Spark Structured Streaming

Lab 03: Reading Network Streams

In this lab, we'll demonstrate streaming using a network connection using the Structured Streaming API. Our streaming job will wait for someone to connect to a network socket and will process the data that arrives on that socket.

Unlike in our prior labs, in this case, we'll be using Streaming Dataframes.

We'll walk through the process in this lab. For this and all the streaming labs, you will require two terminal windows on your cluster. We'll qualify which instructions to type in which terminal window in each step.

Terminal 1: Starting spark-shell

First, we have to ensure we run spark with enough cores to populate one core for the Spark streaming receiver and another for a Spark DStream.

Connect to your cluster using the secure shell, and type the following to start the spark-shell with two cores:

```
spark-shell --master local[2]
```

Terminal 1: Set Up For Network Streaming

Here, we're going to use the more modern Spark 2.0 instantiation of a network stream, using `spark.readStream.format` with a set of options.

```
// Create DataFrame representing the stream of input lines from connection to localhost:9999
val lines = spark.readStream.
  format("socket").
  option("host", "localhost").
  option("port", 9999).load()
```

Terminal 1: Wordcount code

Because we get to read the data as a DataFrame, we can use DataFrame operations such as groupBy.

```
// Split the lines into words
val words = lines.as[String].flatMap(_.split(" "))

// Generate running word count
val wordCounts = words.groupBy("value").count()
```

Terminal 2: Set Up Network Port 9999

We're going to use the network cat program `nc` to provide input to our stream. We do that by the following command

```
nc -lk 9999
```

Terminal 1: Start Streaming

Now we will set up our query of the network along with some parameters about how to process the output. We are going to use "complete" mode, which allows us to count all the words we have seen since the stream started.

```
val query = wordCounts.writeStream.
  outputMode("complete").
  format("console")
query.start().awaitTermination()
```

Terminal 2: Type Words

The terminal will now wait for you to type words. You should type some input for the streaming program. I suggest something such as:

```
Now is the time for the winter of our discontent.
Tomorrow and tomorrow and tomorrow
Creeps forth in its petty pace from day to day
To the last syllable of recorded time.
```

Terminal 1: See The WordCount Output

You should see the words get counted on Terminal 1 soon after you add each file. It should look like the following:

```
-------------------------------------------
Batch: 0
-------------------------------------------
+-----------+-----+
|      value|count|
+-----------+-----+
|     winter|    1|
|        for|    1|
|         is|    1|
|        Now|    1|
|        our|    1|
|discontent.|    1|
|        the|    2|
|         of|    1|
|       time|    1|
+-----------+-----+

-------------------------------------------
Batch: 1
-------------------------------------------
+-----------+-----+
|      value|count|
+-----------+-----+
|   Tomorrow|    1|
|   tomorrow|    2|
|     winter|    1|
|        for|    1|
|         is|    1|
|        Now|    1|
|        our|    1|
|discontent.|    1|
|        the|    2|
|        and|    2|
|         of|    1|
|       time|    1|
+-----------+-----+

-------------------------------------------
Batch: 2
-------------------------------------------
+-----------+-----+
|      value|count|
+-----------+-----+
|   Tomorrow|    1|
|      time.|    1|
|   tomorrow|    2|
|     winter|    1|
|      forth|    1|
|        for|    1|
|         in|    1|
|      petty|    1|
|         is|    1|
|   syllable|    1|
|        its|    1|
|        Now|    1|
|        our|    1|
|discontent.|    1|
|        the|    3|
|     creeps|    1|
|        and|    2|
|         of|    2|
|       time|    1|
|   recorded|    1|
+-----------+-----+
only showing top 20 rows
```

You should now kill the job by hitting control-C in Terminal 1 and Terminal 2. You will likely see many errors on Terminal 1 as the streaming network job shuts down.

This step concludes the lab.