



UNIVERSITAT DE
BARCELONA

Master in Fundamental Principles of Data Science

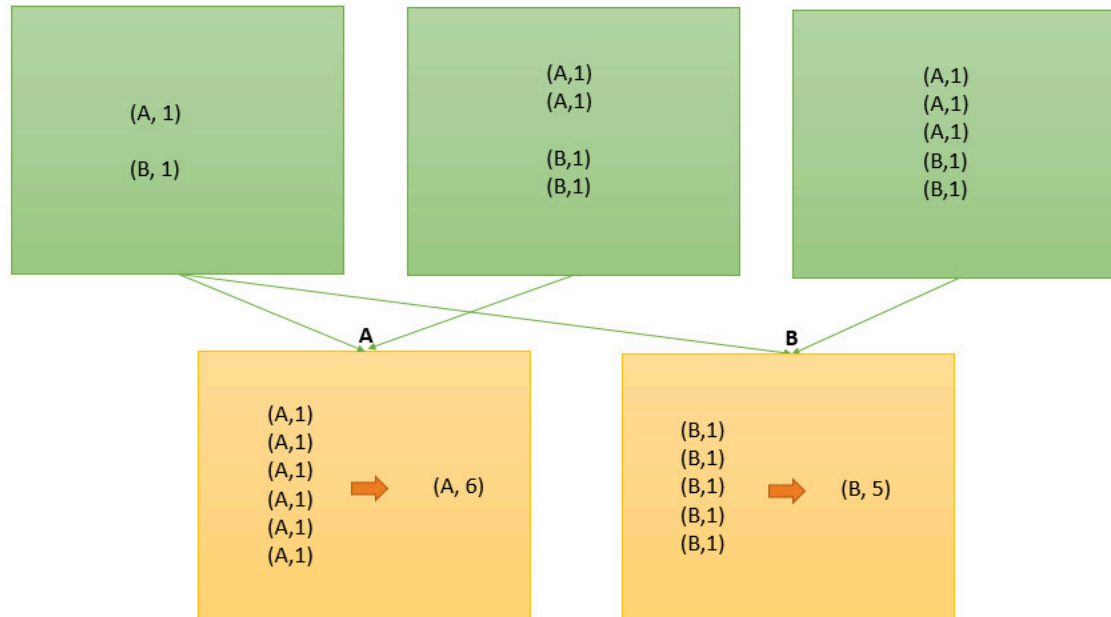
Dr Rohit Kumar



combineByKey vs groupByKey vs reduceByKey

groupByKey

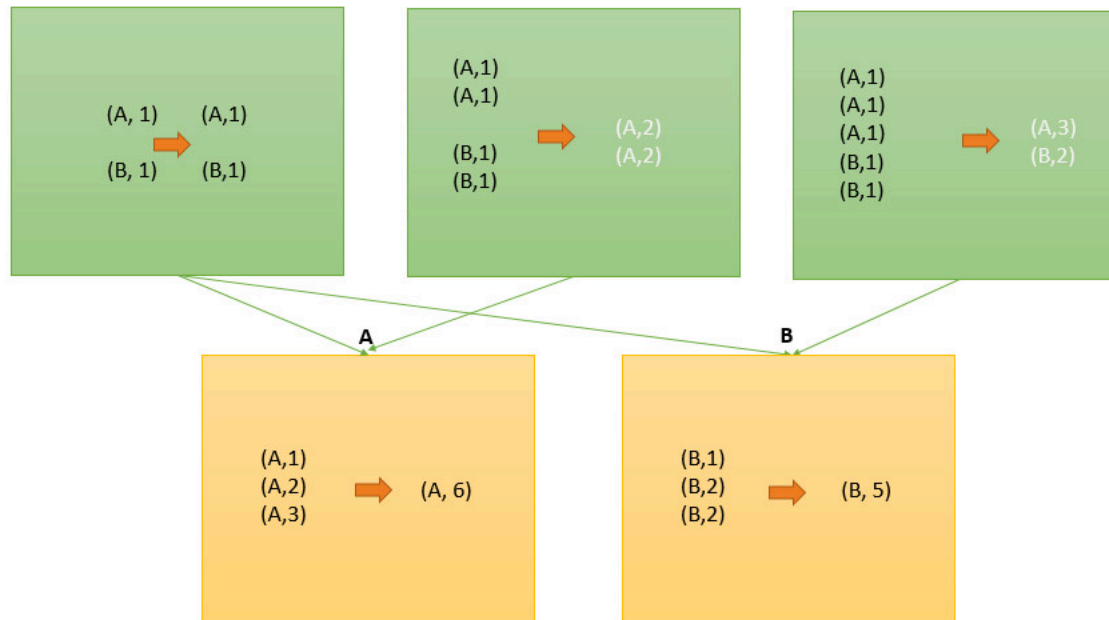
Group the values for each key in the RDD into a single sequence.



<http://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD.groupByKey>

reduceByKey

Merge the values for each key using an associative and commutative reduce function.





combineByKey

reduceByKey	CombineByKey
reduceByKey internally calls combineByKey	CombineByKey is the generic api and is used by reduceByKey and aggregateByKey
the input type and outputType of reduceByKey are the same	CombineByKey is more flexible, hence one can mention the required outputType .

```
rdd.reduceByKey(func)
```

Is actually this

```
rdd.combineByKey(lambda x: x, func, func)
```

combineByKey

- Takes three functions
 - createCombiner, which turns a V into a C (e.g., creates a one-element list)
 - mergeValue, to merge a V into a C (e.g., adds it to the end of a list)
 - mergeCombiners, to combine two C 's into a single one (e.g., merges the lists)



Window Function vs Group by in Spark SQL

Group BY

- Takes three functions
 - createCombiner, which turns a V into a C (e.g., creates a one-element list)
 - mergeValue, to merge a V into a C (e.g., adds it to the end of a list)
 - mergeCombiners, to combine two C 's into a single one (e.g., merges the lists)

Window Function

Window functions operate on a set of rows and return a single value for each row from the underlying query. The term **window** describes the set of rows on which the **function** operates.

It is a popular concept in SQL world which was introduced in Spark SQL in 1.4 version onwards

<https://databricks.com/blog/2015/07/15/introducing-window-functions-in-spark-sql.html>

Window Functions

Spark SQL supports three kinds of window functions:

- ranking functions
- analytic functions
- aggregate functions



Window Function

	SQL	DataFrame API
Ranking functions	rank	rank
	dense_rank	denseRank
	percent_rank	percentRank
	ntile	ntile
	row_number	rowNumber
Analytic functions	cume_dist	cumeDist
	first_value	firstValue
	last_value	lastValue
	lag	lag
	lead	lead

Spark Submit

<https://spark.apache.org/docs/latest/configuration.html>

There are many Spark configuration parameters please check them out

Spark provides three locations to configure the system:

- Spark properties control most application parameters and can be set by using a SparkConf object, or through Java system properties.
- Environment variables can be used to set per-machine settings, such as the IP address, through the conf/spark-env.sh script on each node.
- Logging can be configured through log4j.properties.



Multiple Notebook Options for Spark

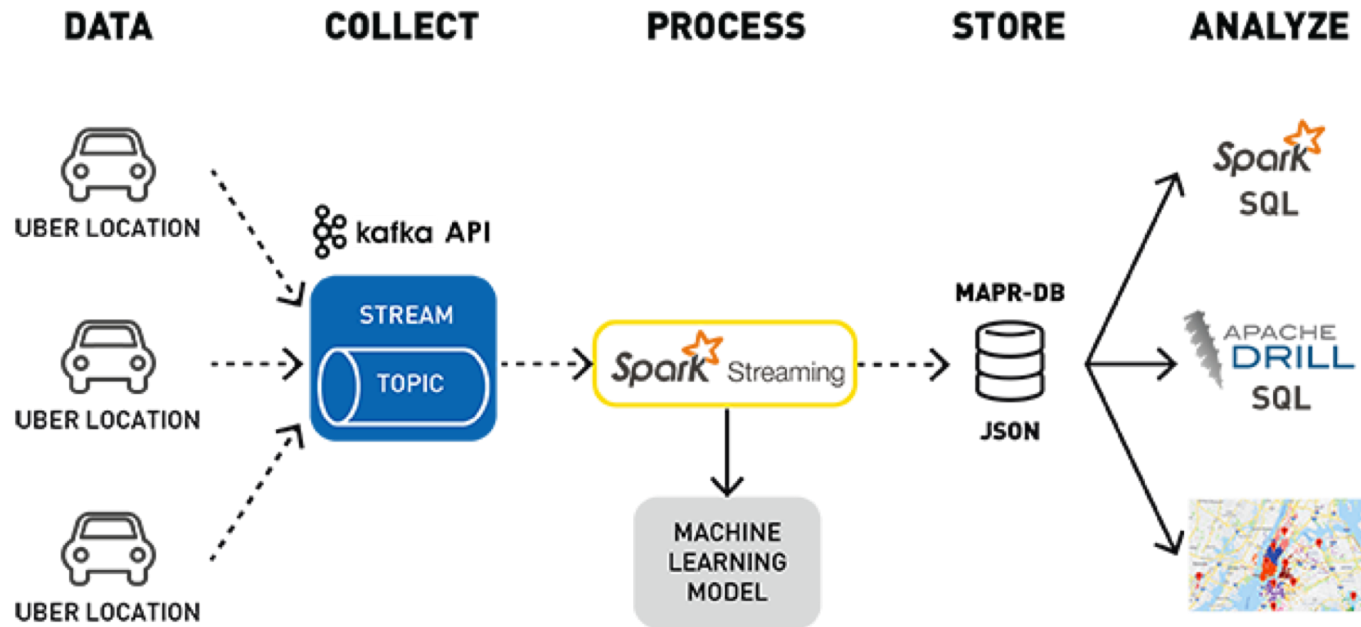
- Data Bricks Notebook
- Zeppelin Notebooks
- Spark Magic



UNIVERSITAT DE
BARCELONA

Spark Streaming

Popular Uber use case



The application of machine learning to geolocation data is being used in telecom, travel, marketing, and manufacturing to identify patterns and trends for services such as recommendations, anomaly detection, and fraud.



Stream Processing Programming Models

Continuous – Real Time



samza



Micro Batch – Near Real Time



Spark Streaming



Example [NetworkWordCount](#).

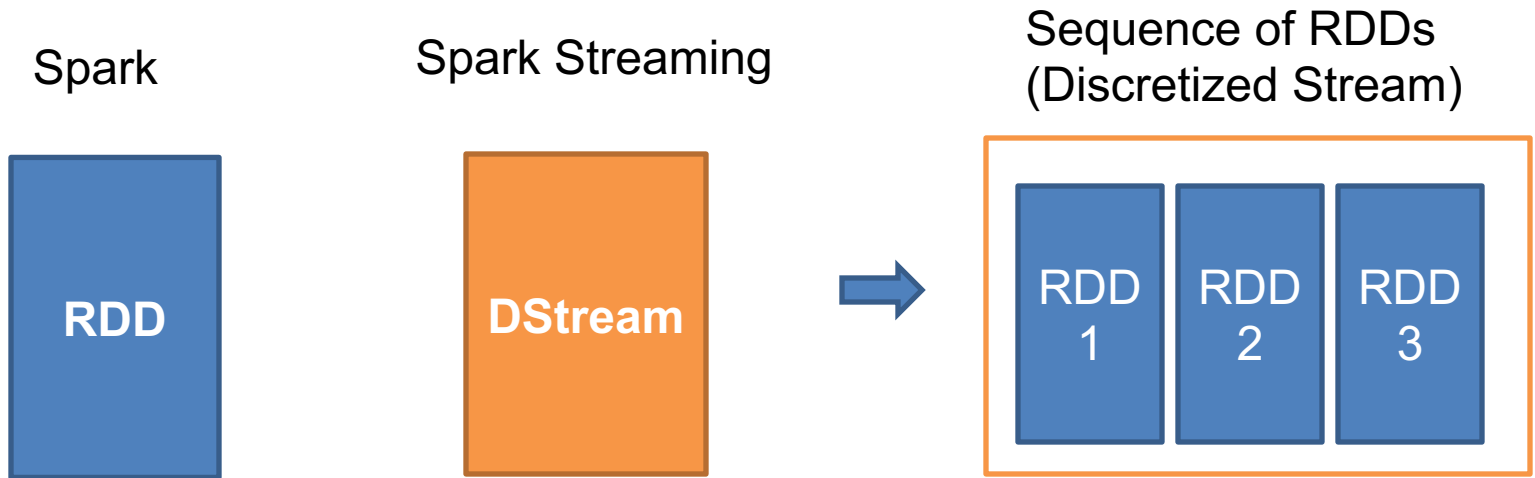
Streaming Setup

A `StreamingContext` object can be created from a `SparkContext` object.

After a context is defined, you have to do the following.

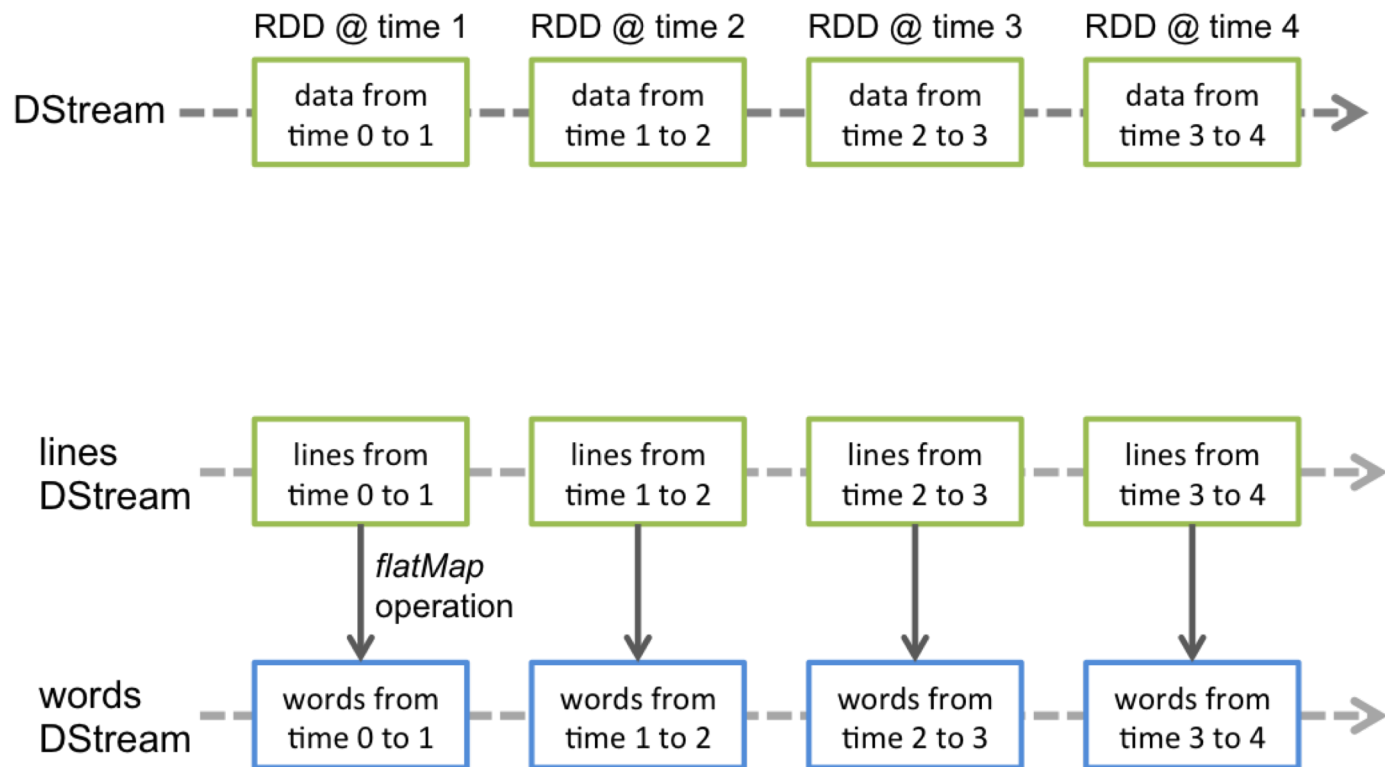
1. Define the input sources by creating input `DStreams`.
2. Define the streaming computations by applying transformation and output operations to `DStreams`.
3. Start receiving data and processing it using `streamingContext.start()`.
4. Wait for the processing to be stopped (manually or due to any error) using `streamingContext.awaitTermination()`.
5. The processing can be manually stopped using `streamingContext.stop()`.

Building Block





DStream





Input DStreams and Receivers

Spark Streaming provides two categories of built-in streaming sources.

- *Basic sources*: Sources directly available in the StreamingContext API. Examples: file systems, and socket connections.
- *Advanced sources*: Sources like Kafka, Flume, Kinesis, etc. are available through extra utility classes.

Points to remember

- When running a Spark Streaming program locally, do not use “local” or “local[1]” as the master URL. Either of these means that only one thread will be used for running tasks locally. If you are using an input DStream based on a receiver (e.g. sockets, Kafka, Flume, etc.), then the single thread will be used to run the receiver, leaving no thread for processing the received data. Hence, when running locally, always use “local[n]” as the master URL, where $n > \text{number of receivers}$ to run.
- Extending the logic to running on a cluster, the number of cores allocated to the Spark Streaming application must be more than the number of receivers. Otherwise the system will receive data, but not be able to process it.

DStream Transformations

- All Basic transformations are supported
 - map
 - flatMap
 - filter
 - union
 - reduceByKey
 - .
 - .



Stream Specific Transformation

The **updateStateByKey** operation allows you to maintain arbitrary state while continuously updating it with new information. To use this, you will have to do two steps.

- Define the state - The state can be an arbitrary data type.
- Define the state update function - Specify with a function how to update the state using the previous state and the new values from an input stream.

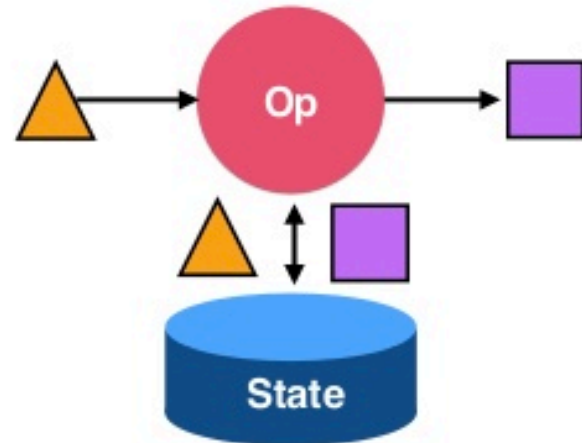
Example: [stateful network wordcount.py](#)

Stateful vs Stateless

Stateless Stream
Processing

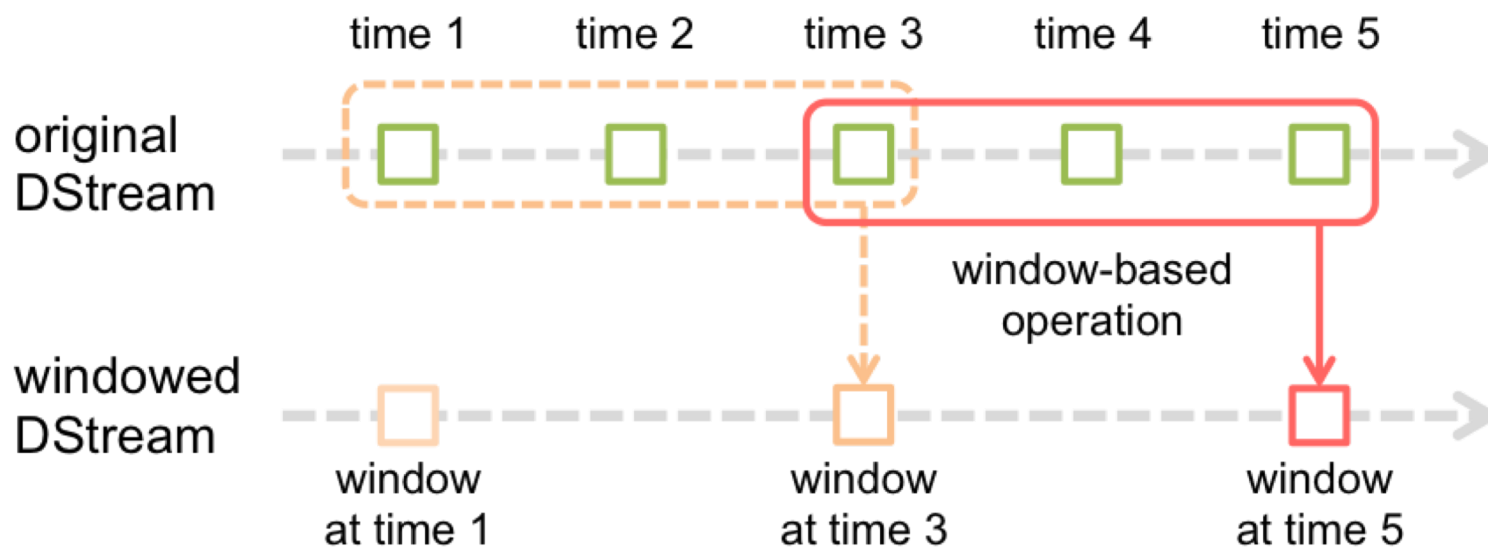


Stateful Stream
Processing



Window Operations

Again a popular term in Stream Computation



Window definition

Window operation needs to specify two parameters.

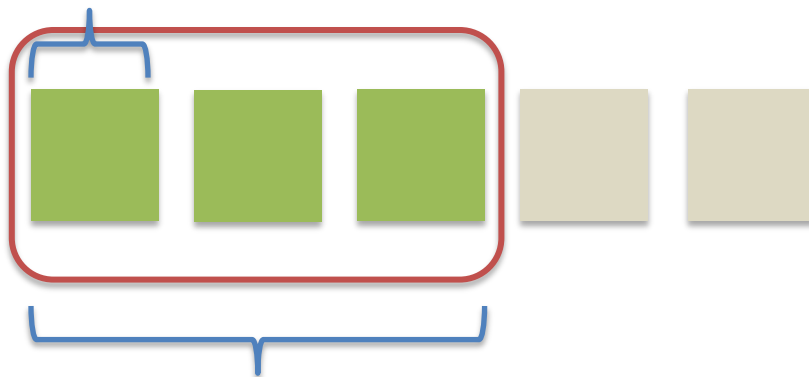
- *window length* - The duration of the window (3 in the previous figure).
- *sliding interval* - The interval at which the window operation is performed (2 in the previous figure).

These two parameters must be multiples of the batch interval of the source DStream (1 in the previous figure).

Window

1st Window

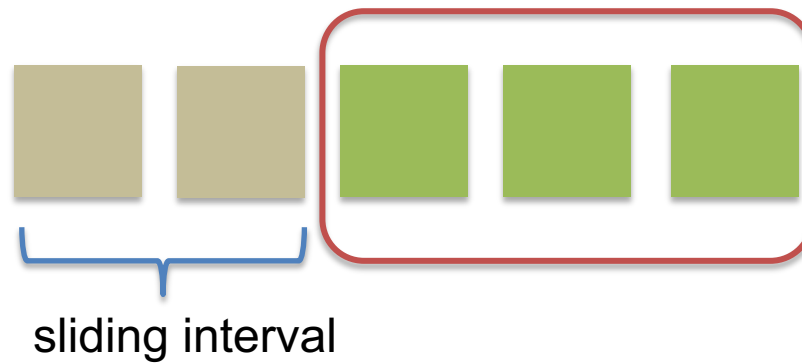
Batch interval



Window interval

Window

2nd Window



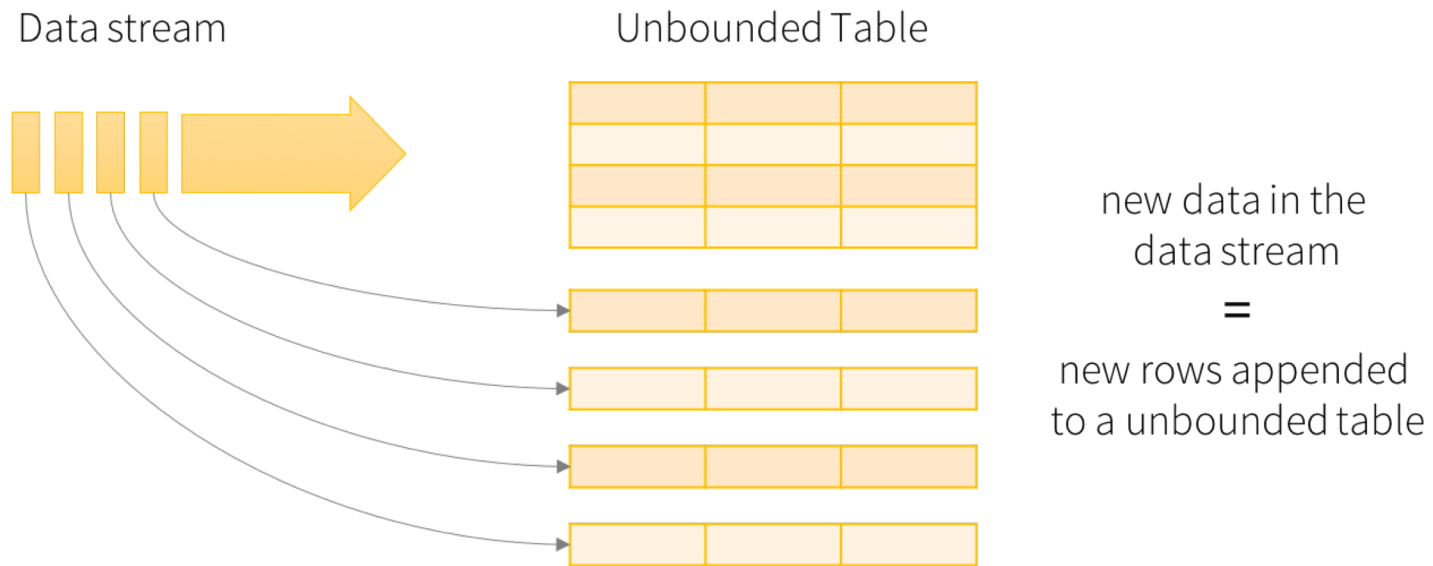
<https://spark.apache.org/docs/latest/streaming-programming-guide.html#transformations-on-dstreams>



UNIVERSITAT DE
BARCELONA

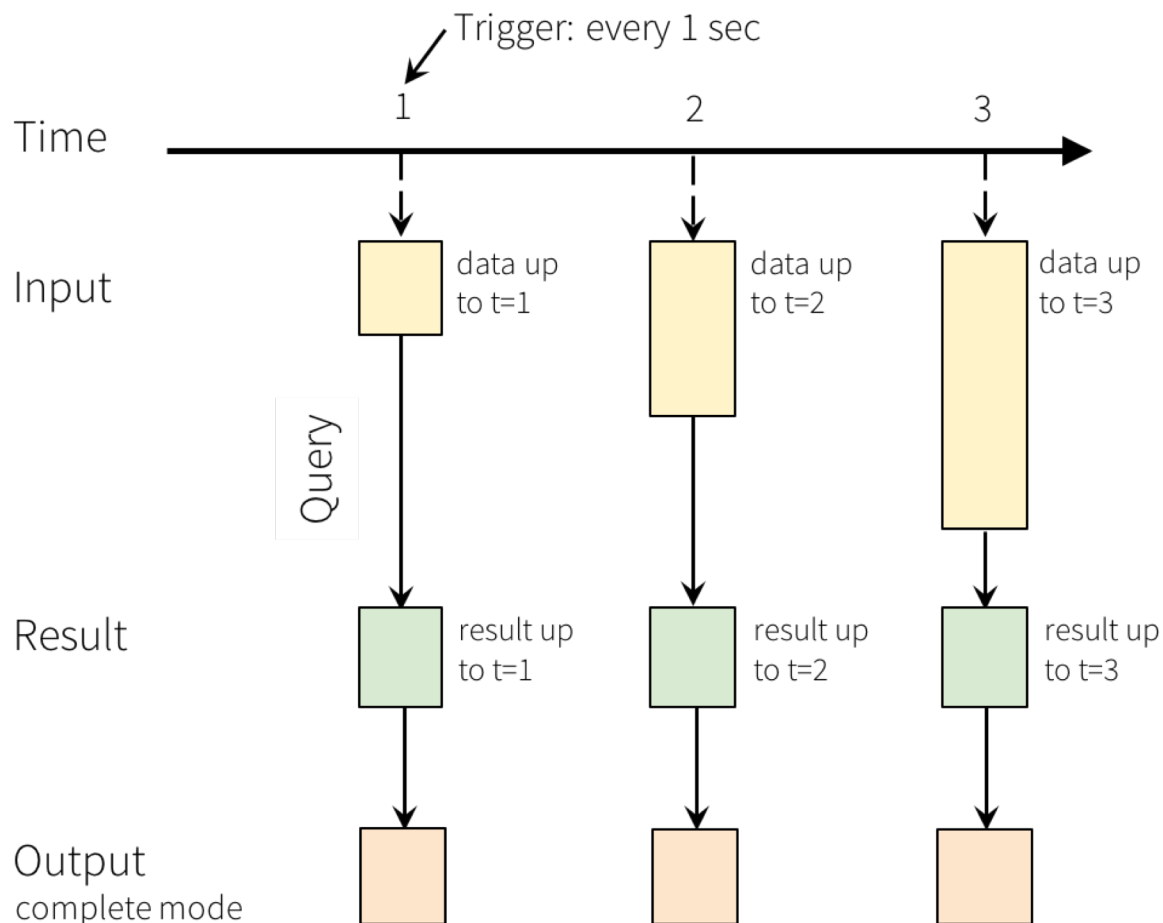
Structured Streaming

Basic Idea



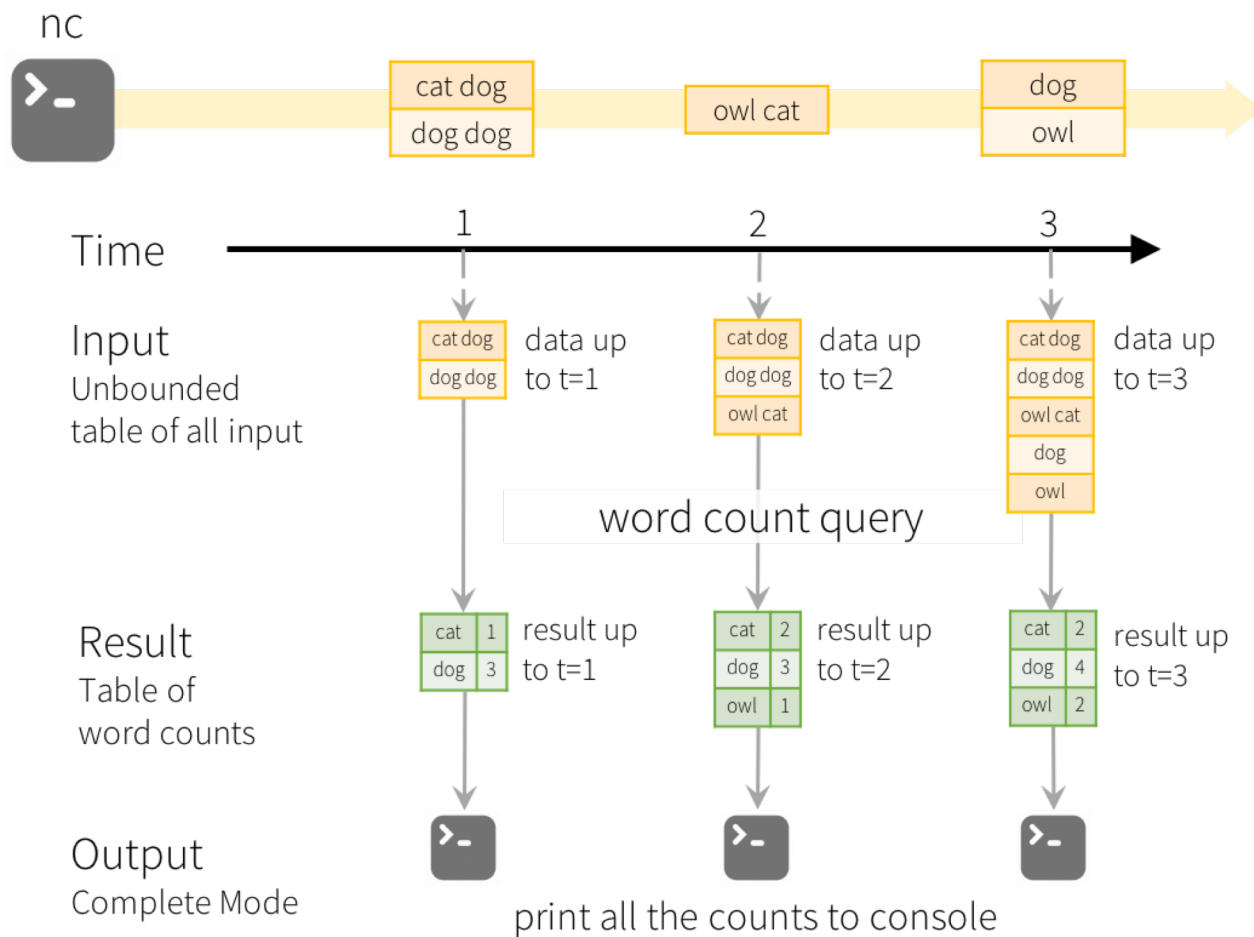
Data stream as an unbounded table

Behind the scene



Programming Model for Structured Streaming

Behind the scene



Model of the Quick Example

Different output Mode

- ***Complete Mode*** - The entire updated Result Table will be written to the external storage. It is up to the storage connector to decide how to handle writing of the entire table.
- ***Append Mode*** - Only the new rows appended in the Result Table since the last trigger will be written to the external storage. This is applicable only on the queries where existing rows in the Result Table are not expected to change.
- ***Update Mode*** - Only the rows that were updated in the Result Table since the last trigger will be written to the external storage (available since Spark 2.1.1).

References

- <https://www.youtube.com/watch?v=g171ndOHgJ0&feature=youtu.be>
- <https://databricks.com/blog/2015/07/15/introducing-window-functions-in-spark-sql.html>
- <https://spark.apache.org/docs/latest/streaming-programming-guide.html>
- <https://www.youtube.com/watch?v=UuRhEmqqhRM>