Week 15 Scala – PySpark equivalent programs

Week 15 is based on Spark streaming where we need real time stream. We will use socket and file

Generalized changes that are required in every program

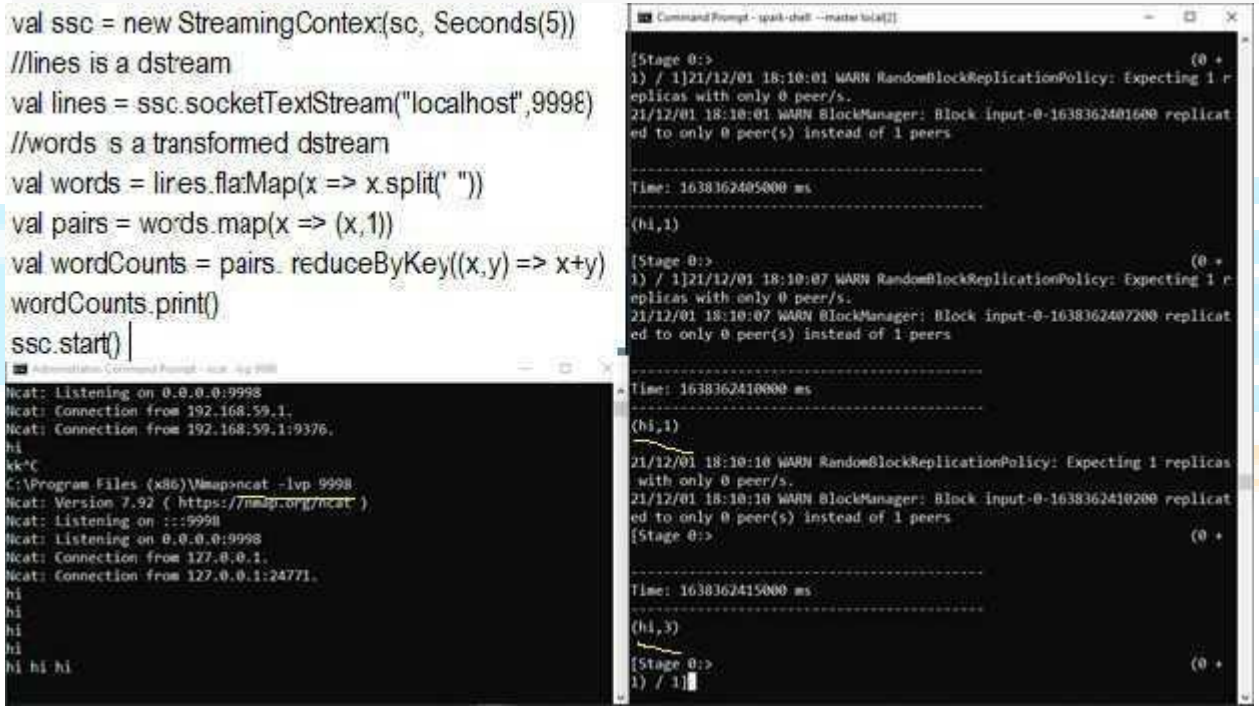
1. To start cmd prompt for PySpark. We write PySpark instead of scala-shell.
2. Remove all val, var keyword as python does not have val and var types.
3. Anonymous functions are replaced with lambda in python.
4. Comment is given using # in python instead of // in scala

Note

1. Best practice is to use your own itversity hdfs location in the program for input and output files. You can also use Linux root as shown in video.
2. There are be many ways to get the output for particular problem, we are showcasing one way.
3. Changes are highlighted in yellow.
4. Ncat is Linux utility. For windows, follow below steps.

Steps for streaming program execution on windows.

1. Download the Free Nmap Security Scanner for Linux/Mac/Windows Download nmap-stable setup and install
2. In code give localhost 9998
3. Run the code... It will give error because no port is listening ... That's ok
4. Open cmd – go to Nmap folder
5. ncat -lvp 9998
6. start typing words
7. Cross check in your program



Problem Statement: Write a real time word count program

Solution:

Scala Spark Program	PySpark Program
Spark-shell –master local[2]	PySpark –master local[2]

import org.apache.spark._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
 	
//creating spark streaming context
val ssc = new StreamingContext(sc, Seconds(2))
//lines is a dstream
val lines = ssc.socketTextStream("localhost",9998)
//words is a transformed dstream
val words = lines.flatMap(x => x.split(" "))
val pairs = words.map(x => (x,1))
val wordCounts = pairs. reduceByKey((x,y) => x+y)
wordCounts.print()
ssc.start()
	from PySpark import *
	from PySpark.streaming import *
	
	sc.setLogLevel("ERROR")
	#creating spark streaming context
	ssc = StreamingContext(sc, 2)
	#lines is a dstream
	lines = ssc.socketTextStream("localhost", 9998)
	#words is a transformed dstream
	words = lines.flatMap(lambda x: x.split())
	pairs = words.map(lambda x: (x, 1))
	wordCounts = pairs.reduceByKey(lambda x, y: x + y)
	wordCounts.pprint()
	ssc.start()

Specific changes that are required in above program

1. In scala we give Seconds(2) whereas in python you can give directly 2
2. We use pprint in python. The pprint module provides a capability to “pretty-print” arbitrary Python data structures in a well-formatted and more readable way.

Problem Statement: Write real time stateless word count program in IDE

Solution:

Scala Spark Program	PySpark Program
Create word.scala	Create word.py

import org.apache.spark.SparkContext
import org.apache.spark.streaming.Seconds
import org.apache.spark.streaming.StreamingContext
 	
object StreamingWordCount extends App{
val sc = new SparkContext("local[*]","wordcount")
//creating spark streaming context
val ssc = new StreamingContext(sc, Seconds(5))
//lines is a dstream
val lines = ssc.socketTextStream("localhost",9998)
//words is a transformed dstream
val words = lines.flatMap(x => x.split(" "))
val pairs = words.map(x => (x,1))
val wordCounts = pairs. reduceByKey((x,y) => x+y)
wordCounts.print()
ssc.start()
ssc.awaitTermination()
}
	from PySpark import *
	from PySpark.streaming import *
	sc =SparkContext("local[2]","APP")
	sc.setLogLevel("ERROR")
	#creating spark streaming context
	ssc = StreamingContext(sc, 2)
	#lines is a dstream
	lines = ssc.socketTextStream("localhost", 9998)
	#words is a transformed dstream
	words = lines.flatMap(lambda x: x.split())
	pairs = words.map(lambda x: (x, 1))
	wordCounts = pairs.reduceByKey(lambda x, y: x + y)
	wordCounts.pprint()
	ssc.start()
	ssc.awaitTermination()

Specific changes that are required in above program

1. In IDE you need to create SparkContext object
2. Last line we need to write code to wait for termination which is same.
3. Green color is the change from PySpark shell to IDE
4. Yellow highlight is change from scala to PySpark in shell

Problem Statement: Write a real time stateful word count program in IDE

Solution:

Scala Spark Program	PySpark Program
Create word1.scala	Create word1.py

import org.apache.spark.SparkContext
import org.apache.spark.streaming.Seconds
import org.apache.spark.streaming.StreamingContext
 	
object StreamingWordCount extends App{
val sc = new SparkContext("local[*]","wordcount")
//creating spark streaming context
val ssc = new StreamingContext(sc, Seconds(5))
//lines is a dstream
val lines = ssc.socketTextStream("localhost",9998)
 	
ssc.checkpoint(".")
def updatefunc(newValues:Seq[Int],previousState:Option[Int]): Option[Int]={
val newCount= previousState.getOrElse(0) + newValues.sum
Some(newCount)
}
val words = lines.flatMap(x => x.split(" "))
val pairs = words.map(x => (x,1))
val wordCounts = pairs.updateStateByKey(updatefunc)
wordCounts.print()
ssc.start()
ssc.awaitTermination()
}
	from PySpark import *
	from PySpark.streaming import *
	
	sc =SparkContext("local[2]","APP")
	sc.setLogLevel("ERROR")
	#creating spark streaming context
	ssc = StreamingContext(sc, 2)
	#lines is a dstream
	lines = ssc.socketTextStream("localhost", 9998)
	
	ssc.checkpoint(".")
	def updatefunc(newValues, previousState):
	  if previousState is None :
	  previousState = 0
	  return sum(newValues, previousState)
	
	words = lines.flatMap(lambda x: x.split())
	pairs = words.map(lambda x: (x, 1))
	wordCounts = pairs.updateStateByKey(updatefunc)
	wordCounts.pprint()
	ssc.start()
	ssc.awaitTermination()

Specific changes that are required in above program

1. updateFunc function definition is changed
2. Yellow highlight is change in previous and this scala program
3. Green highlight is change between scala to PySpark.

Problem Statement: Write a real time stateful word count program using sliding window in IDE

Solution:

Scala Spark Program	PySpark Program
Create word1.scala	Create word1.py

import org.apache.spark.SparkContext
import org.apache.spark.streaming.Seconds
import org.apache.spark.streaming.StreamingContext
 	
object StreamingWordCount extends App{
val sc = new SparkContext("local[*]","wordcount")
//creating spark streaming context
val ssc = new StreamingContext(sc, Seconds(5))
//lines is a dstream
val lines = ssc.socketTextStream("localhost",9998)
ssc.checkpoint(".")
//words is a transformed dstream
val wordCounts = lines.flatMap(x => x.split(" "))
.map(x => (x,1))
.reduceByKeyAndWindow((x,y)=>x+y,(x,y)=>x-y,Seconds(10),Seconds(2))
wordCounts.print()
ssc.start()
ssc.awaitTermination()
}
	from PySpark import *
	from PySpark.streaming import *
	sc =SparkContext("local[2]","APP")
	sc.setLogLevel("ERROR")
	#creating spark streaming context
	ssc = StreamingContext(sc, 2)
	sss.checkpoint(".")
	#lines is a dstream
	lines = ssc.socketTextStream("localhost", 9998)
	#words is a transformed dstream
	wordCounts = lines.flatMap(lambda x: x.split()) \
	.words.map(lambda x: (x, 1)) \
	.reduceByKeyAndWindow(lambda x, y: int(x) + int(y), lambda x, y: int(x) - int(y), 10, 2)
	
	wordCounts.pprint()
	ssc.start()
	ssc.awaitTermination()

// show filter
.filter(x => x._2>0)
	#add filter to print count > 2
	.filter(lambda x:x[1]>2)

Specific changes that are required in above program

1. We are converting x and y into int and seconds are given directly.

Problem Statement: Write a real time stateful word count program using sliding window and named function in IDE

Solution:

Scala Spark Program	PySpark Program

import org.apache.spark.SparkContext
import org.apache.spark.streaming.Seconds
import org.apache.spark.streaming.StreamingContext
 	
object StreamingWordCount extends App{
val sc = new SparkContext("local[*]","wordcount")
//creating spark streaming context
val ssc = new StreamingContext(sc, Seconds(5))
//lines is a dstream
val lines = ssc.socketTextStream("localhost",9998)
ssc.checkpoint(".")
def summaryFuct(x:Int, y:Int)={x+y}
 	
def inverseFuct(x:Int, y:Int)={x-y}
 	
//words is a transformed dstream
val wordCounts = lines.flatMap(x => x.split(" "))
.map(x => (x,1))
.reduceByKeyAndWindow(summaryFuct(_,_),inverseFuct(_,_),Seconds(10),Seconds(2))
.filter(x => x._2>0)
 	
wordCounts.print()
ssc.start()
ssc.awaitTermination()
}
	from PySpark import *
	from PySpark.streaming import *
	sc =SparkContext("local[2]","APP")
	sc.setLogLevel("ERROR")
	#creating spark streaming context
	ssc = StreamingContext(sc, 2)
	sss.checkpoint(".")
	#lines is a dstream
	lines = ssc.socketTextStream("localhost", 9998)
	
	def summaryFuct(x,y):
	  return x + y
	
	def inverseFuct(x, y):
	  return x - y
	
	#words is a transformed dstream
	wordCounts = lines.flatMap(lambda x: x.split()) \
	.words.map(lambda x: (x, 1)) \
	.reduceByKeyAndWindow(summaryFuct, inverseFuct, 10, 2) \
	.filter(lambda x:x[1]>0)
	
	wordCounts.pprint()
	ssc.start()
	ssc.awaitTermination()

Specific changes that are required in above program

1. Function definition is different in python. We don’t specify datatype for parameters. Also, while calling function, just give function name.

Problem Statement: Write a real time stateful word count program using sliding window and named function in IDE. Implement reduceByWindow method which does not need pair RDD

Solution:

Scala Spark Program	PySpark Program

import org.apache.spark.SparkContext
import org.apache.spark.streaming.Seconds
import org.apache.spark.streaming.StreamingContext
 	
object StreamingWordCount extends App{
val sc = new SparkContext("local[*]","wordcount")
//creating spark streaming context
val ssc = new StreamingContext(sc, Seconds(5))
//lines is a dstream
val lines = ssc.socketTextStream("localhost",9998)
ssc.checkpoint(".")
def summaryFuct(x:String, y: String)={ (x.toInt + y.toInt).toString()}
def inverseFuct(x: String, y: String)= { (x.toInt - y.toInt).toString()}
 	
//words is a transformed dstream
val wordCounts = lines.~~flatMap(x => x.split(" "))~~
~~.map(x => (x,1))~~
.reduceByWindow(summaryFuct(_,_),inverseFuct(_,_),Seconds(10),Seconds(2))
~~.filter(x => x._2>0)~~
 	
wordCounts.print()
ssc.start()
ssc.awaitTermination()
}
	from PySpark import *
	from PySpark.streaming import *
	sc =SparkContext("local[2]","APP")
	sc.setLogLevel("ERROR")
	#creating spark streaming context
	ssc = StreamingContext(sc, 2)
	sss.checkpoint(".")
	#lines is a dstream
	lines = ssc.socketTextStream("localhost", 9998)
	
	def summaryfuct(x, y):
	  return str((int(x)+int(y)))
	
	def inversefuct(x, y):
	  return str((int(x)-int(y)))
	
	wordCounts= lines.reduceByWindow(summaryfuct, inversefuct, 10, 2)
	
	wordCounts.pprint()
	ssc.start()
	ssc.awaitTermination()

Specific changes that are required in above program

1. reduceByWindow does not need pair RDD hence we don't need flapMap and map

Problem Statement: Write a real time program to count number of lines in window.

Solution:

Scala Spark Program	PySpark Program

import org.apache.spark.SparkContext
import org.apache.spark.streaming.Seconds
import org.apache.spark.streaming.StreamingContext
 	
object StreamingWordCount extends App{
val sc = new SparkContext("local[*]","wordcount")
//creating spark streaming context
val ssc = new StreamingContext(sc, Seconds(2))
//lines is a dstream
val lines = ssc.socketTextStream("localhost",9998)
ssc.checkpoint(".")
 	
//words is a transformed dstream
val wordCounts = lines.countByWindow(Seconds(10),Seconds(2))
 	
wordCounts.print()
ssc.start()
ssc.awaitTermination()
}
	from PySpark import *
	from PySpark.streaming import *
	sc =SparkContext("local[2]","APP")
	sc.setLogLevel("ERROR")
	#creating spark streaming context
	ssc = StreamingContext(sc, 2)
	sss.checkpoint(".")
	#lines is a dstream
	lines = ssc.socketTextStream("localhost", 9998)
	
	wordCounts= lines.countByWindow(10, 2)
	
	wordCounts.pprint()
	ssc.start()
	ssc.awaitTermination()

Specific changes that are required in above program

1. To specify seconds for interval in python, specify directly.