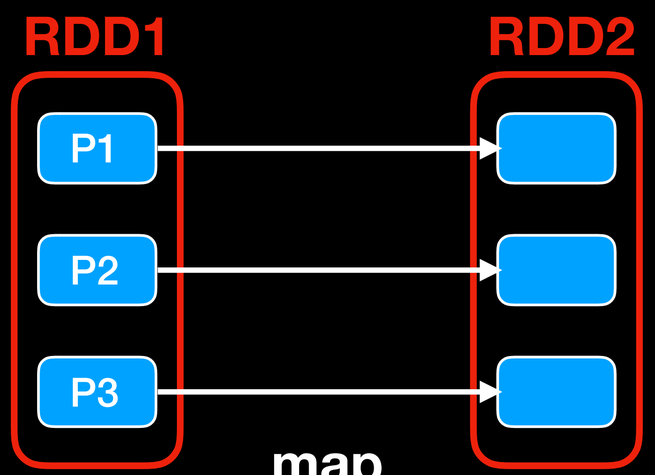
- high-level abstraction called discretized stream (DStream), which represents a continuous stream of data.



- streaming
batch interval

Spark Streaming: Operations

RDD transformations



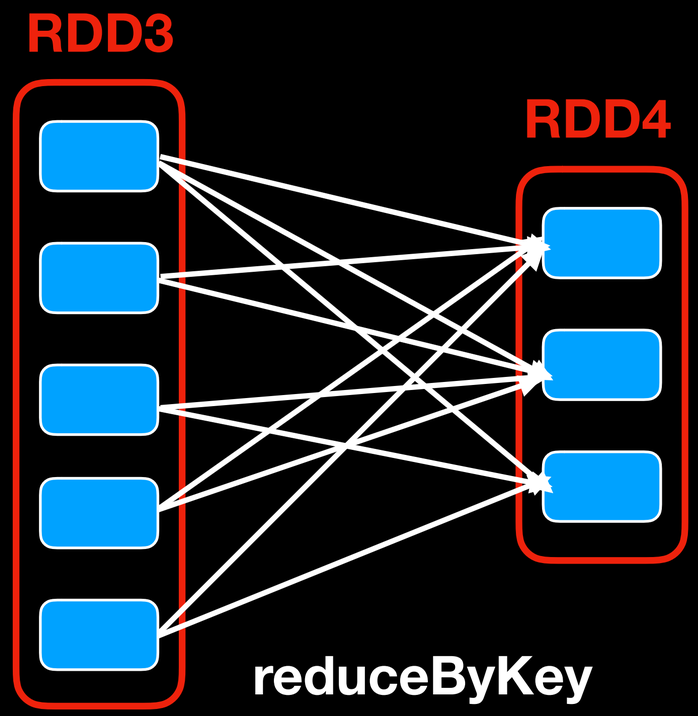
map

flatMap

filter

union

...



reduceByKey

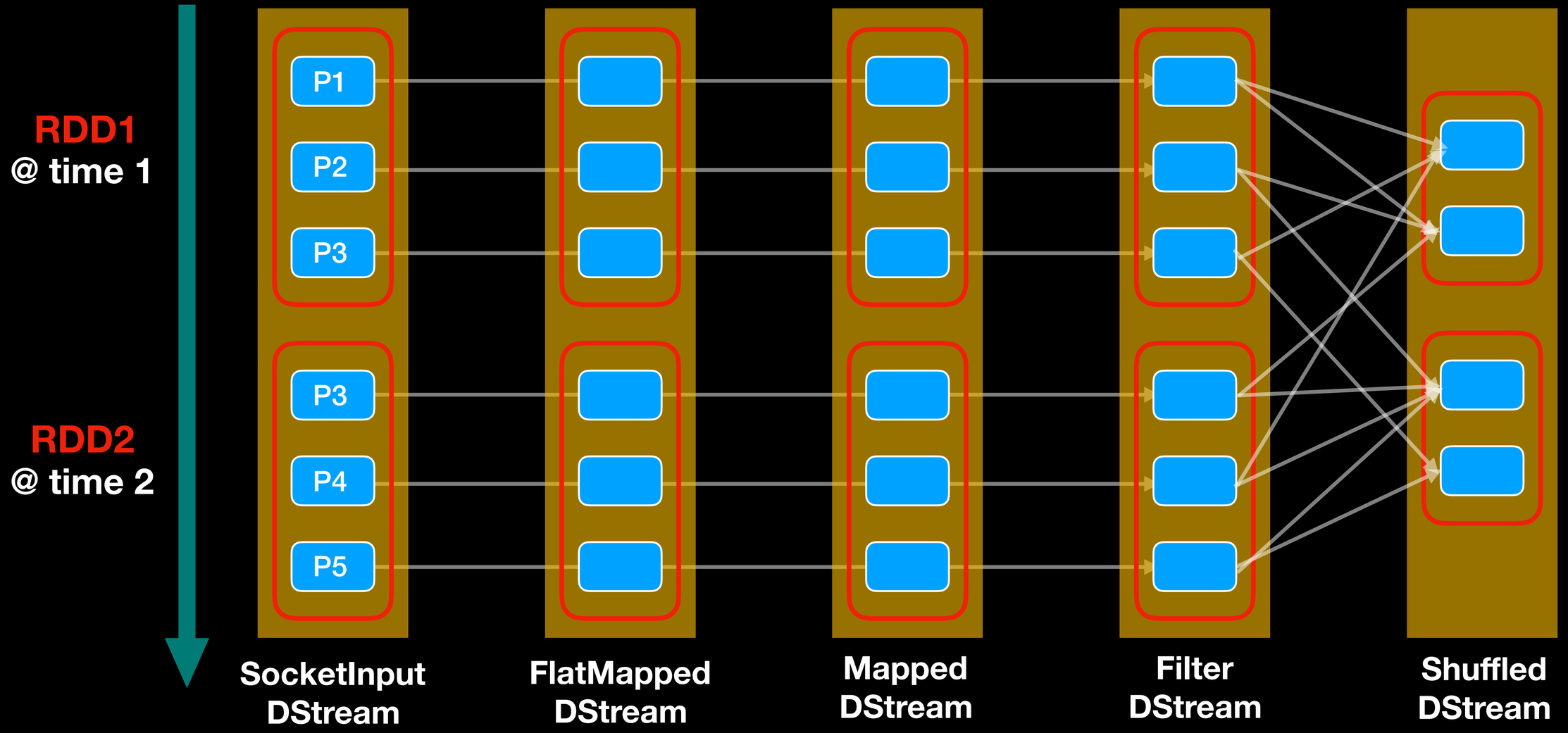
groupByKey

aggregateByKey

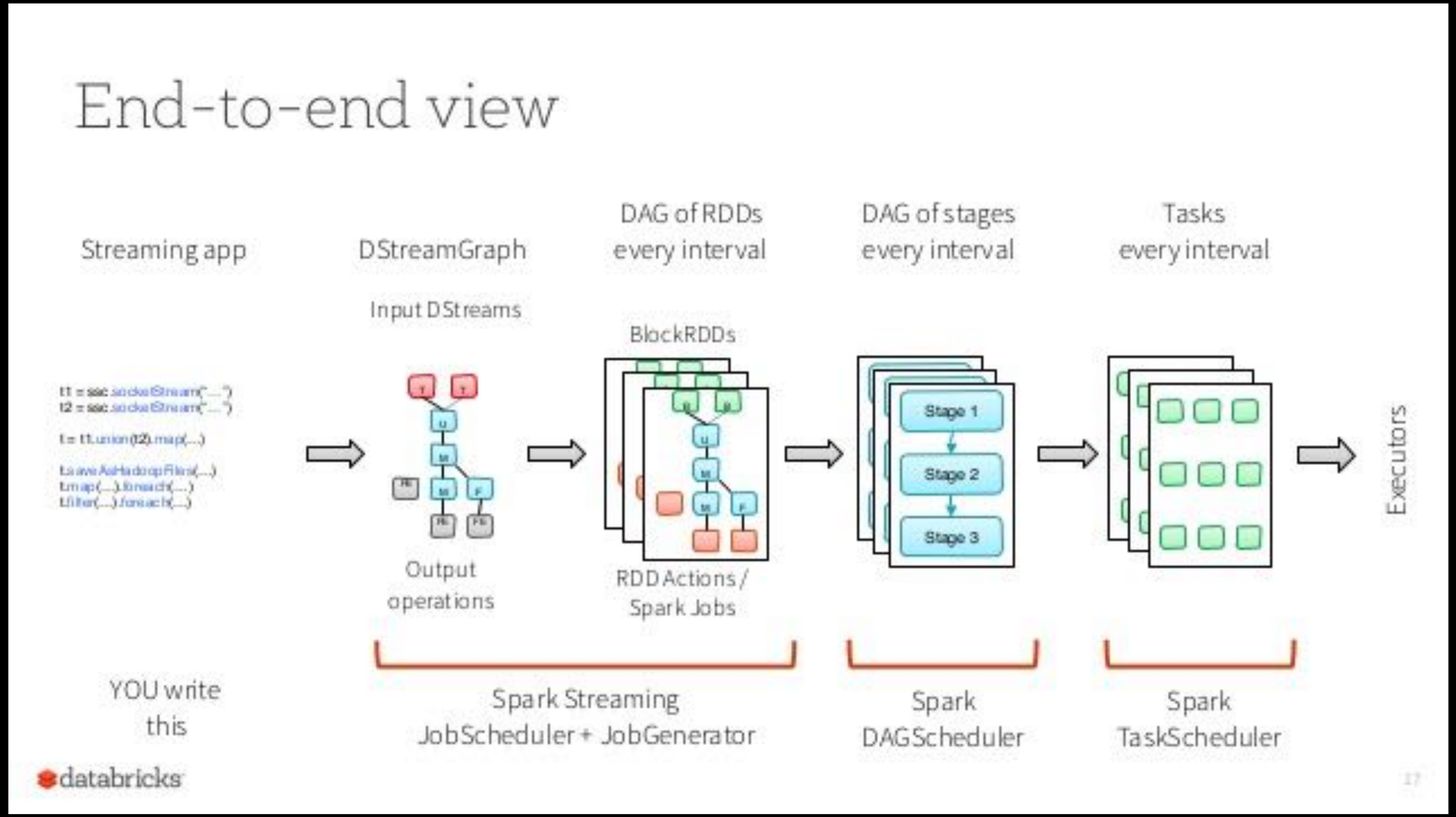
...

Spark Streaming: Operations

DStream transformations



Spark Streaming: Scheduler



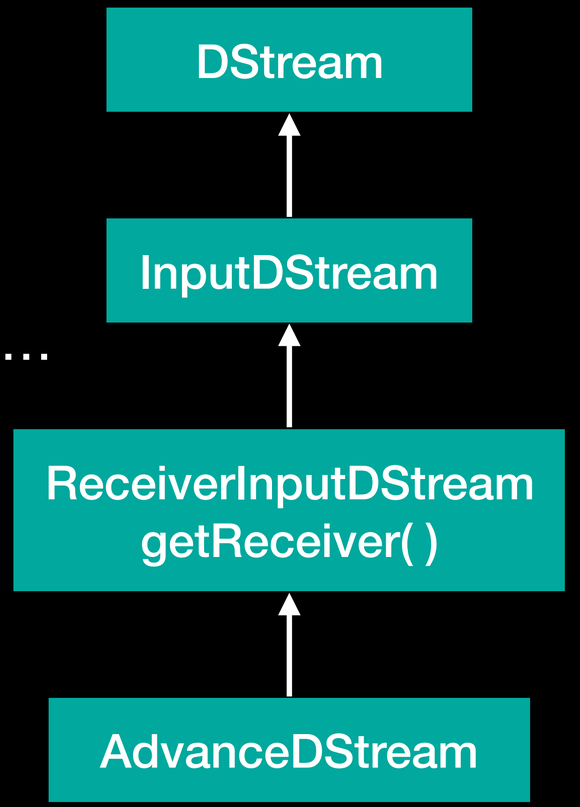
Spark Streaming: key concepts

A DStream is created from a StreamingContext

DStream provides two categories

- Basic sources
  - FileInputDStream, QueueInputDStream
- Advance sources
  - KafkaDStream, FlumeDStream, SocketTextDStream, TwitterDStream…

Every InputDStream is associated with a **Receiver**



Spark Streaming: key concepts

- DStream Transformations
  - Regular transformations such as **map**, **flatMap**, **filter** …
  - Pair transformations such as **reduceByKey**, **groupByKey**, **join** …

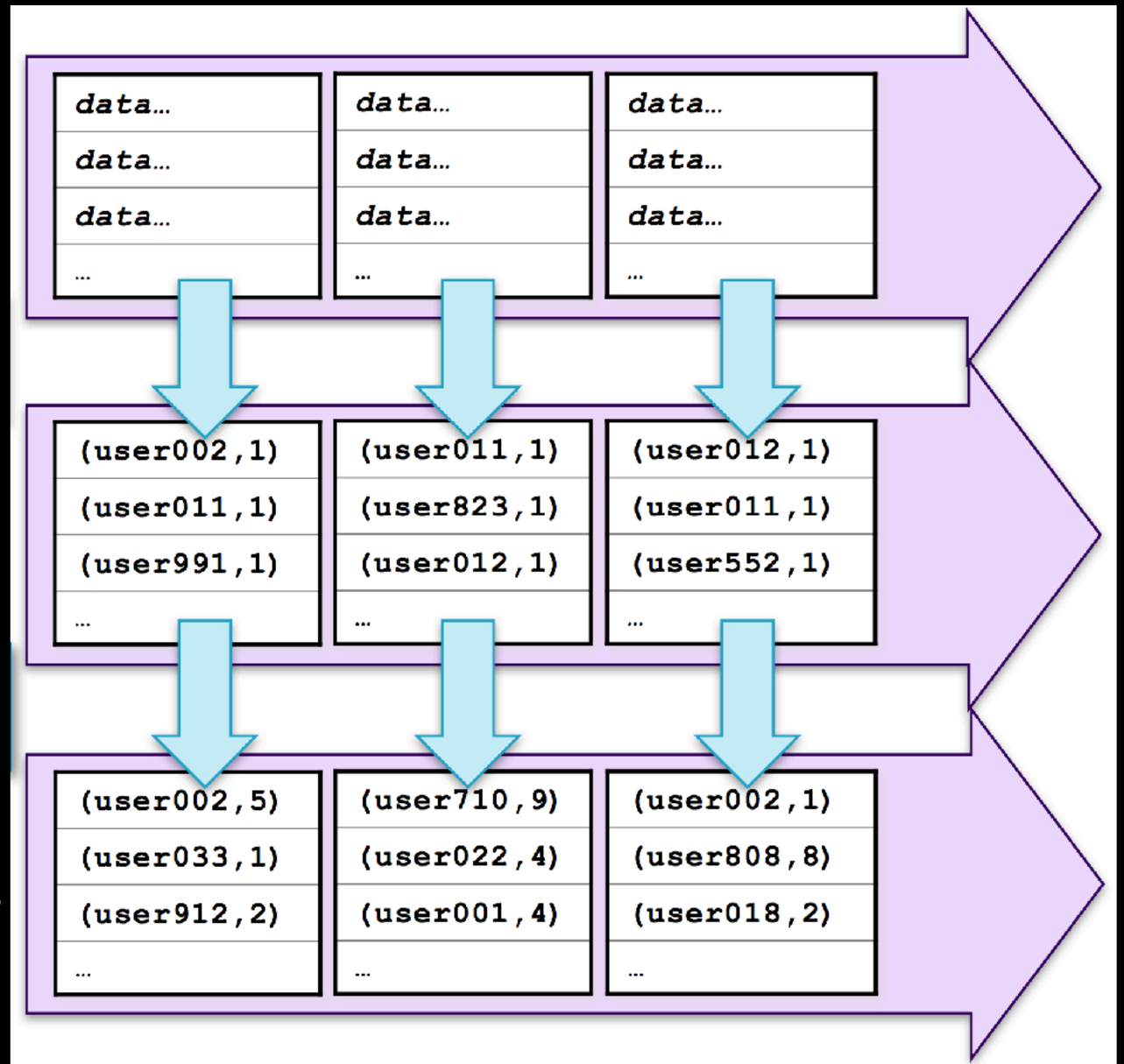
- DStream Output Operations
  - Console output (print / pprint)
  - File output (saveAsTextFiles)
  - Executing other functions (foreachRDD)

Spark Streaming: sample code

```
userreqs = logs.map(line =>
  (line.split('  ')(2),1))
```

Language: Scala

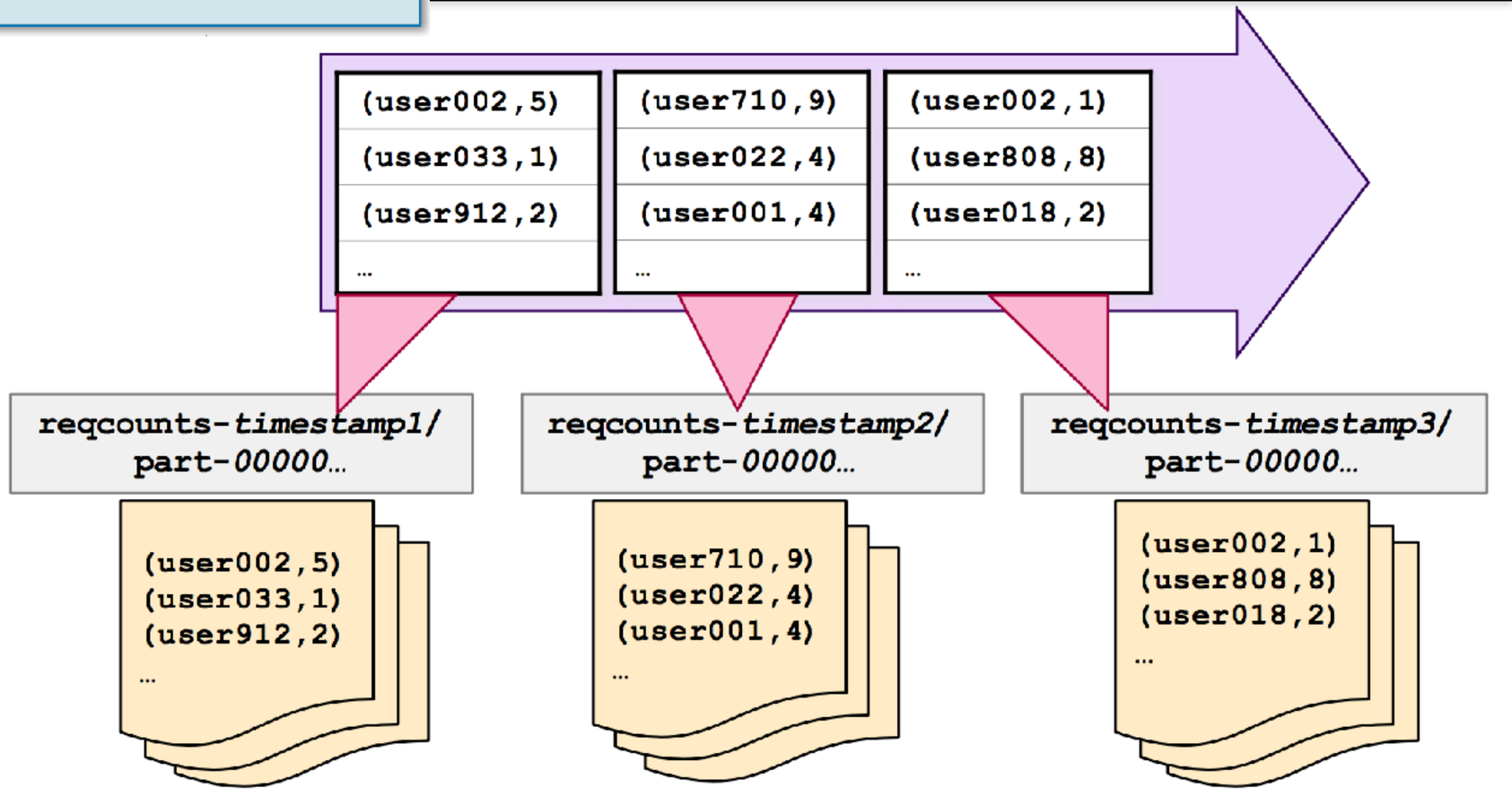
```
reqcounts = userreqs.
  reduceByKey((x,y) => x+y)
```



Spark Streaming: sample code

Language: Scala

```
val userreqs = logs
  .map(line => (line.split(' ')(2),1))
  .reduceByKey((v1,v2) => v1+v2)
userreqs.print()
userreqs.saveAsTextFiles("…/outdir/reqcounts")
```



Agenda

- Why Need Streaming
- Spark Streaming Overview
- DStream and Key Concepts
- A Quick Example
- State and Window Operation
- Demo

A Quick Example: Streaming wordcount

```scala
import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._ // not necessary since Spark 1.3

// Create a local StreamingContext with two working thread and batch interval of 1 second.
// The master requires 2 cores to prevent from a starvation scenario.

val conf = new SparkConf().setMaster("local[2]").setAppName("NetworkWordCount")
val ssc = new StreamingContext(conf, Seconds(1))
```

```scala
// Create a DStream that will connect to hostname:port, like localhost:9999
val lines = ssc.socketTextStream("localhost", 9999)
```

```scala
// Split each line into words
val words = lines.flatMap(_.split(" "))
```

```scala
import org.apache.spark.streaming.StreamingContext._ // not necessary since Spark 1.3
// Count each word in each batch
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey(_ + _)

// Print the first ten elements of each RDD generated in this DStream to the console
wordCounts.print()
```

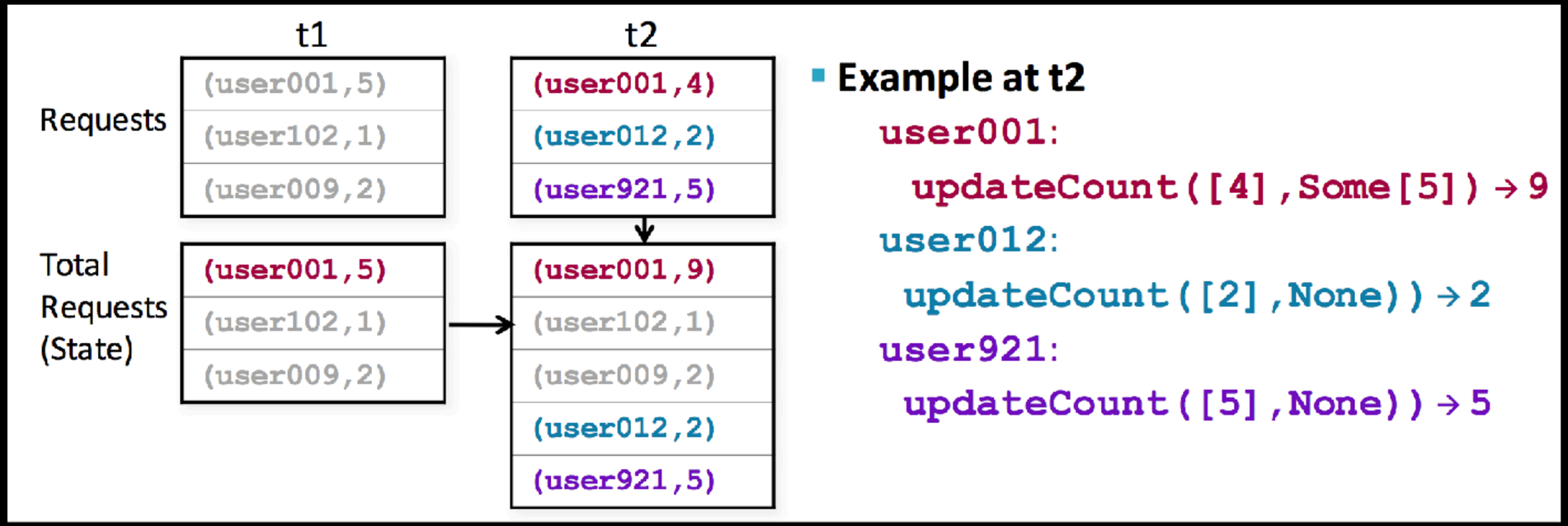
```scala
ssc.start()             // Start the computation
ssc.awaitTermination()  // Wait for the computation to terminate
```

```
# TERMINAL 1:
# Running Netcat

$ nc -lk 9999

hello world



...
```

Agenda

- Why Need Streaming
- Spark Streaming Overview
- DStream and Key Concepts
- A Quick Example
- State and Window Operation
- Demo

State Operation



- **Example at t2**

```
user001:
  updateCount([4],Some[5]) → 9
user012:
  updateCount([2],None)) → 2
user921:
  updateCount([5],None)) → 5
```

New Values

Current State (or None)

```
def updateFunction(newValues: Seq[Int], runningCount: Option[Int]): Option[Int] = {
  val newCount = runningCount.getOrElse(0) + newValues.sum
  Some(newCount)
}
```

```
// Update state using `updateStateByKey`
val runningCounts = pairs.updateStateByKey[Int](updateFunction _)
```

Window operation

Batch interval vs. window length vs. sliding interval

- Batch Interval : how often data is divided into batches
- Window Length : how often a windowed transformation is computed
- Sliding Interval : how far back in time the windowed transformation goes

window(*windowLength*, *slideInterval*)

reduceByWindow(*func*, *windowLength*, *slideInterval*)

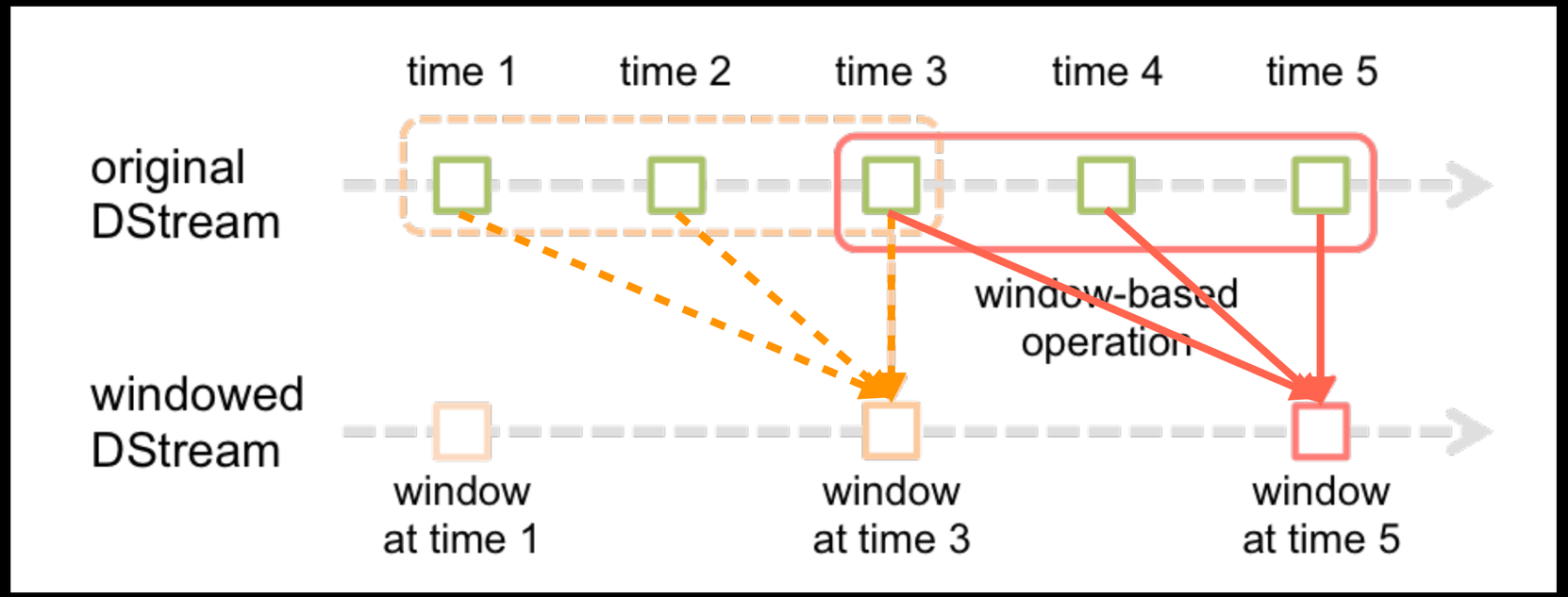
countByWindow(*windowLength*, *slideInterval*)

reduceByKeyAndWindow(*func*, *windowLength*, *slideInterval*, [*numTasks*])

reduceByKeyAndWindow(*func*, *invFunc*, *windowLength*, *slideInterval*, [*numTasks*])

countByValueAndWindow(*windowLength*, *slideInterval*, [*numTasks*])

Window operation



- **Batch interval : 1**
- **window length : 3**
- **sliding interval : 2**

```
// Reduce last 30 seconds of data, every 10 seconds
val windowedWordCounts = pairs.reduceByKeyAndWindow((a:Int,b:Int) => (a + b), Seconds(30), Seconds(10))
```

Agenda

- Why Need Streaming
- Spark Streaming Overview
- DStream and Key Concepts
- A Quick Example
- State and Window Operation
- Demo