Apache Spark Streaming

Borja Moreno Pozo

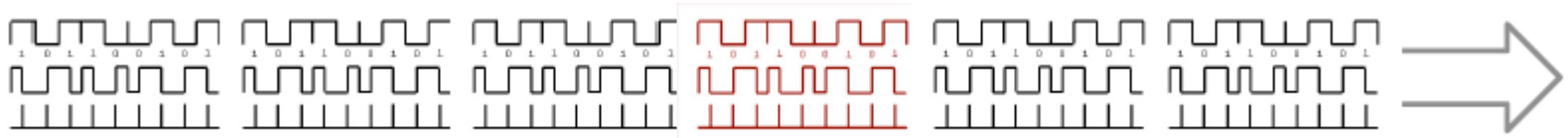
INDICE

- ¿Qué es streaming?
- ¿Ques es Apache Spark Streaming?
- Arquitectura
- Ejemplo

Que es streaming

- Motivación I
  Muchas aplicaciones deben procesar grandes volúmenes de datos de formar continua y proveer resultados en tiempo real
  - Detección de fraude en transacciones bancarias

  - Detección de anomalías en sensores de datos



Que es streaming

- Motivación II
  - Necesitamos poder escalar el rendimiento sobre Cluster de muchos nodos para manejar el volumen de datos
  - Pero debemos tratar los datos con latencias de unos pocos segundos

Que es streaming

- Requisitos

  Necesitamos un framework que nos permita construir estas aplicaciones de tratamiento complejo de datos en tiempo real

  - Debemos poder escalar a Clusters de gran tamaño
  - Manejar latencias en torno al segundo
  - Tener un modelo de programación simple
  - Integrar el procesamiento Batch & Real-time
  - Manejar los cambios de estado de forma eficiente y a prueba de fallos

Que es streaming

- Los frameworks actuales no pueden trabajar con ambos grandes volúmenes de datos y baja latencia
  - O procesamos en Stream 100s de MB/s con baja latencia
  - O procesamos en Batch TBs de datos con latencia elevada
- Esto obliga a mantener 2 stacks distintos; uno para Batch y otro para Streaming
  - Esto implica trabajar con dos interfaces diferentes
  - Duplicar el esfuerzo de implementación
  - Duplicar los costes operacionales
  - Ambos sistemas frecuentemente reprocesan los mismos datos

Que es streaming

- Procesamiento de Streams con estado
  Los sistemas tradicionales de streaming tienen un modelo de procesamiento basado en estados
  - Cada nodo mantiene un estado
  - Cada nuevo registro actualiza el estado de los datos del nodo y envía los registros actualizados al resto

El estado se pierde si el nodo falla

Hacer que el procesamiento en tiempo real a base de estados sea a prueba de fallos es complejo

Que es Spark Streaming

- Spark Streaming realiza 3 tareas
  - Recibe streams de las fuentes de datos
  - Los procesa en un Cluster Spark
  - Publica los resultados en bases de datos externas o directamente a paneles de control

- Spark Streaming es escalable, a prueba de fallos y con latencias en el entorno de los segundos

Que es Spark Streaming

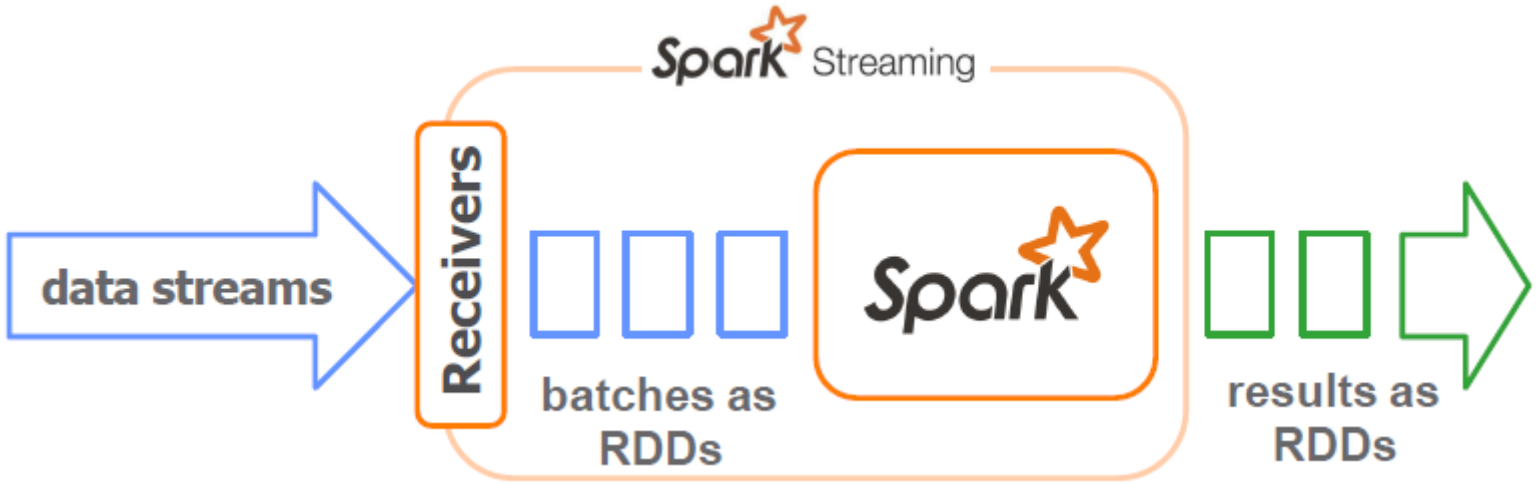
- Fuentes de datos
  - Nativas
    - Kafka, Flume, Kinesis
    - TCP sockets
    - HDFS, S3
  - Se puede implementar un receiver propio
  - También es posible generar una secuencia de RDDs y tratarla como un Stream

Que es Spark Streaming

- Destino de datos
  - Nativas
    - HDFS,S3
  - Mediante conectores
    - Cassandra
    - Hbase
  - Enviando los ficheros de forma explicita

Que es Spark Streaming

- Como funciona Spark Streaming
  - El Stream de datos se agrupa en lotes de unos segundos de “ancho”
  - Spark maneja cada lote como un RDD y lo procesa utilizando primitivas de RDDs
  - Los resultados se emiten por lote procesado

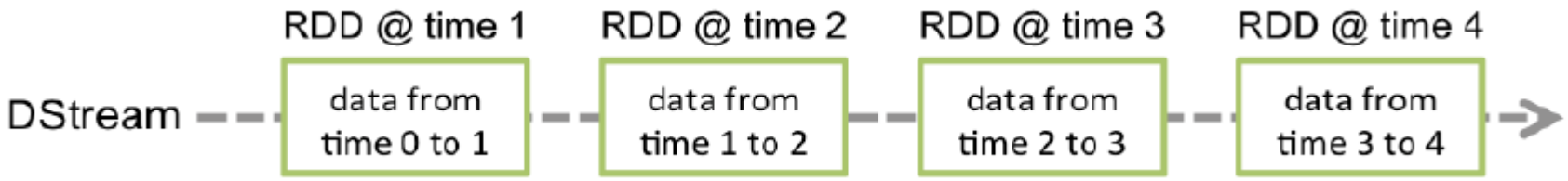


Que es Spark Streaming

- Modelo de Programación de Spark Streaming
  - Discretized Stream (Dstream)
    - Representa un stream de datos
    - Se implementa como una secuencia de RDDs
  - Dstream API ~ RDD API
    - API funcional bajo Scala, Java, Python
    - Permite asociar los Dstream a diversas fuentes
    - Aplicar las operaciones distribuidas de RDDs

Que es Spark Streaming

- Modelo de Programación de Spark Streaming: Discretized Stream (Dstream) Representa una entrada de datos en tiempo real en un intervalo de tiempo determinado.
  - Se puede crear desde múltiples fuentes de datos (Kafka, Twitter, etc)
  - La entrada de datos es dividida en micro batches. Vamos recuperando la entrada de datos cada un X tiempo definido como una ventana.
  - Cada micro-batch es un RDD.
  - El Dstream es una secuencia de RDD's

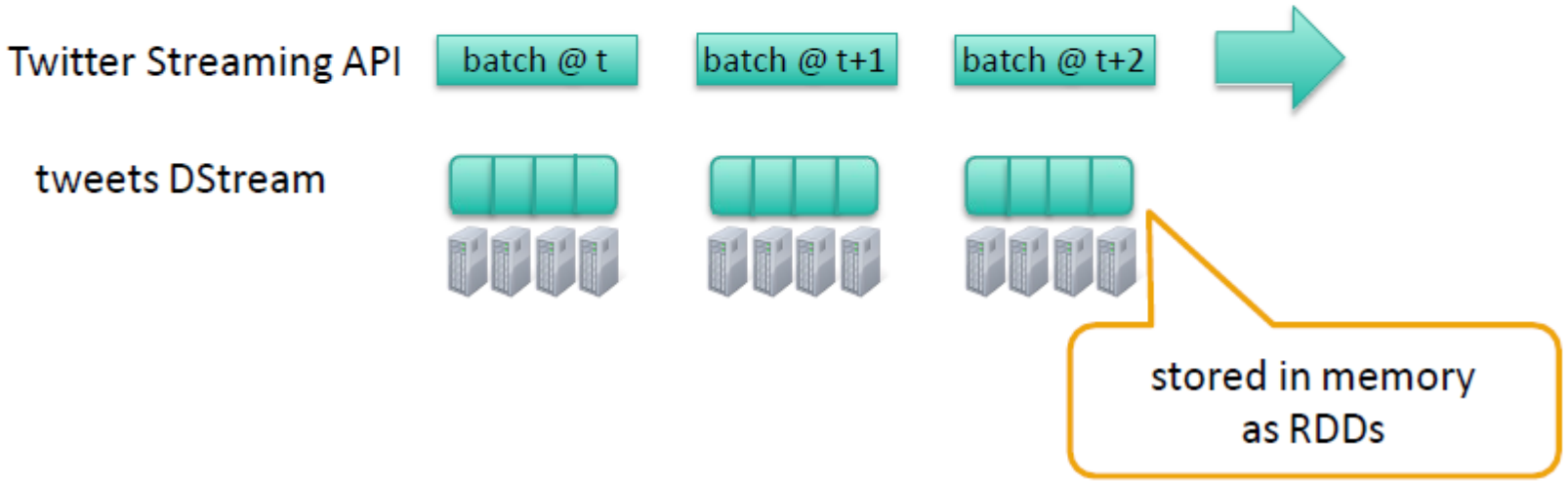


Que es Spark Streaming

- Ejemplo – Twitter I

```
val ssc = new StreamingContext(sparkContext, Seconds(1))
val tweets = TwitterUtils.createStream(ssc, auth)
```

Input DStream



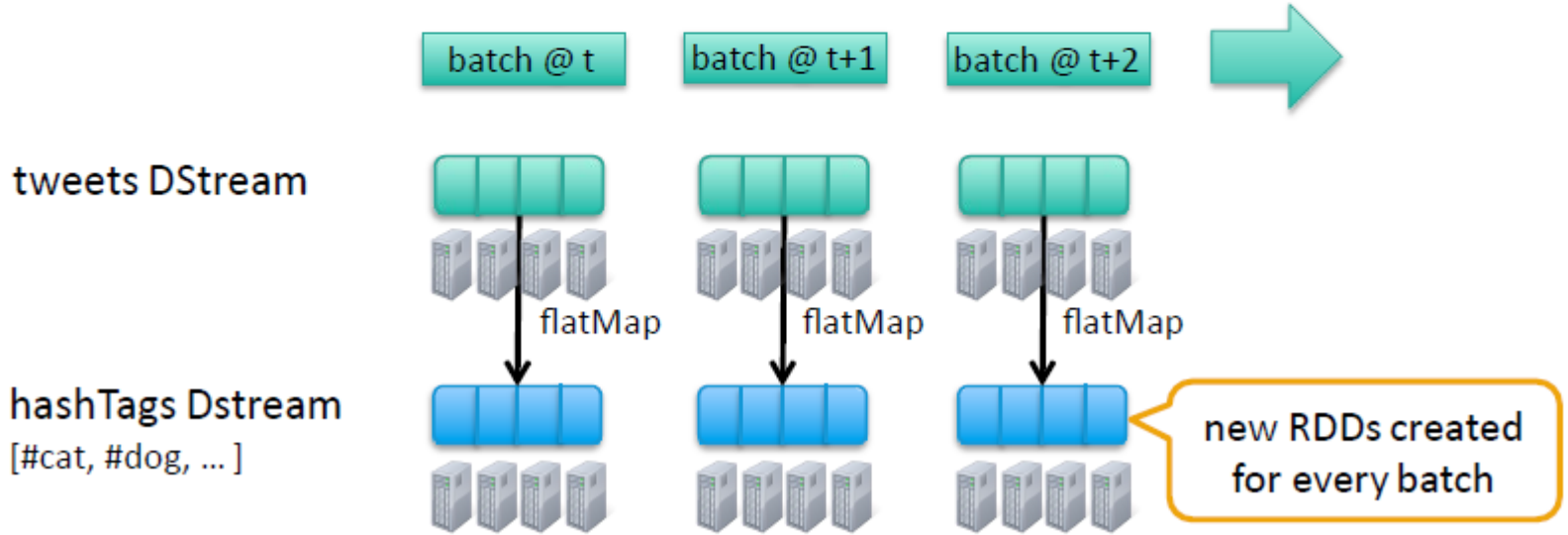
Que es Spark Streaming

- Ejemplo – Twitter II

```
val tweets = TwitterUtils.createStream(ssc, None)
val hashTags = tweets.flatMap(status => getTags(status))
```

transformed DStream

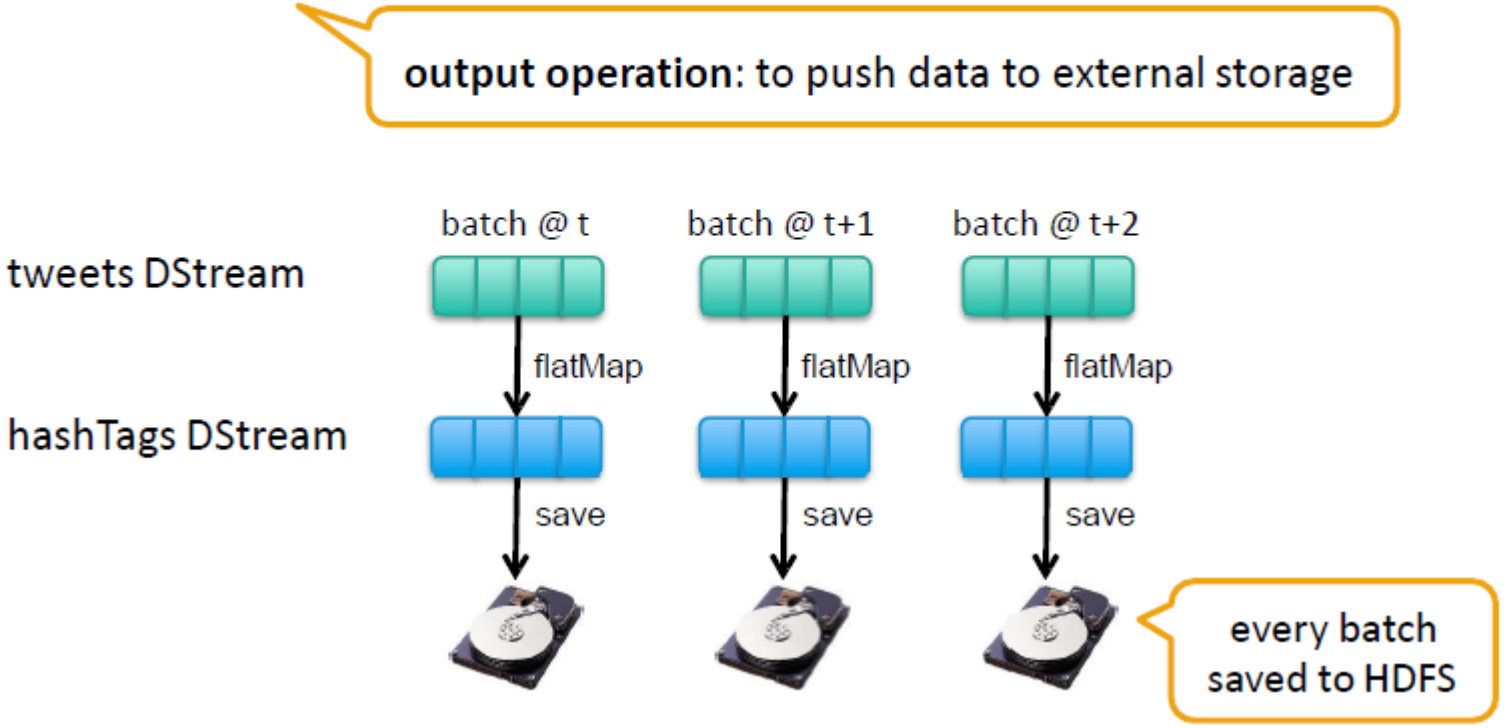
transformation: modify data in one DStream to create another DStream



Que es Spark Streaming

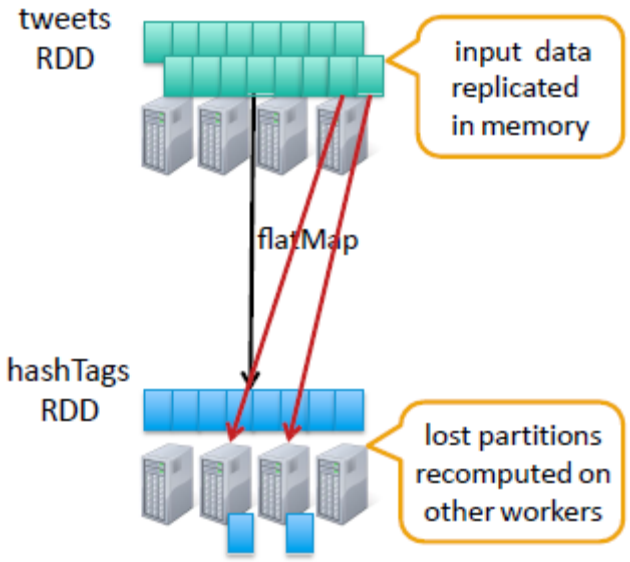
- Ejemplo – Twitter III

```
val tweets = TwitterUtils.createStream(ssc, None)
val hashTags = tweets.flatMap(status => getTags(status))
hashTags.saveAsHadoopFiles("hdfs://...")
```



Que es Spark Streaming

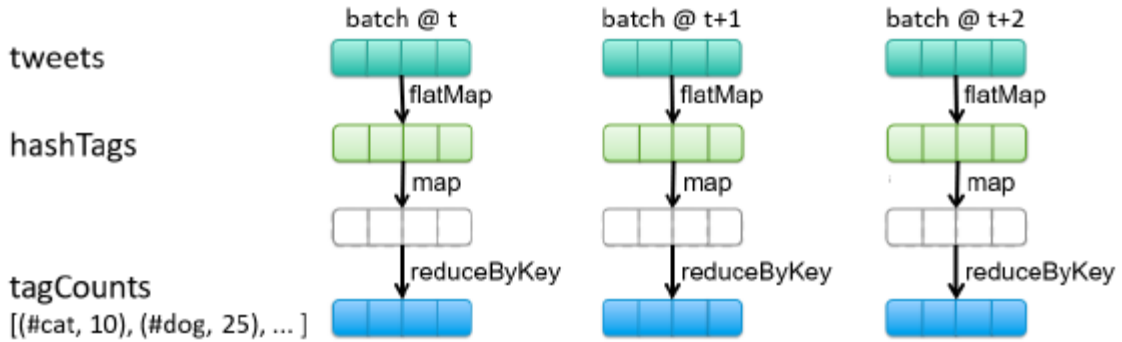
- Tolerancia a fallos
  - Los lotes de datos son almacenados en memoria de forma distribuida
  - Si un nodo trabajador falla, los datos se pueden recomputar a partir de la información replicada
  - Todas las transformaciones son a prueba de fallos y se aplican solo una vez



Que es Spark Streaming

- Operaciones globales

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



Que es Spark Streaming

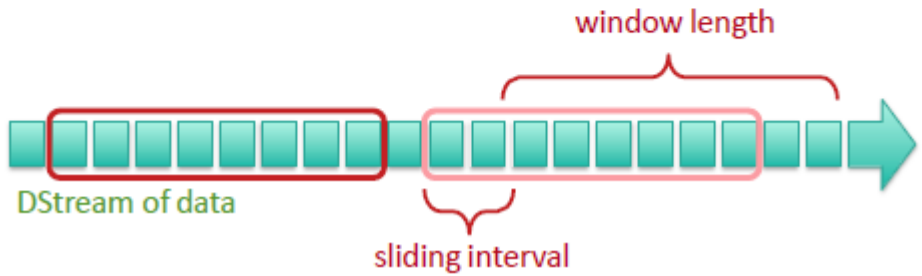
- Operaciones con ventanas I. Agregación asociativa sobre ventanas

```
val tweets = TwitterUtils.createStream(ssc, auth)
val hashTags = tweets.flatMap(status => getTags(status))
val tagCounts = hashTags.window(Minutes(1), Seconds(5)).countByValue()
```

sliding window operation

window length

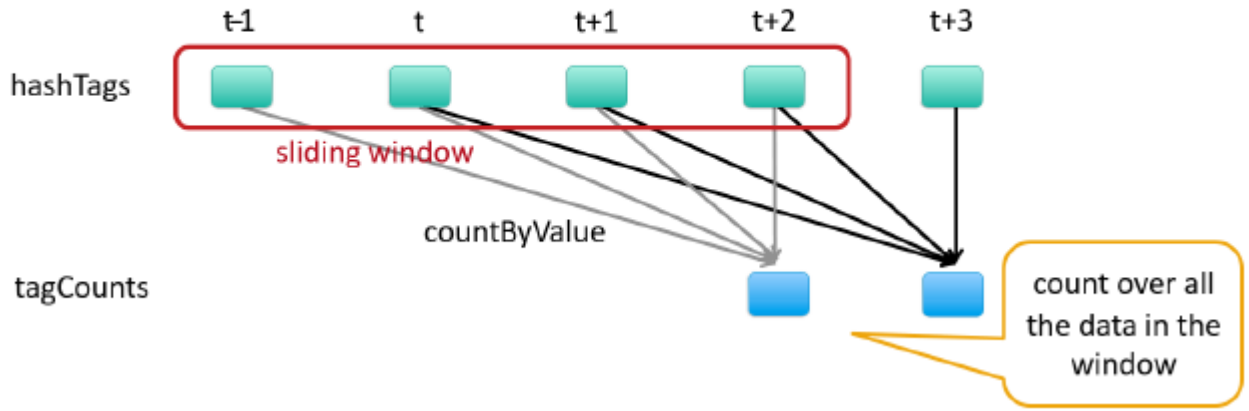
sliding interval



Que es Spark Streaming

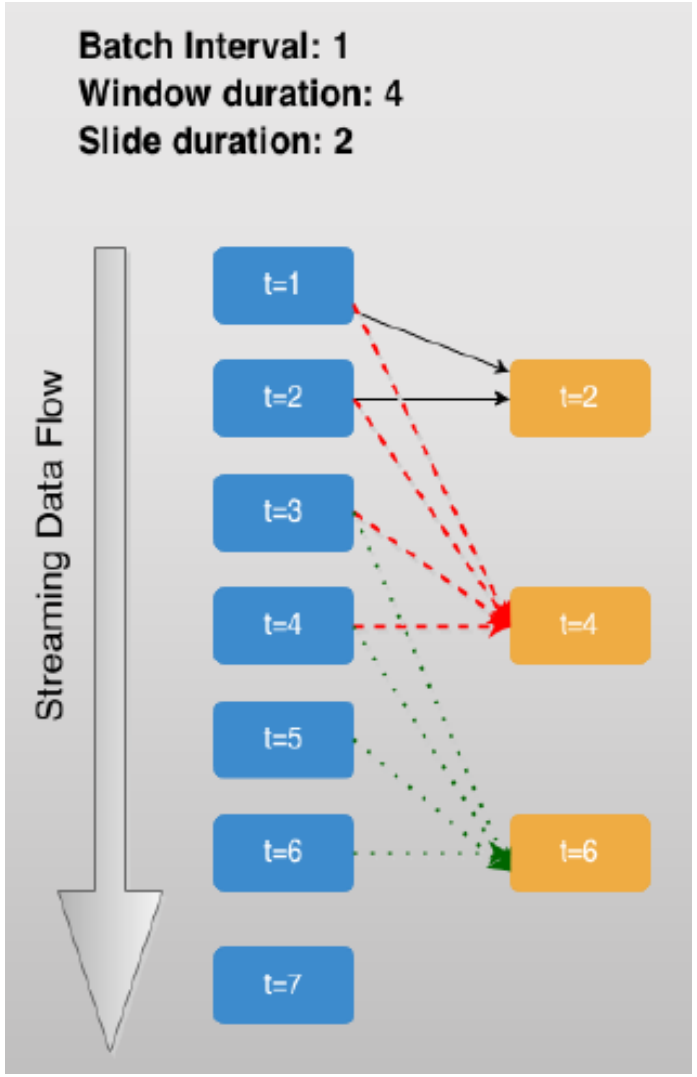
- Operaciones con ventanas II. Agregación asociativa sobre ventanas

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



Que es Spark Streaming

- Operaciones con ventanas III



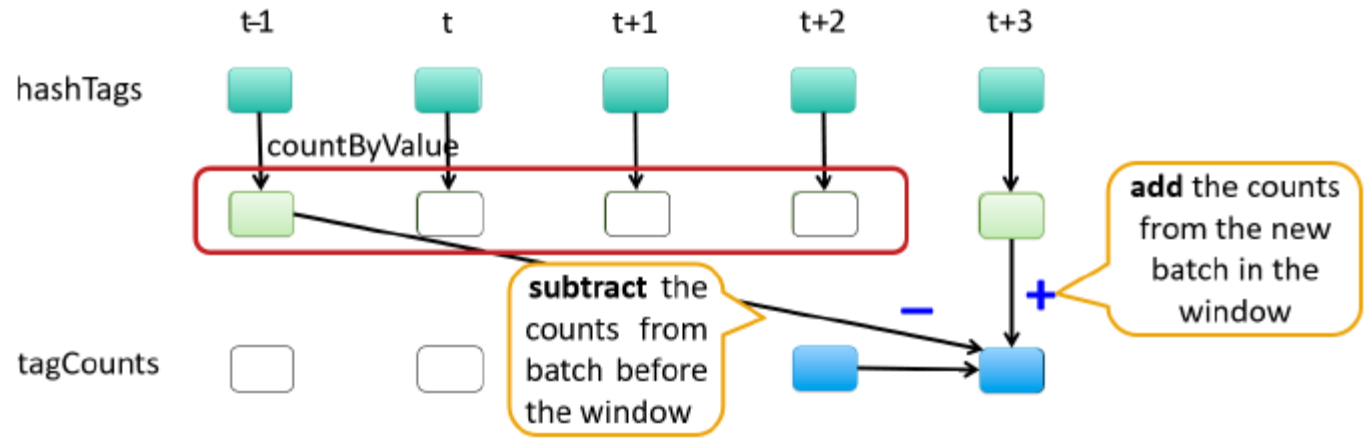
Que es Spark Streaming

- Operaciones con ventanas IV
  - Asociación incremental agregada sobre ventanas(Incremental Associative Aggregation over a Window)
    Para tamaño de ventanas muy grandes con ciclos muy pequeños, la agregación puede ser el cuello de botella. Para evitarlo 2 requerimientos
    - Funcion de reducion inversa. Tiene que ser una función inyectiva. El usuario tiene que ser capaz de dar una función inversa para dar cuenta de los datos antiguos que salen de la ventana. Para un wordcount la función inversa sería la resta de los counts.
      wordStream.reduceByKeyAndWindow((x: Int, y: Int) => x+y, (x: Int, y: Int) => x-y, windowSize, slidingInterval)
    - Dstream Checkpointing. Hacer checkpoints de manera frecuente puede reducir el peso del stream porque muchos rdds serán persistidos
  - Asociación incremental agregada con Pruning (Incremental Associative Agregation with Pruning)
    El key space puede crecer mucho mientras que la función inversa ayuda a actualizar los valores pero no a limpiarlos. En el wordcount, habrá muchas palabras que tengan un count 0 y puede ser innecesarias. Si se aplica un filtro para las keys que no sean importantes

Que es Spark Streaming

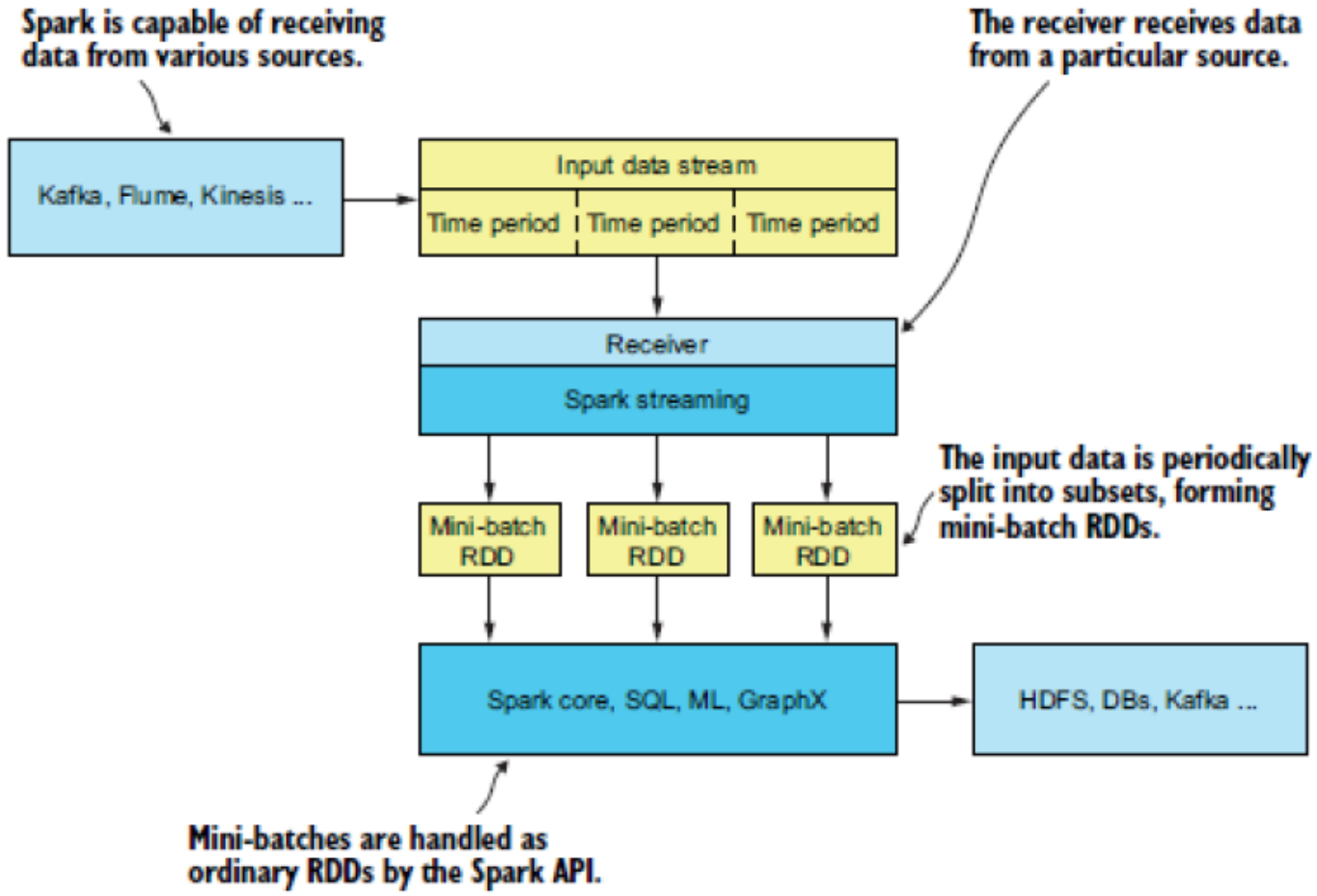
- Operaciones con ventanas III

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



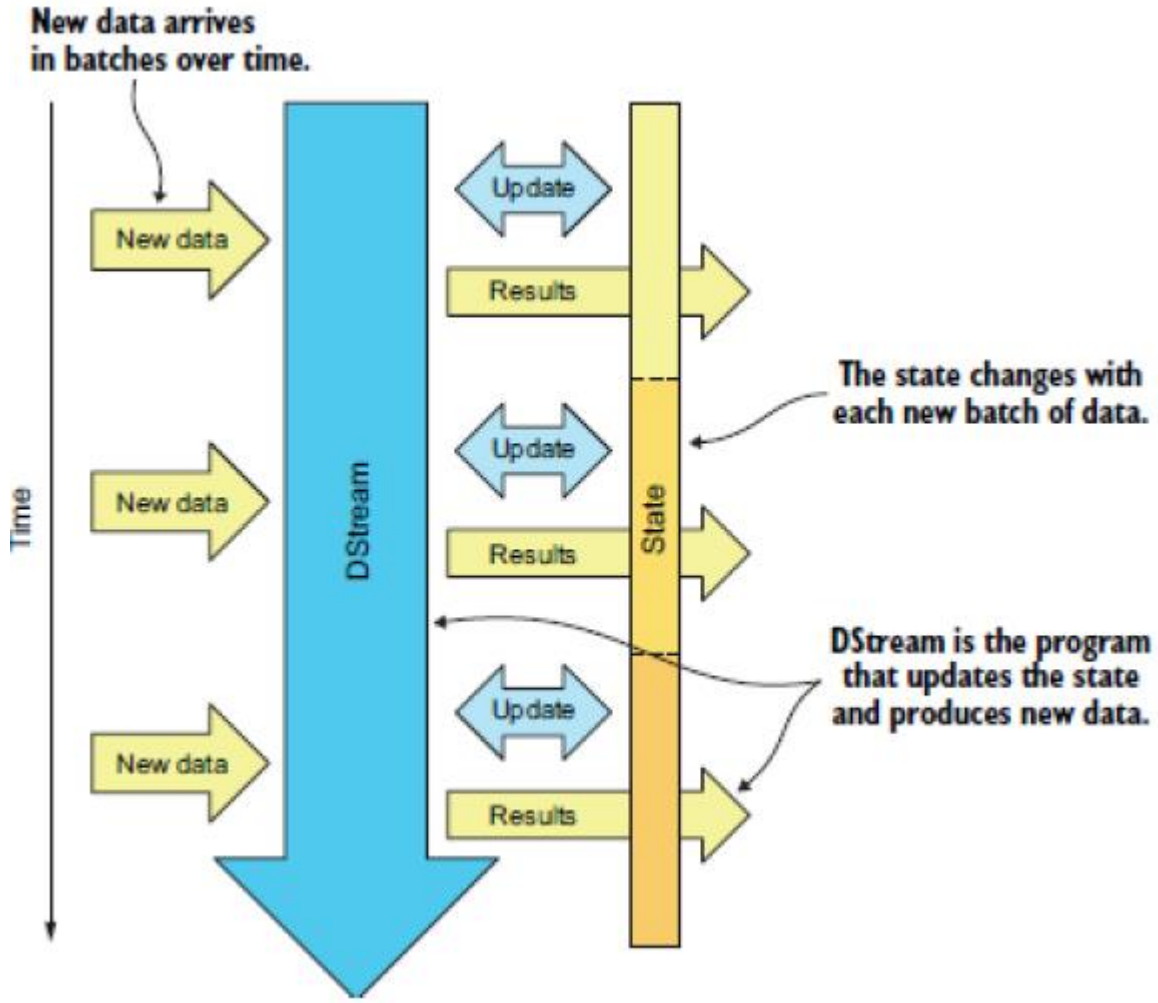
Que es Spark Streaming

- Esquema flujo procesamiento



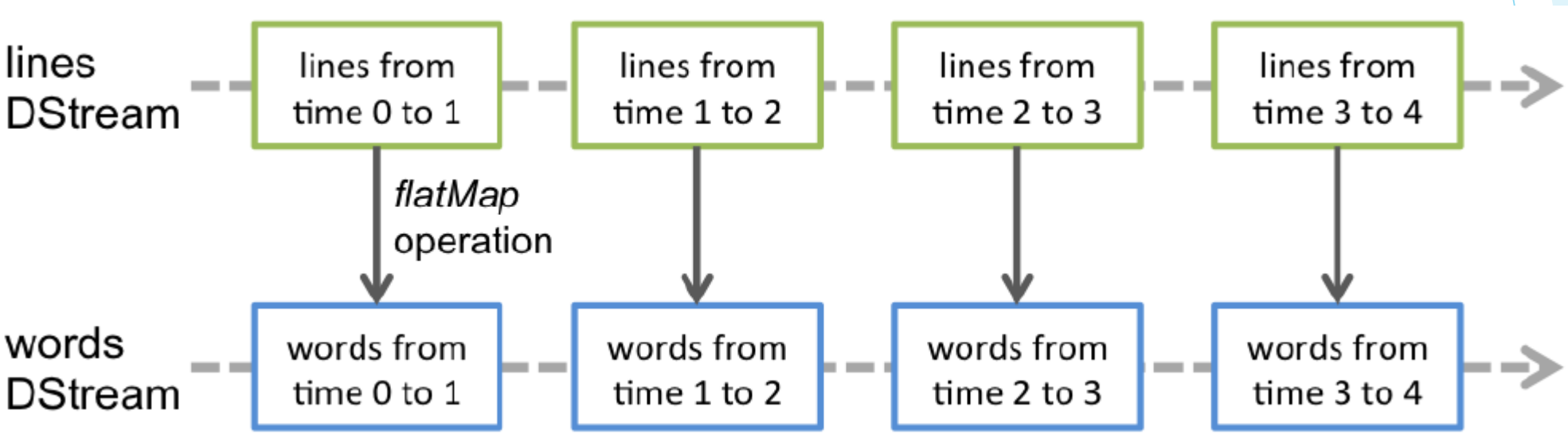
Que es Spark Streaming

- Esquema flujo procesamiento



Que es Spark Streaming

- Esquema flujo procesamiento



Que es Spark Streaming

- Acciones sobre Dstreams:
  - Compartir estados entre los diferentes streams mediante métodos como **updateStateByKey** podemos crear “variables compartidas” entre cada stream
  - Hacer operaciones de tipo unión entre diferentes Streams con el método **union**
  - Aplicar checkpoint en caso de error, para que el Stream se reanude desde ese directorio donde guardemos el ultimo estado con **setCheckpointDir**
  - Unificar estados por una key y una ventana de tiempo con **reduceByKeyAndWindow**

Que es Spark Streaming

- Flujo básico de programa – Scala (http://spark.apache.org/docs/latest/streaming-programming-guide.html)

1. Definir el contexto de ejecución inicializando un objeto StreamingContext (SparkContext, sliding_Interval_time). Solo puede existir un StreamingContext activo a la vez.

```
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
// create a StreamingContext
val ssc = new StreamingContext(sc, Seconds(10))
```

Que es Spark Streaming

- Flujo básico de programa – Scala (http://spark.apache.org/docs/latest/streaming-programming-guide.html)

2. Especificar las fuentes de entrada creando los correspondientes Dstreams

```
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
// create a StreamingContext
val ssc = new StreamingContext(sc, Seconds(10))
// create a DStream that will connect to serverIP:serverPort
val lines = ssc.socketTextStream(serverIP, serverPort)
```

Que es Spark Streaming

- Flujo básico de programa – Scala (http://spark.apache.org/docs/latest/streaming-programming-guide.html)

3. Definir el procesamiento del stream usando el API de transformacoines sobre DStreams

```
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
// create a StreamingContext
val ssc = new StreamingContext(sc, Seconds(10))
// create a DStream that will connect to serverIP:serverPort
val lines = ssc.socketTextStream(serverIP, serverPort)
// split each line into words
val words = lines.flatMap(_.split(" "))
// count each word in each batch
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey(_ + _)
```

Que es Spark Streaming

- Flujo básico de programa – Scala (http://spark.apache.org/docs/latest/streaming-programming-guide.html)

4. Arrancar el StreamingContext para comenzar a recibir y procesar datos

```
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
// create a StreamingContext
val ssc = new StreamingContext(sc, Seconds(10))
// create a DStream that will connect to serverIP:serverPort
val lines = ssc.socketTextStream(serverIP, serverPort)
// split each line into words
val words = lines.flatMap(_.split(" "))

// count each word in each batch
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey(_ + _)
// print a few of the counts to the console
wordCounts.print()

ssc.start()             // start the computation
```

Que es Spark Streaming

- Flujo básico de programa – Scala (http://spark.apache.org/docs/latest/streaming-programming-guide.html)

4. Esperar a la finalización del procesamiento con StreamingContext.awaitTermination()

```
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
// create a StreamingContext
val ssc = new StreamingContext(sc, Seconds(10))
// create a DStream that will connect to serverIP:serverPort
val lines = ssc.socketTextStream(serverIP, serverPort)
// split each line into words
val words = lines.flatMap(_.split(" "))

// count each word in each batch
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey(_ + _)
// print a few of the counts to the console
wordCounts.print()

ssc.start()             // start the computation
ssc.awaitTermination()  // wait for the computation to terminate
```

Que es Spark Streaming

- Limitacions del Dstream
  - Dificultad para trabajar con event-time y late-data
    - Spark Streaming trabaja con tiempo de micro-batch
  - Interoperabilidad limitada entre procesamiento batch y streaming
    - RDD y Dstream son APIs similares pero no equivalentes
  - Las garantías end-to-end quedan a discreción del desarrollador

Que es Spark Streaming

- Streams + Batch
  - Spark Streaming permite combinar Streams en tiempo real con datos históricos
    - Generea modelos históricos de datos
    - Usa los modelos para procesar stream de datos en tiempo real
  - Spark Streaming se integra con Mlib, GraphX, …
    - Aprendizaje por lotes, predicción en tiempo real
    - Aprendizaje y predicción en tiempo real
  - Consultas interactivas sobre Streams con SQL
    - Select * from table_from_streaming_data

Que es Spark Streaming

- Ventajas de una interfaz unificada
  - Explora los datos interactivamente para encontrar problemas
  - Utilizar el mismo código Spark para procesar logs masivos
  - Aplica el mismo código en Spark Streaming para procesamiento en tiempo real

```
$ ./spark--shell
scala> val file = sc.hadoopFile("smallLogs")
...
scala> val filtered = file.filter(_.contains("ERROR"))
...
scala> val mapped = filtered.map(...)
```

```
object ProcessProductionData {
  def main(args: Array[String]) {
    val sc = new SparkContext(...)
    val file = sc.hadoopFile("productionLogs")
    val filtered = file.filter(_.contains("ERROR"))
    val mapped = filtered.map(...)
    ...
  }
}
```

```
object ProcessLiveStream {
  def main(args: Array[String]) {
    val sc = new StreamingContext(...)
    val stream = KafkaUtil.createStream(...)
    val filtered = stream.filter(_.contains("ERROR"))
    val mapped = filtered.map(...)
    ...
  }
}
```