UNIVERSITAT DE BARCELONA

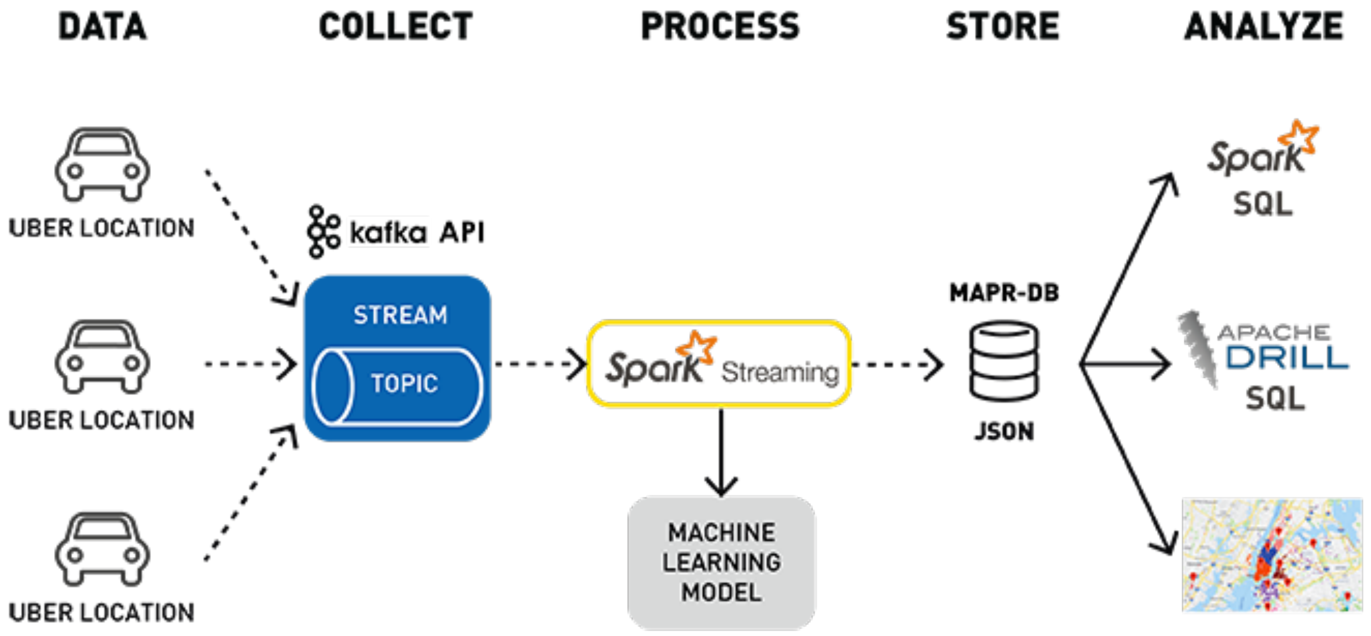
Master in Fundamental Principles of Data Science

Dr Rohit Kumar

Spark Streaming

UNIVERSITAT DE BARCELONA

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.

UNIVERSITAT DE BARCELONA

Stream Processing Programing Models

Continuous – Real Time

APACHE STORM™
Distributed • Resilient • Real-time

samza

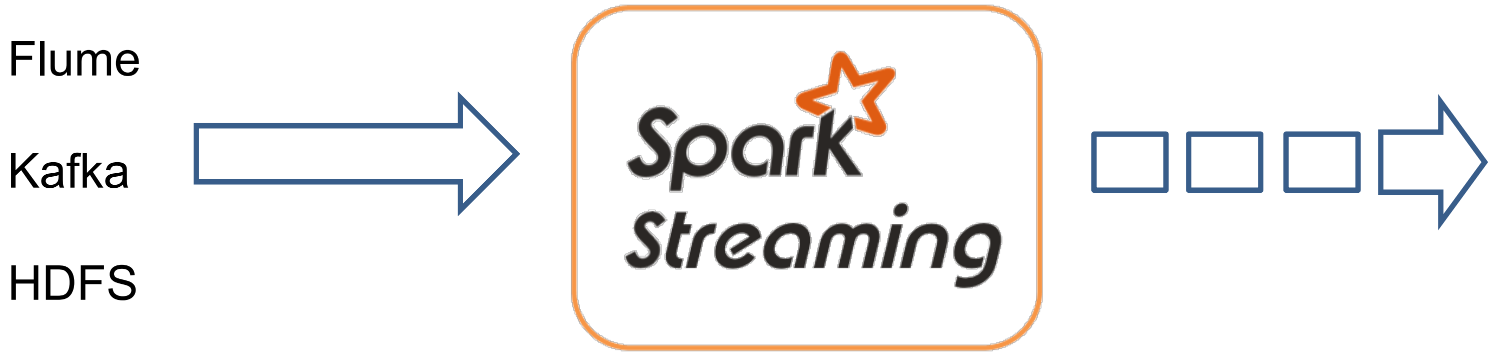
Flink

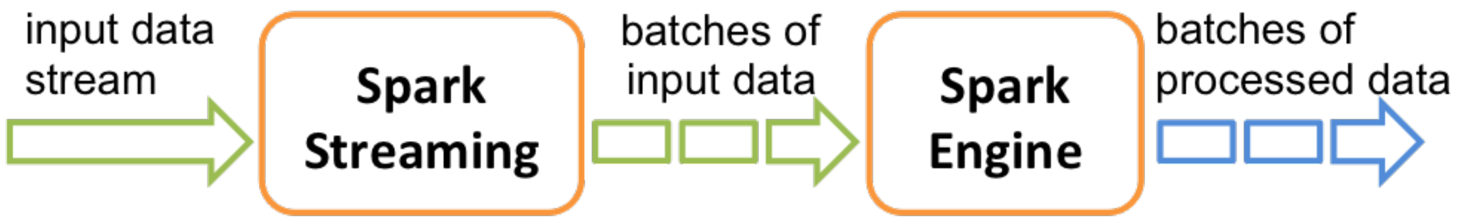
Micro Batch – Near Real Time

Spark Streaming

Flink

Spark Streaming





Example NetworkWordCount.

UNIVERSITAT DE BARCELONA

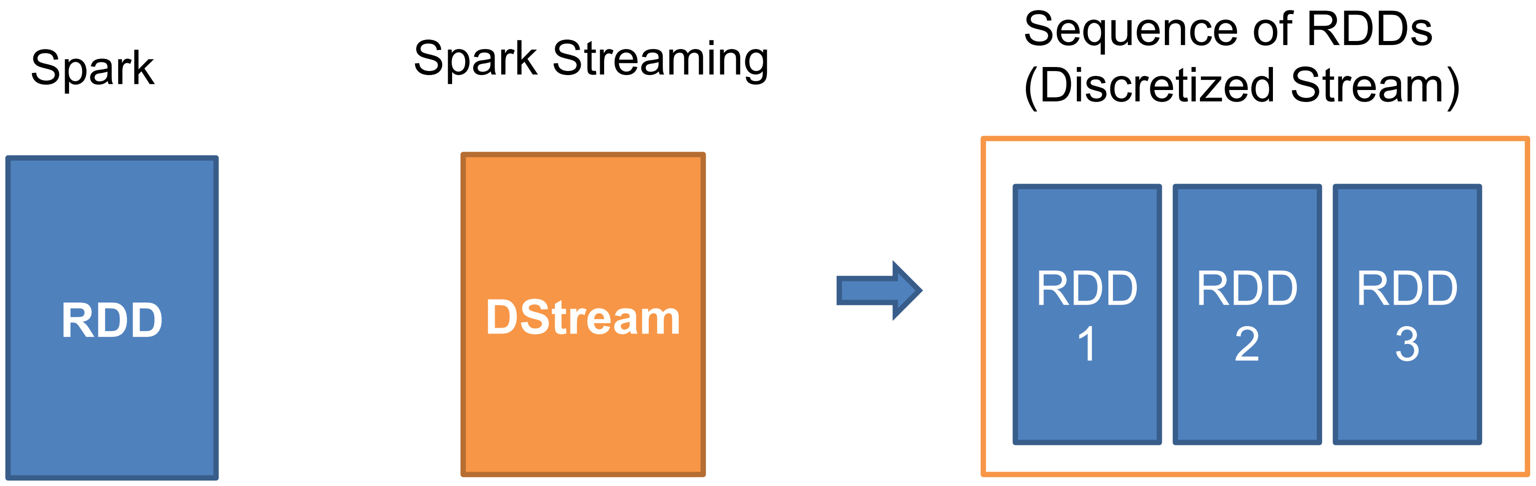
Streaming Setup

A StreamingContext object can be created from a SparkContext object.

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

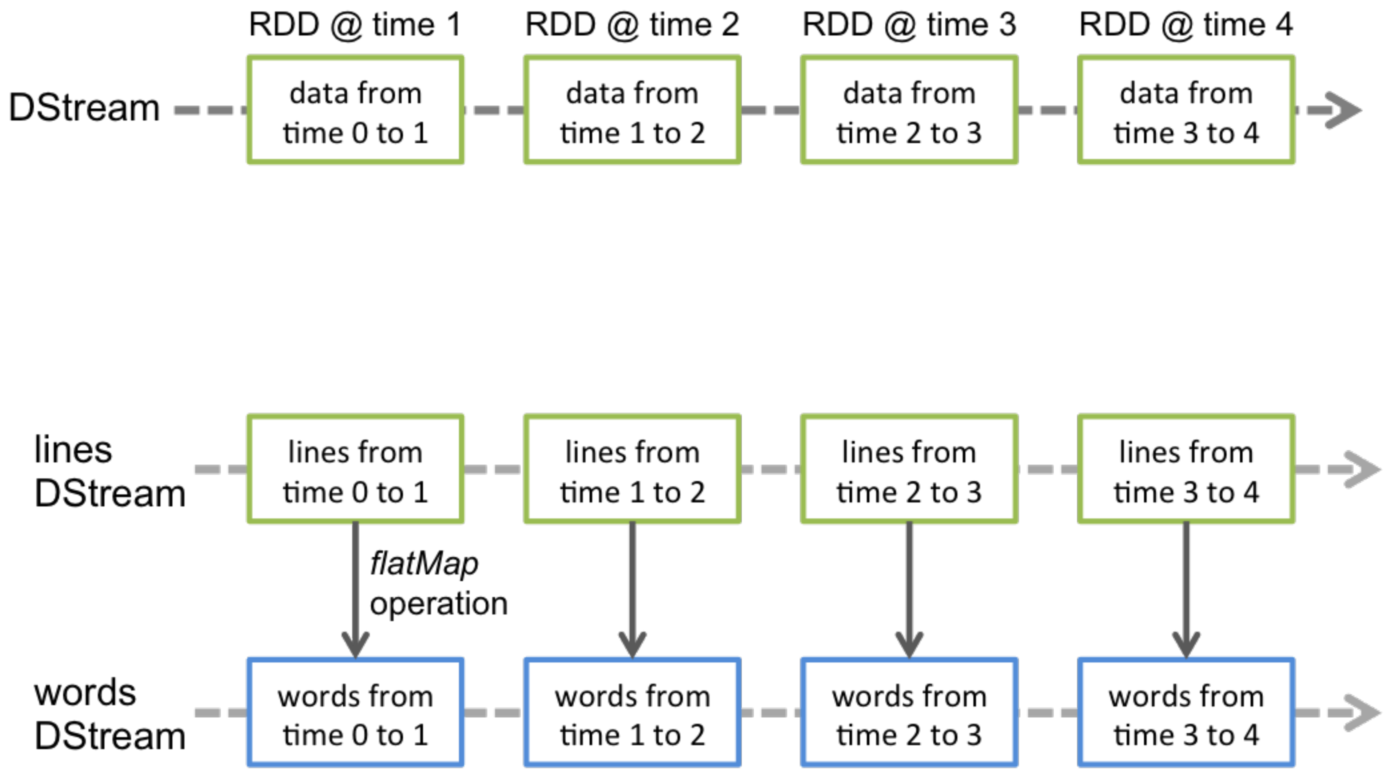
1. Define the input sources by creating input DStreams.
2. Define the streaming computations by applying transformation and output operations to DStreams.
3. Start receiving data and processing it using streamingContext.start().
4. Wait for the processing to be stopped (manually or due to any error)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

Building Block



UNIVERSITAT DE BARCELONA

DStream



Important Note:

- Once a context has been started, no new streaming computations can be set up or added to it.
- Once a context has been stopped, it cannot be restarted.
- Only one StreamingContext can be active in a JVM at the same time.
- stop() on StreamingContext also stops the SparkContext. To stop only the StreamingContext, set the optional parameter of stop() called stopSparkContext to false.
- A SparkContext can be re-used to create multiple StreamingContexts, as long as the previous StreamingContext is stopped (without stopping the SparkContext) before the next StreamingContext is created

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 > number of receivers to run.

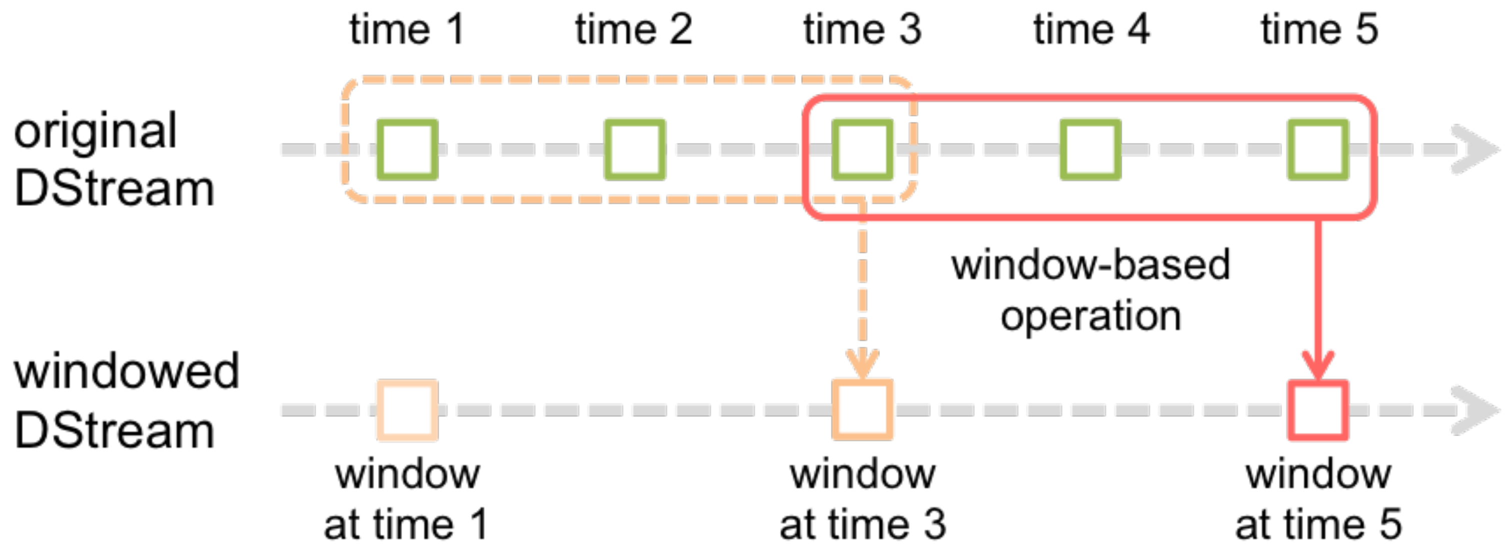
- Extendng the logic to running on a cluster, the number of cores allocated to the Spark Streaming application must be more than the number of receivers. Otherwise the system will receive data, but not be able to process it.

DStream Transfromations

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

Window Operations

Again a popular term in Stream Computation



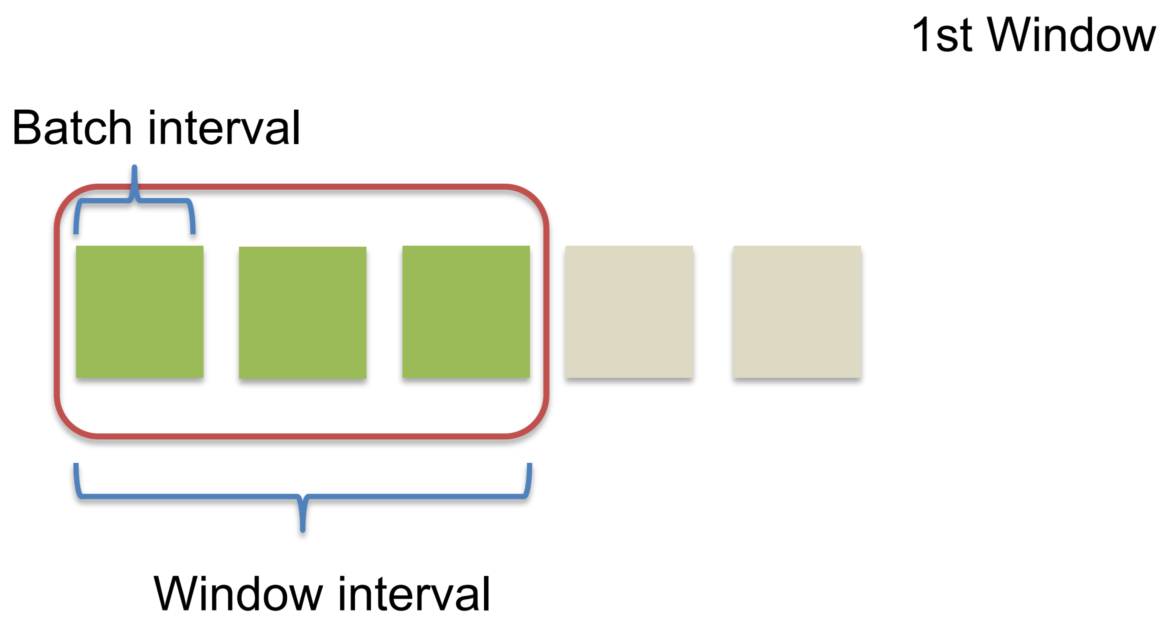
Window definition

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).

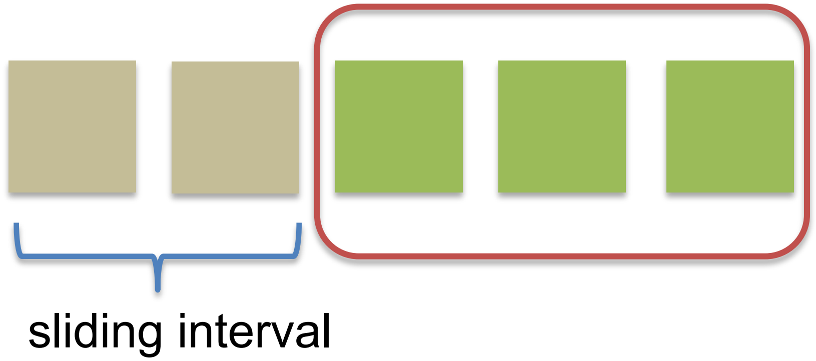
These two parameters must be multiples of the batch interval of the source DStream (1 in the previous figure).

Window



Window

2nd Window



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

Stream Specific Transformation

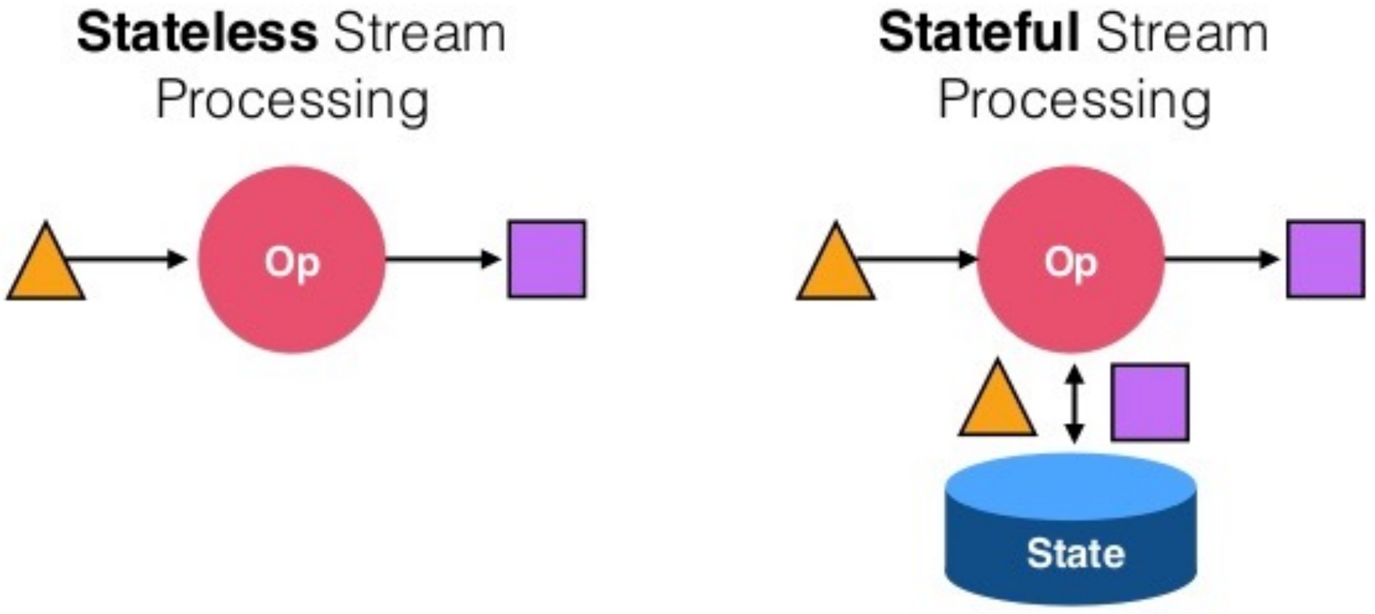
The **updateStateByKey** operation allows you to maintain arbitrary state while continuously updating it with new information. To use this, you will have to do two steps.

- 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



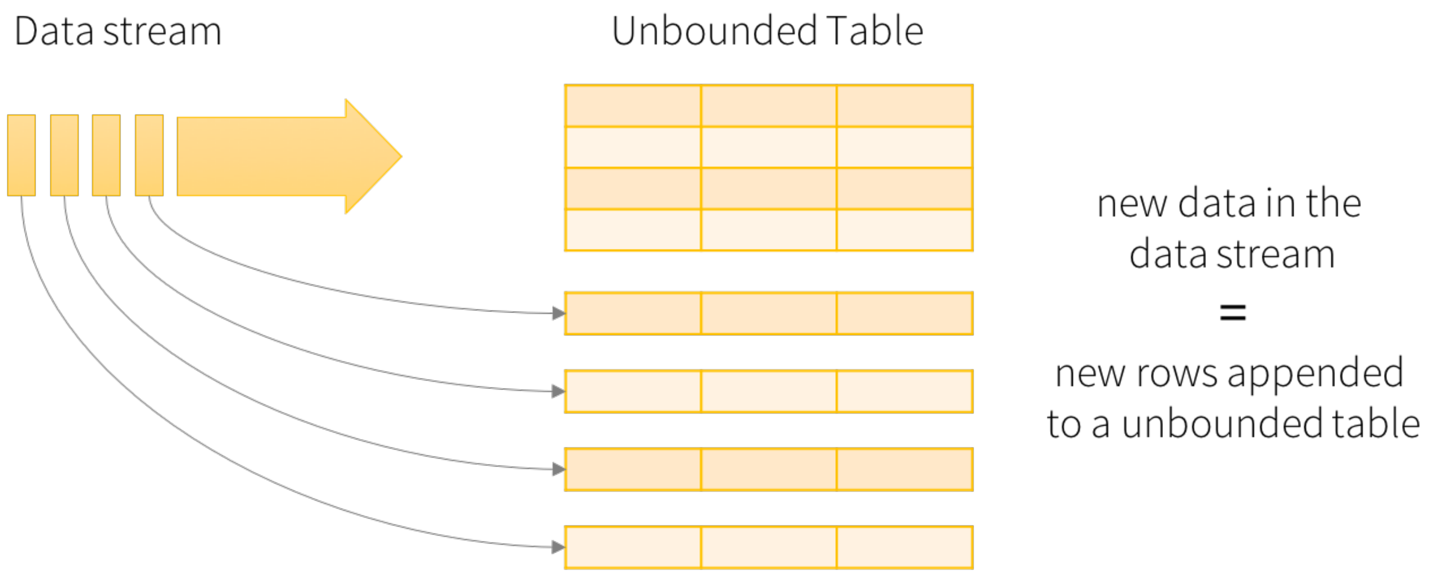
Checkpointing

- *Usage of stateful transformations* - If either updateStateByKey or reduceByKeyAndWindow (with inverse function) is used in the application, then the checkpoint directory must be provided to allow for periodic RDD checkpointing.
- *Recovering from failures of the driver running the application*.

UNIVERSITAT DE BARCELONA

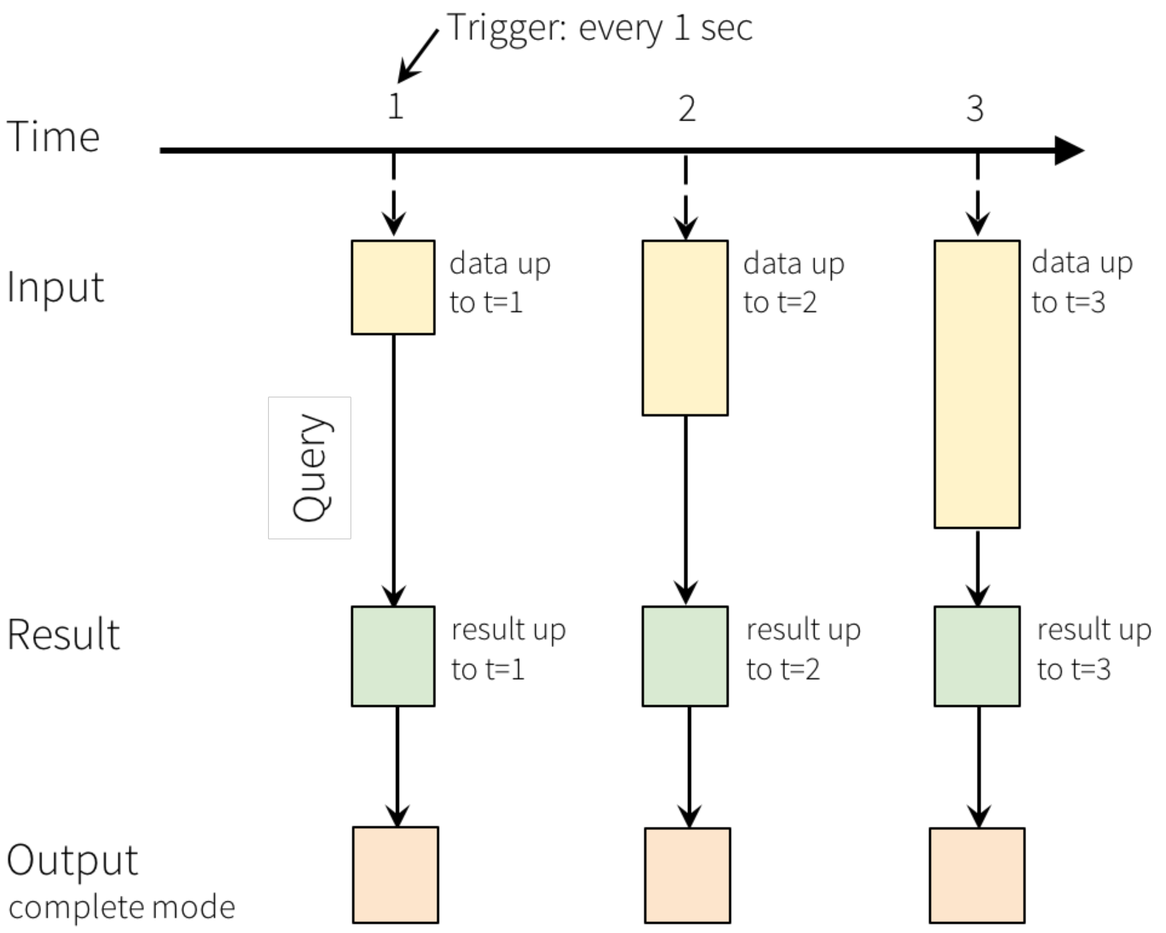
Structured Streaming

Basic Idea


Data stream
Unbounded Table
new data in the data stream
=
new rows appended to a unbounded table

Data stream as an unbounded table

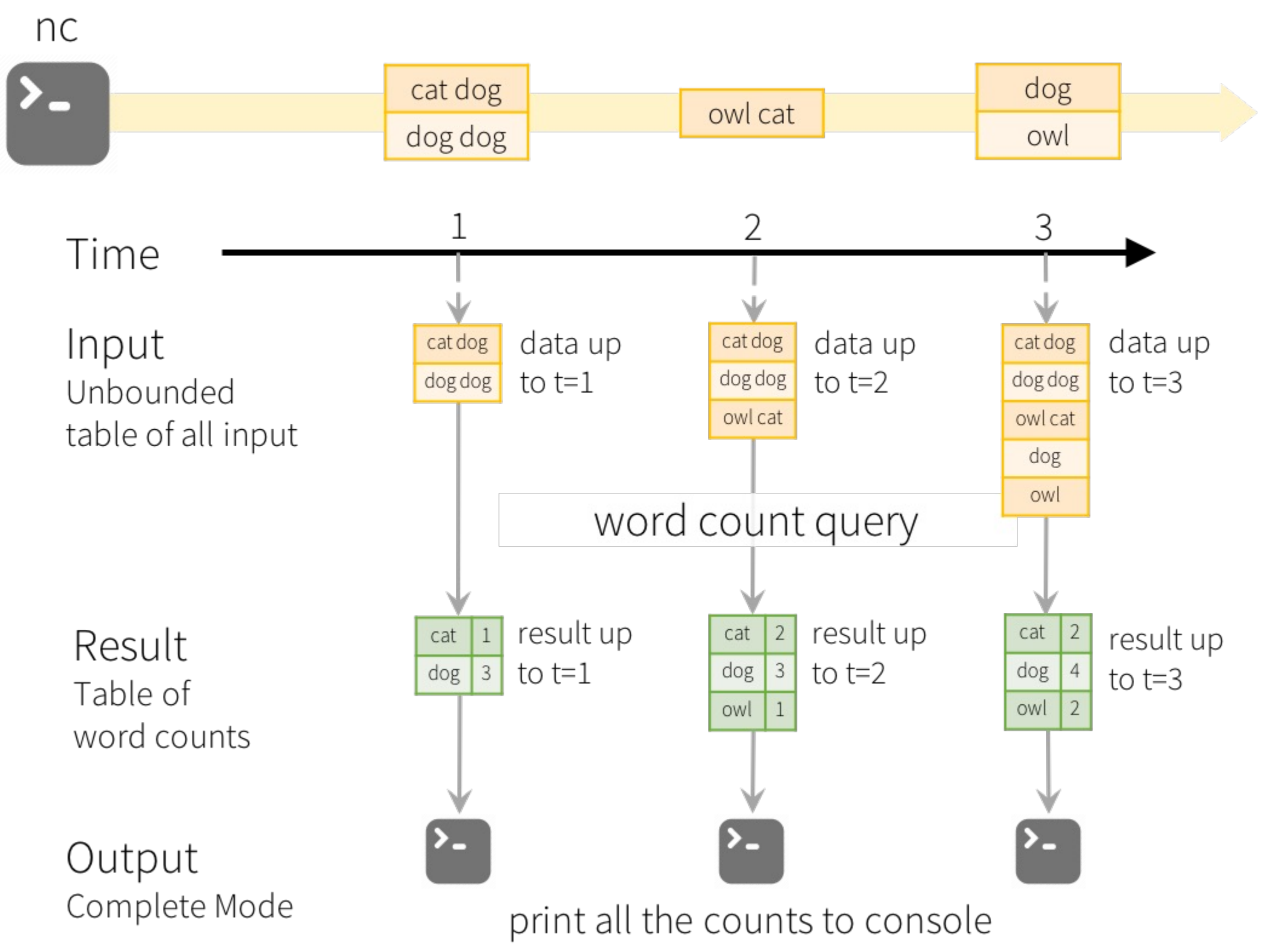
Behind the scene



Programming Model for Structured Streaming

UNIVERSITAT DE BARCELONA

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).

Practise in Databricks

- https://docs.microsoft.com/en-us/azure/databricks/_static/notebooks/structured-streaming-python.html

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