**Assignment22.2**

**Stateful Streaming in Apache Spark**

Apache Spark is a general processing engine built on top of the Hadoop eco-system. Spark has a complete setup and a unified framework to process any kind of data. Spark can do batch processing as well as stream processing. Spark has a powerful SQL engine to run SQL queries on the data; it also has an integrated Machine Learning library called MlLib and a graph processing library called GraphX. As it can integrate many things into it, we identify Spark as a unified framework rather than a processing engine.

Now coming to the real-time stream processing engine of Spark. Spark doesn’t process the data in real time it does a near-real-time processing. It means it processes the data in micro batches, in just a few milliseconds.

Here we have got a program where Spark’s streaming context will process the data in micro batches but generally, this processing is stateless. Let’s take we have defined the streaming Context to run for every 10 seconds, it will process the data that is arrived within that 10 seconds, to process the previous data we have something called windows concept, windows cannot give the accumulated results from the starting timestamp.

But what if you need to the accumulate the results from the start of the streaming job. Which means you need to check the previous state of the RDD in order to update the new state of the RDD. This is what is known as stateful streaming in Spark.

Spark provides 2 API’s to perform stateful streaming, which is updateStateByKey and mapWithState.

Now we will see how to perform stateful streaming of wordcount using updateStateByKey. UpdateStateByKey is a function of Dstreams in Spark which accepts an update function as its parameter. In that update function, you need to provide the following parameters newState for the key which is a seq of values and the previous state of key as an Option[?].

Let’s take a word count program, let’s say for the first 10 seconds we have given this data hello every one from acadgild. Now the wordcount program result will be

(one,1)

(hello,1)

(from,1)

(acadgild,1)

(every,1)

Now without writing the updateStateByKey function, if you give some other data, in the next 10 seconds i.e. let’s assume we give the same line hello every one from acadigld. Now we will get the same result in the next 10 seconds also i.e.,

(one,1)

(hello,1)

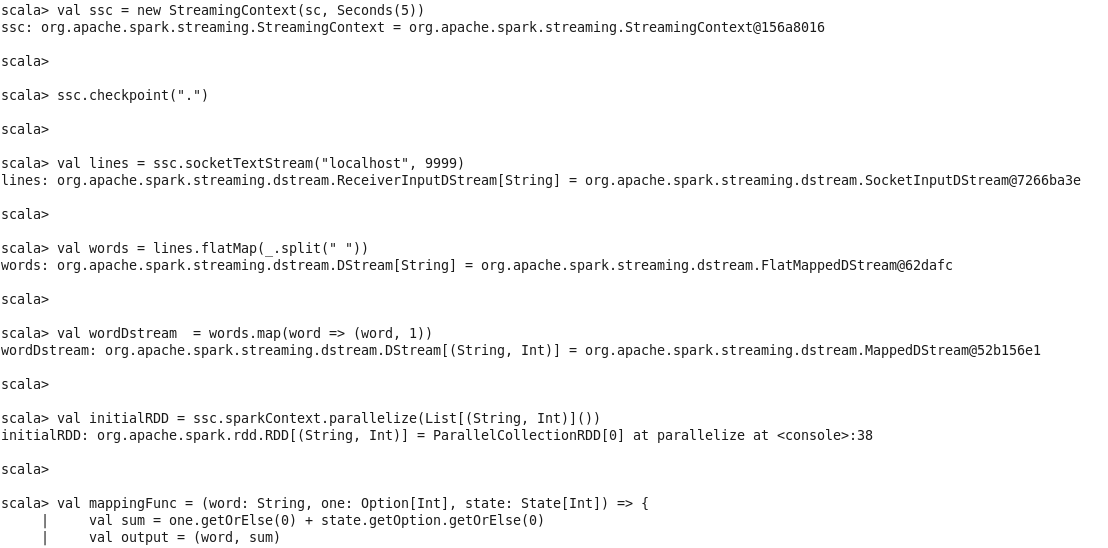
(from,1)

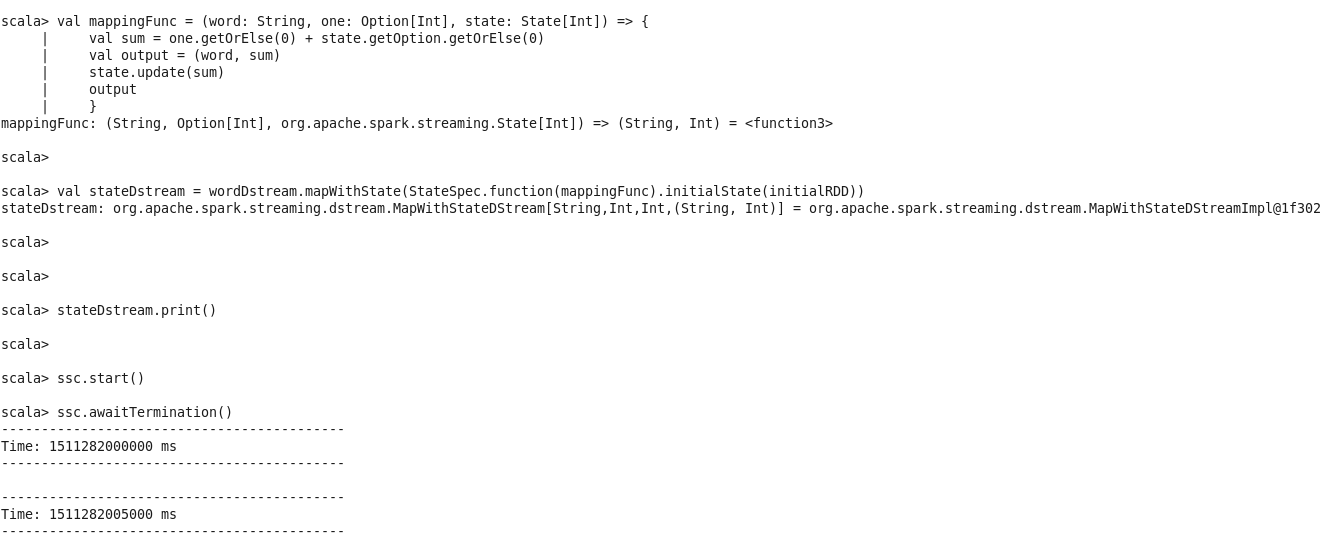
(acadgild,1)

(every,1)

Now, what if we need an accumulated result of the wordcount which counts my previous results also. This is where stateful streaming comes into the act. In stateful streaming, your key’s previous state will be preserved and it will be updated with new results.









The major difference here is the addition of the update function and the addition of updateStateByKey function to Dstream.

val mappingFunc = (word: String, one: Option[Int], state: State[Int]) => {

val sum = one.getOrElse(0) + state.getOption.getOrElse(0)

val output = (word, sum)

state.update(sum)

output

}

val stateDstream = wordDstream.mapWithState(StateSpec.function(mappingFunc).initialState(initialRDD))

The updateFunc will work on each key, for every key in the RDD, this update function will run, it will take the last state of your key and it will check for the new values for your and the data operation whatever you want to do for your key and return the new values as a Some().

For working with this update function, you need to mandatorily provide a Checkpoint directory for your SparkStreamingContext as

ssc.checkpoint(“hdfs://localhost:9000/WordCount\_checkpoint”)

Because your intermediate values will be stored in this checkpoint directory for fault tolerance, it is suggested that you give your checkpoint directory in HDFS for more fault tolerance.

In the above update function, we are getting the new values of that key as a Seq[Int] and the oldValues of that key as Option[Int](Which is already calculated). Now inside the function, we aggregating the newValues of the key using the foldLeft function and then we are getting the old state value of the key and we are adding the both to the Some() and returning the updated sum of the values.

As we can see from the above screenshot we get the updated states of the words ‘this’ and ‘is’.

This is how we perform stateful streaming using updateStateByKey function.