Spark Streaming

Lab 01: Reading File Streams

In this lab, we'll demonstrate streaming using files in an HDFS directory. Our streaming job will wait for a file to arrive in the directory and then immediately count the words in it.

Note that a new RDD is created and immediately processed each time a file is discovered.

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: Making The Input Directory And 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:

```
hdfs dfs -mkdir /tmp/streaming-input
spark-shell --master local[2]
```

Terminal 1: Imports and Creating your Spark Context

We have to import the Spark Streaming module and some definitions. We also have to decide how often we want our micro-batches to run. For this example, we'll poll the input directory every 10 seconds

```
import org.apache.spark.SparkConf
import org.apache.spark.streaming.{Seconds, StreamingContext}

// Create the streaming context
val ssc = new StreamingContext(sc, Seconds(10))
```

Note that the streaming context is created from our Spark Context, which in the spark-shell is in the variable `sc`.

Terminal 1: Wordcount code

Now we simply use the Scala wordcount code we are familiar with. The only difference with previous wordcount examples is that we're reading our text file with `sc.textFileStream` instead of `sc.textFile`.

```
    val lines = ssc.textFileStream("/tmp/streaming-input")
    val words = lines.flatMap(_.split(" "))
    val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
    wordCounts.print()
```

Terminal 1: Start The Stream

Now we simply start our stream using `ssc.start`.

```
ssc.start()
ssc.awaitTermination()  // Wait for the computation to terminate
```

Terminal 2: Copy A File Into /tmp/streaming-input

We'll now copy all of Shakespeare into the input directory, followed by the text of some logs about 30 seconds later. ~~~ bash hdfs dfs -cp /data/shakespeare/input/all-shakespeare.txt /tmp/streaming-input hdfs dfs -cp /data/logs/log.txt /tmp/streaming-input ~~~

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:

```
-------------------------------------------
Time: 1504475990000 ms
-------------------------------------------
(hack'd.,1)
(House,1)
(nobleman,10)
(Never,,1)
(dream'd,15)
(stuck,,1)
(perpetual.,2)
(bombast,2)
(unluckily,,2)
(consideration,,3)
...

-------------------------------------------
Time: 1504476000000 ms
-------------------------------------------

-------------------------------------------
Time: 1504476010000 ms
-------------------------------------------
(cluster:,1)
(ERROR  mysql,1)
(angry,1)
(mysql,1)
(at,1)
(ERROR  php:,1)
(are,1)
(replace,1)
(with,1)
(me?,1)
...
```

You should now kill the job by hitting control-C in Terminal 1.

This step concludes the lab.