DSE 230 : Spark Streaming

- Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Data can be ingested from many sources like Kafka, Kinesis, or TCP sockets, and can be processed using complex algorithms expressed with high-level functions like `map`, `reduce`, `join` and `window`. Finally, processed data can be pushed out to filesystems, databases, and live dashboards.
- Spark Streaming receives live input data streams and divides the data into batches, which are then processed by the Spark engine to generate the final stream of results in batches.
- Spark Streaming provides a high-level abstraction called discretized stream or DStream, which represents a continuous stream of data. DStreams can be created either from input data streams from sources such as Kafka, and Kinesis, or by applying high-level operations on other DStreams. Internally, a DStream is represented as a sequence of RDDs.

Quick example

Count the number of words received from a data server listening on a TCP socket

Run network_wordcount.py using the following instructions:

1. Open a new terminal window in Jupyter Lab and run:
   - apt install netcat
   - nc -lk 9999
2. Open a second terminal window, navigate to the work directory and run the following:
   - spark-submit network_wordcount.py localhost 9999
3. Enter an input statement in the first window and watch it processed in the second

Import StreamingContext which is the main entry point for streaming functionality. We create a local StreamingContext with two execution threads and batch interval of 1 second

In [ ]:
```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

# Create a local StreamingContext with two working thread and batch interval of 1 secon
sc = SparkContext("local[2]", "NetworkWordCount")
ssc = StreamingContext(sc, 1)
```

Create a `DStream` that represents the data from the TCP source

In [ ]:
```
lines = ssc.socketTextStream("localhost", 9999)
```

Each record is a line of text - split the lines by space into words.

flatMap is a one-to-many DStream operation that creates a new DStream by generating multiple new records from each record in the source DStream. In this case, each line will be split into multiple words and the stream of words is represented as the words DStream

In [ ]:
```
words = lines.flatMap(lambda line: line.split(" "))
```

Count the words - The words DStream is further mapped (one-to-one transformation) to a DStream of (word, 1) pairs, which is then reduced to get the frequency of words in each batch of data. Finally, wordCounts.pprint() will print a few of the counts generated every second.

In [ ]:
```
pairs = words.map(lambda word: (word, 1))
wordCounts = pairs.reduceByKey(lambda x, y: x + y)

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

Note that when these lines are executed, Spark Streaming only sets up the computation it will perform when it is started, and no real processing has started yet. To start the processing after all the transformations have been setup, we finally call

In [ ]:
```
ssc.start()             # Start the computation
ssc.awaitTermination()  # Wait for the computation to terminate
```