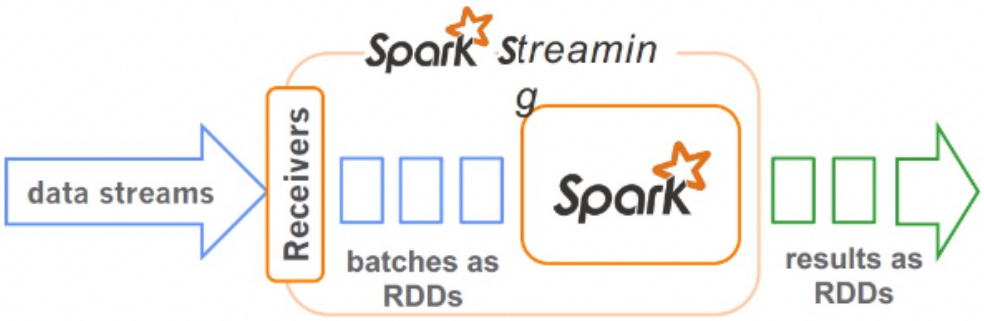
Spark Streaming

Spark Streaming

- Scalable, fault-tolerant stream processing system.
- Receive data streams from input sources, process them in a cluster, push out to databases/dashboards.

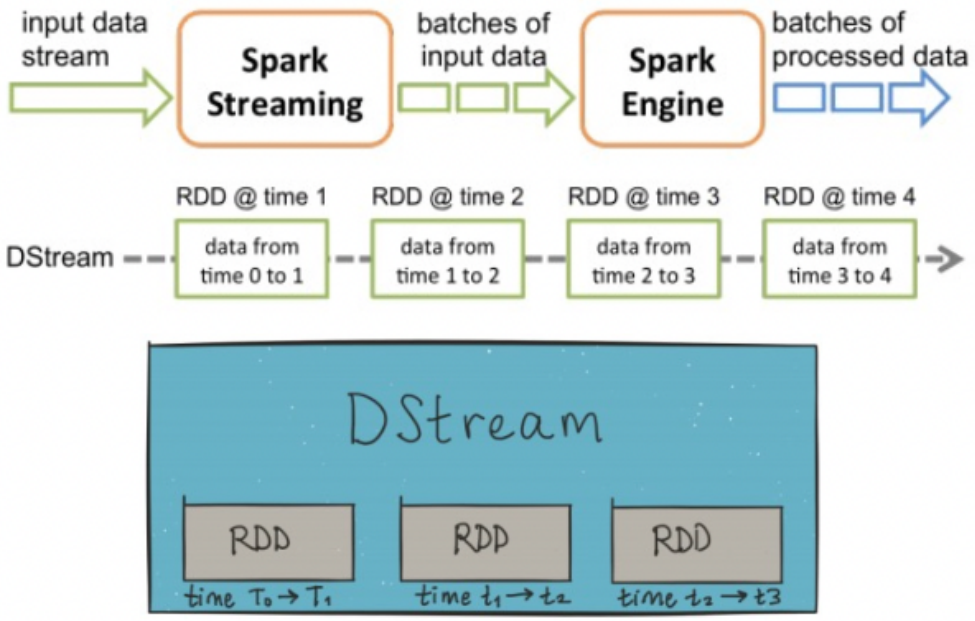
How does it work?

- The stream is treated as a series of very small, deterministic batches of data
- Spark treats each batch of data as RDDs and processes them using RDD operations
- Processed results are pushed out in batches

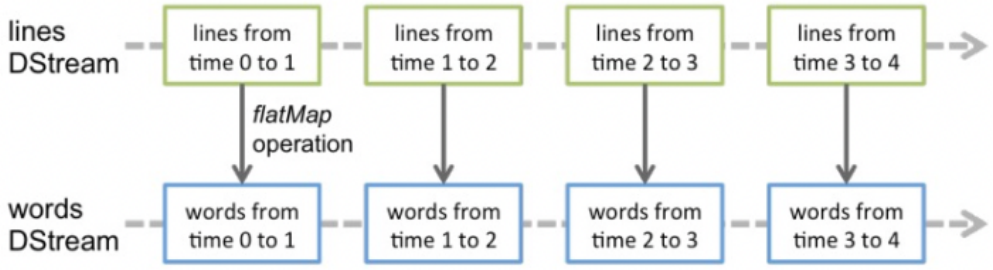


Discretized Stream (DStream)

- Sequence of RDDs representing a stream of data



- Any operation applied on a DStream translates to operations on the underlying RDDs



Streaming Context

- The main entry point of all Spark Streaming functionality

```
val conf = new
SparkConf().setAppName(appName).setMaster(master)
val ssc = new StreamingContext(conf, batchinterval)
```

- appname: name of the application
- master: a Spark, Mesos, or YARN cluster URL
- batchinternval: time interval (in second) of each batch

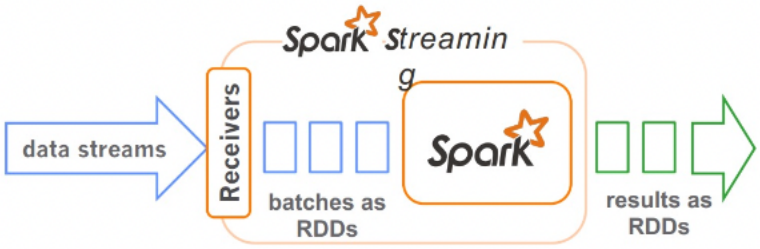
Operation on DStreams

Three categories

- Input operation.
- Transformation operation.
- Output operation.

Input operations

- Every input DStream is associated with a Receiver object
- Two built-in categories of streaming sources:
  - Basic sources, e.g., file systems, socket connection
  - Advanced sources, e.g., Twitter, Kafka



- Basic sources
  - Socket connection

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

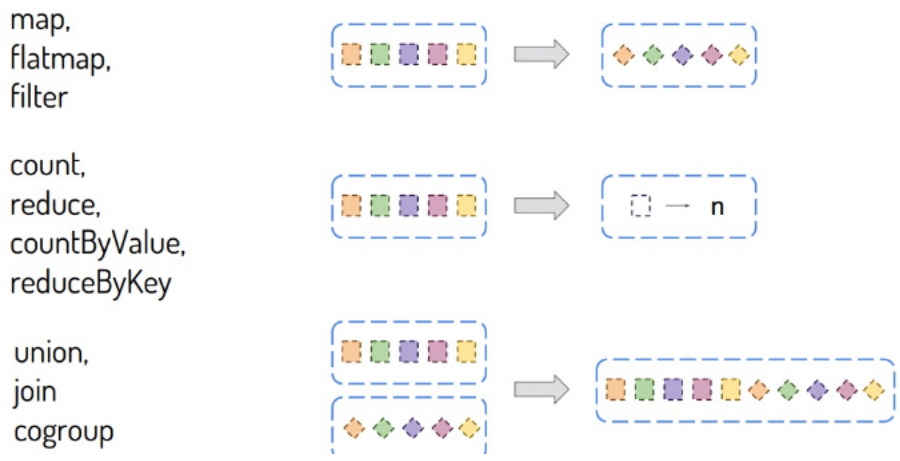
  - File stream

```
streamingContext.fileStream[…](dataDirectory)
```

- Advanced sources

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

Transformation



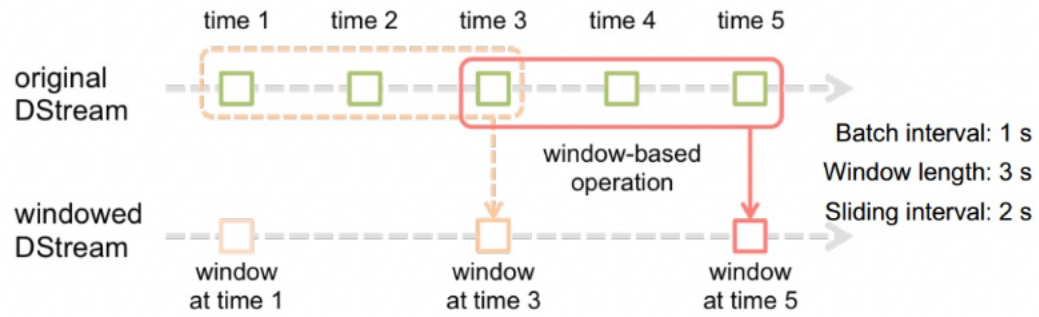
Transformation	Meaning
map (func)	Return a new DStream by passing each element of the source DStream through a function func
flatmap(func)	Similar to map, but each input item can be mapped to 0 or more output items
filter(func)	Return a new DStream by selecting only the records of the source DStream on which func returns true

Transformation	Meaning
count	Return a new DStream of single-element RDDs by counting the number of elements in each RDD of the source DStream
countbyValue	Returns a new DStream of (K, Long) pairs where the value of each key is its frequency in each RDD of the source DStream.
reduce(func)	Return a new DStream of single-element RDDs by aggregating the elements in each RDD of the source DStream using a function func (which takes two arguments and returns one).
reducebyKey(func)	When called on a DStream of (K, V) pairs, return a new DStream of (K, V) pairs where the values for each key are aggregated using the given reduce function

Transformation	Meaning
union(otherStream)	Return a new DStream that contains the union of the elements in the source DStream and otherDStream.
join(otherStream)	When called on two DStreams of (K, V) and (K, W) pairs, return a new DStream of (K, (V, W)) pairs with all pairs of elements for each key.

Window Operations

- Spark provides a set of transformations that apply to a sliding window of data
- A window is defined by: window length and siding interval



- *window(windowLength, slideInterval)*
  - Returns a new DStream which is computed based on windowed batches

- *countByWindow(windowLength, slideInterval)*
  - Returns a sliding window count of elements in the stream.

- *reduceByWindow(func, windowLength, slideInterval)*
  - Returns a new single-element DStream, created by aggregating elements in the stream over a sliding interval using func.

Output Operation

- Push out DStream's data to external systems, e.g., a database or a file system

Operation	Meaning
print	Prints the first ten elements of every batch of data in a DStream on the driver node running the application
saveAsTextFiles	Save this DStream's contents as text files
saveAsHadoopFiles	Save this DStream's contents as Hadoop files.
foreachRDD(func)	Applies a function, func, to each RDD generated from the stream

Example

Word Count

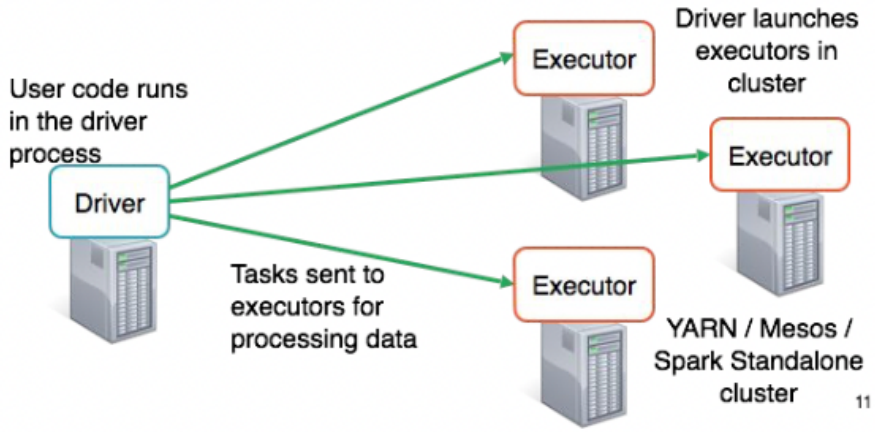
```
val context = new StreamingContext(conf, Seconds(1))
val lines = context.socketTextStream(...)
val words = lines.flatMap(_.split(" "))
val wordCounts = words.map(x => (x, 1)).reduceByKey(_+_)
wordCounts.print()
context.start()
```

Print the DStream contents on screen

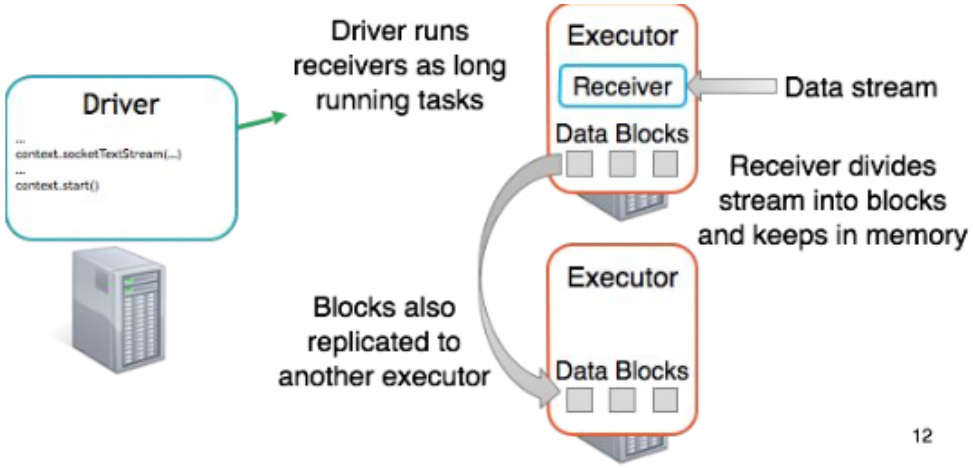
Start the streaming job

Lifecycle of a streaming app

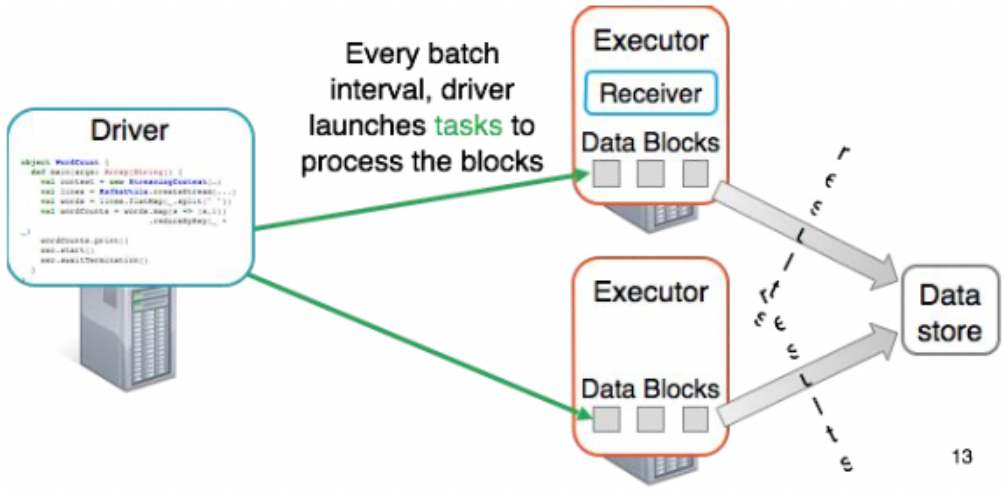
Execution in any Spark Application



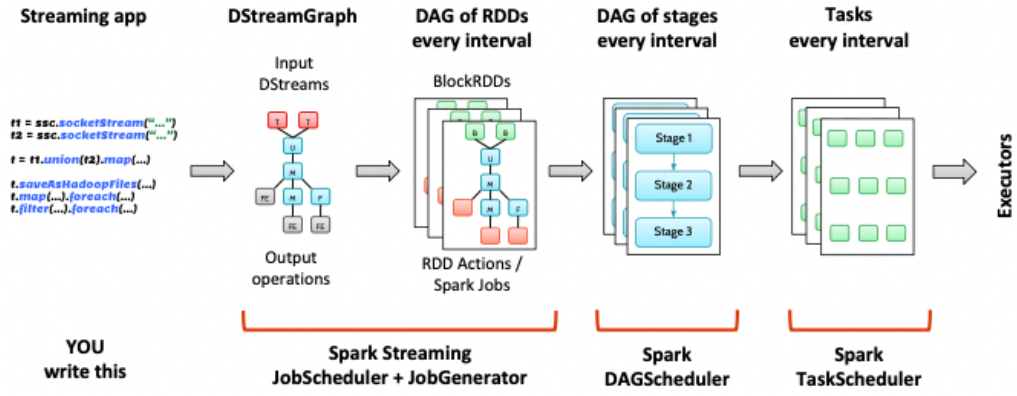
Receiving Data



Processing Data



End-to-end view



Dynamic Load Balancing

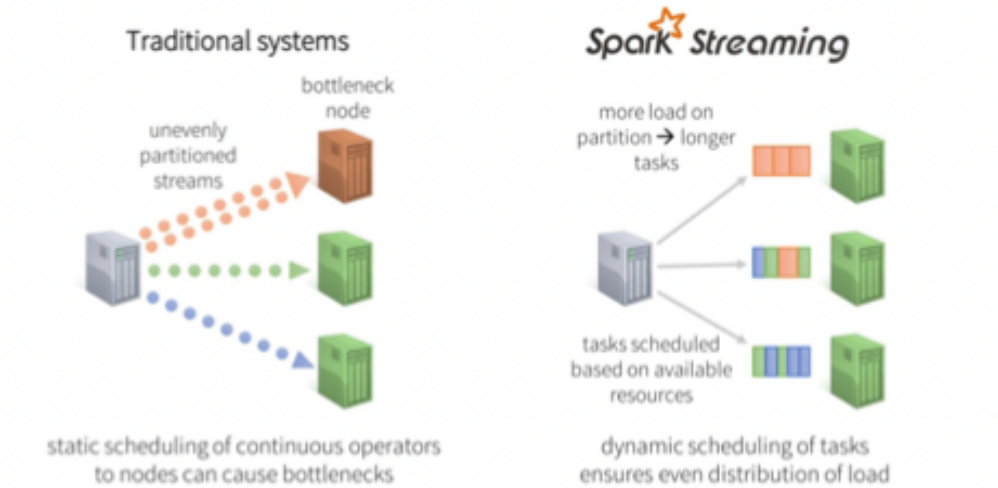


Figure 3: Dynamic load balancing

Fast failure and Straggler recovery

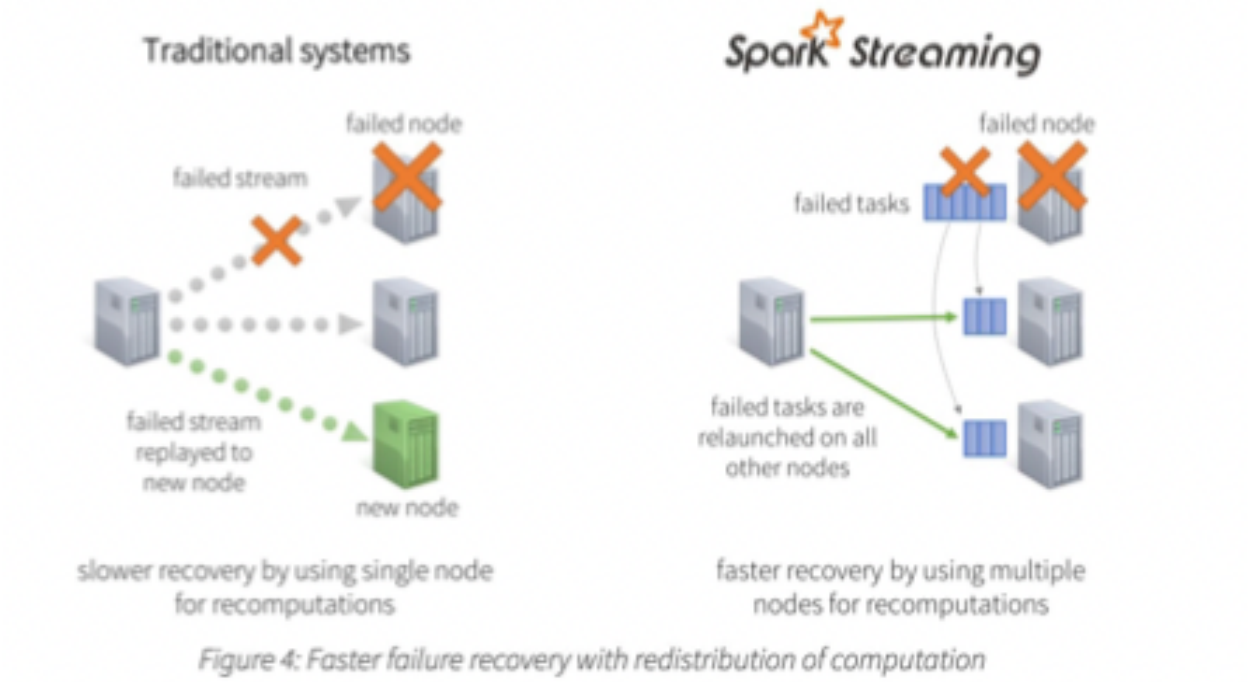


Figure 4: Faster failure recovery with redistribution of computation

Table of Contents

Spark Streaming ***1***

How does it work? ***1***

Discretized Stream (DStream) ***2***

Streaming Context ***3***

Operation on DStreams ***3***

Three categories ***3***

Input operations ***3***

Transformation ***4***

Window Operations ***5***

Output Operation ***6***

Example ***7***

Lifecycle of a streaming app ***7***

Execution in any Spark Application ***7***

Receiving Data ***7***

Processing Data ***8***

End-to-end view ***8***

Dynamic Load Balancing ***8***

Fast failure and Straggler recovery ***9***