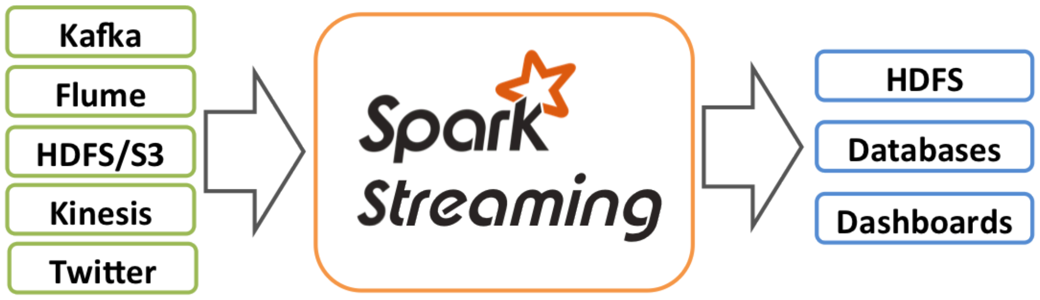
Lab: Spark Streaming

Status: Under review

This tutorial is intended to help you get started with the official Spark Streaming Programming Guide.

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. In fact, you can apply Spark's machine learning and graph processing algorithms on data streams.



Setup

Requirements

The tutorial assumes the following technical requirements:

- AWS Cloud9 environment
- Python 3.10 (included in Cloud9)
- Netcat (included in Cloud9)
- Pyspark

Configuration

Install pyspark as follows:

```
pip install pyspark
```

Spark Streaming Examples:

The following examples will read from a TCP stream and count the number of words during a specific window.

- In a new terminal, start the *netcat* application on port 9999:

```
      nc -lk 9999
```

Ex1: Network wordcount

Save the following code in a file called **network_wordcount.py**.

```
import sys

from pyspark import SparkContext
from pyspark.streaming import StreamingContext

if __name__ == "__main__":
    if len(sys.argv) != 3:
        # print("Usage: network_wordcount.py <hostname> <port>", file=sys.stderr)
        sys.exit(-1)
    sc = SparkContext(appName="PythonStreamingNetworkWordCount")
    ssc = StreamingContext(sc, 1)

    sc.setLogLevel("WARN")

    lines = ssc.socketTextStream(sys.argv[1], int(sys.argv[2]))

    counts = lines.flatMap(lambda line: line.split(" "))\
                  .map(lambda word: (word, 1))\
                  .reduceByKey(lambda a, b: a + b)

    counts.pprint()

    ssc.start()
    ssc.awaitTermination()
```

- To run the example, open a new terminal and execute the following command:

```
spark-submit network_wordcount.py localhost 9999
```

Ex2: Structured wordcount

Save the following code in a file called **structured_network_wordcount.py**.

```
import sys

from pyspark.sql import SparkSession
from pyspark.sql.functions import explode
from pyspark.sql.functions import split

if __name__ == "__main__":
    if len(sys.argv) != 3:
        # print("Usage: structured_network_wordcount.py <hostname> <port>", file=sys.stderr)
        sys.exit(-1)

    host = sys.argv[1]
    port = int(sys.argv[2])

    spark = SparkSession\
        .builder\
        .appName("StructuredNetworkWordCount")\
        .getOrCreate()

    spark.sparkContext.setLogLevel("WARN")

    # Create DataFrame representing the stream of input lines from connection to host:port
    lines = spark\
        .readStream\
        .format('socket')\
        .option('host', host)\
        .option('port', port)\
        .load()

    # Split the lines into words
    words = lines.select(
        # explode turns each item in an array into a separate row
        explode(
            split(lines.value, ' ')
        ).alias('word')
    )

    # Generate running word count
    wordCounts = words.groupBy('word').count()

    # Start running the query that prints the running counts to the console
    query = wordCounts\
        .writeStream\
        .outputMode('complete')\
        .format('console')\
        .start()

    query.awaitTermination()
```

- To run the example, open a new terminal and execute the following command:

```
spark-submit structured_network_wordcount.py localhost 9999
```