

Session 26:
SPARK STREAMING
Assignment 1
- Prachi Mohite

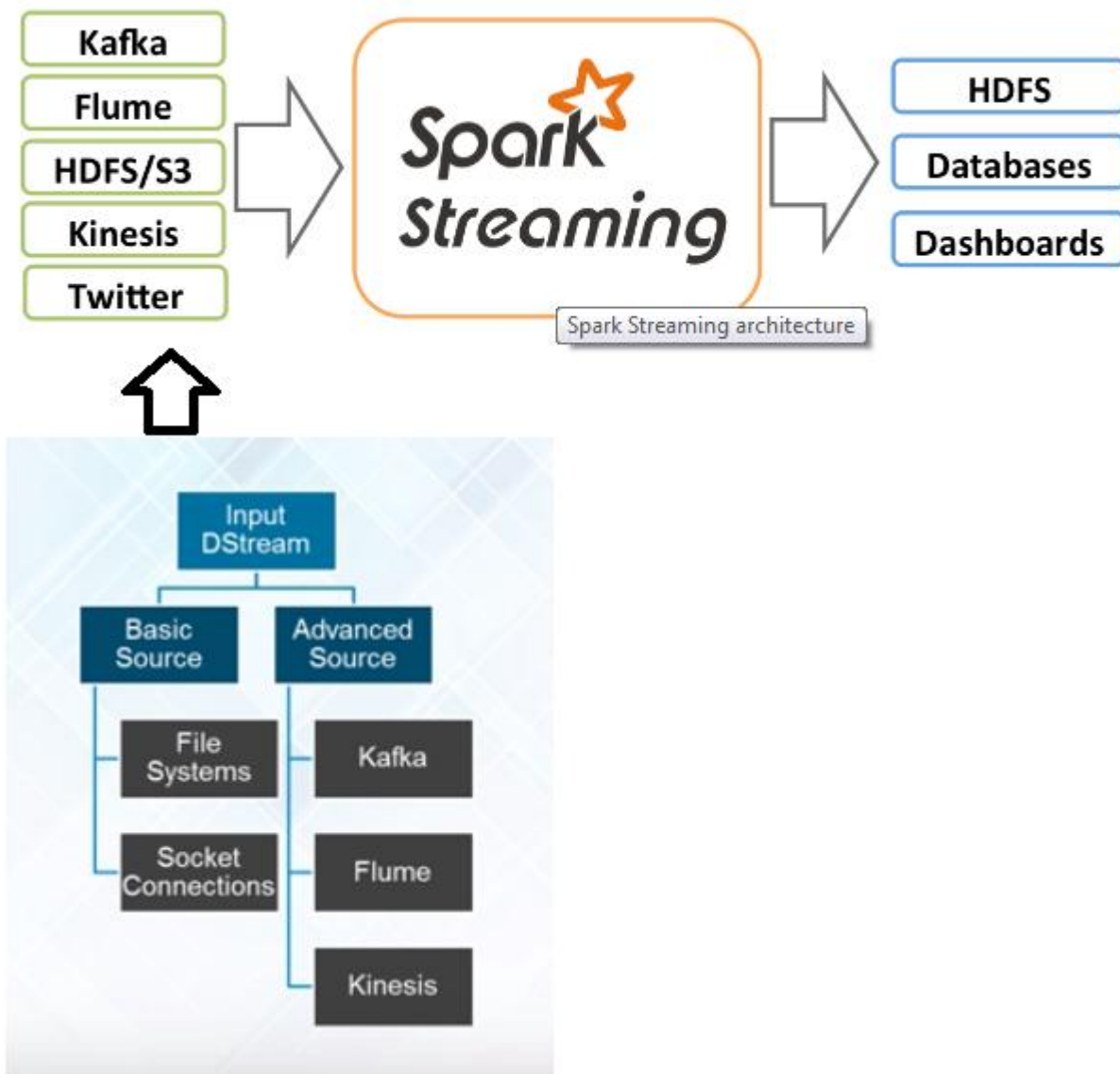
Some Briefs about Spark Streaming

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Spark Streaming can be used to stream live data and processing can happen in real time. Spark Streaming's ever-growing user base consists of household names like Uber, Netflix and Pinterest.

Spark Streaming Features

1. **Scaling:** Spark Streaming can easily scale to hundreds of nodes.
2. **Speed:** It achieves low latency.
3. **Fault Tolerance:** Spark has the ability to efficiently recover from failures.
4. **Integration:** Spark integrates with batch and real-time processing.
5. **Business Analysis:** Spark Streaming is used to track the behavior of customers which can be used in business analysis.

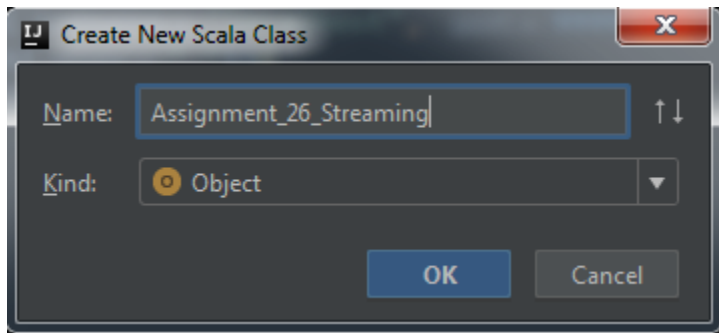
Spark Streaming workflow has four high-level stages. The first is to stream data from various sources. These sources can be streaming data sources like Akka, Kafka, Flume, AWS or Parquet for real-time streaming. The second type of sources includes HBase, MySQL, PostgreSQL, Elastic Search, Mongo DB and Cassandra for static/batch streaming. Once this happens, Spark can be used to perform Machine Learning on the data through its MLlib API. Further, Spark SQL is used to perform further operations on this data. Finally, the streaming output can be stored into various data storage systems like HBase, Cassandra, MemSQL, Kafka, Elastic Search, HDFS and local file system.



Note: For below assignment we are considering the basic source of data streaming i.e. socket connections. To achieve the same we will be executing the net cat on Windows machine and listen it through Spark Streaming

For This assignment we will be using IDEA IntelliJ.

1. Created Project in IDEA IntelliJ and added object for this assignment as below



2. Created the spark context and streaming context as below

Spark Context

```
//Created Spark Context Object
val conf = new
SparkConf().setMaster("local[*]").setAppName("Assignment_26")
val sc = new SparkContext(conf)

sc.setLogLevel("WARN")
println("Spark Context Created")
```

Spark Streaming Context

```
//Create the streaming Object
val ssc = new StreamingContext(sc, Seconds(15))
```

To write above code we have to import all the spark streaming packages

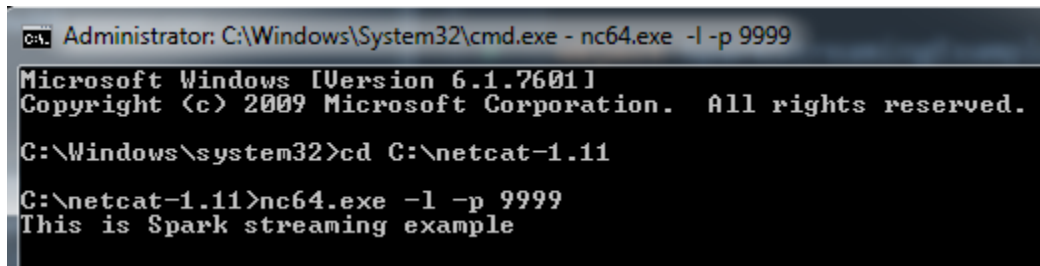
```
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.{SparkConf, SparkContext}
```

Here the first argument while creating the streaming context is object of spark context and second argument specifies the time interval when the input stream will be read.

To have live streaming from socket connections we will be using net cat utility, computer networking utility for reading from and writing to network connections using TCP or UDP.

1. We have downloaded the net cat utility for Windows.

2. Opened the command prompt as administrator and navigated to path of net cat utility placed in file system
3. Ran command `nc64.exe -l -p 9999` (64 indicates 64 bit machine, -l for local)



```
Administrator: C:\Windows\System32\cmd.exe - nc64.exe -l -p 9999
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\netcat-1.11
C:\netcat-1.11>nc64.exe -l -p 9999
This is Spark streaming example
```

We can configure this on VM as well.

Task 1

Read stream of Strings, fetch the words which can be converted to numbers. Filter out the rows, where the sum of numbers in that line is odd. Provide the sum of all the remaining numbers in that batch.

Solution Approach –

1. All required objects are created as above.
2. We have to create after listening from port 9999 which has tuple for word and number associated with that
3. As per the requirement to have to check if line has any numbers then add those and print the line if sum is ODD. This can be done while data is streaming
4. However we have to get the sum of all the remaining lines which has even sum. To achieve this we have to declare an accumulator which will hold the sum of numbers belonging to lines having even sum.

First created a function which should return the sum of Integer (Number) for words which can be converted to numbers in the given input line

```
//Defined a function which get the line from streaming and converts
the words to numbers
def getLineSum(line:String):Int={
  var number:Int=0;
  val lineWords = line.split(" ");
  for(x <- lineWords) {
    try {
      //If words can be converted to number it toInt won't throw a
      exception
      val f = x.toInt
```

```

        number = number + f
    }
    catch {
        case ex: Exception => {}
    }
}
return

```

Read the stream of strings and apply a for loop through `foreachRDD`, where `DStream` will be further divided into `RDD` and operation can be performed on every `RDD`

Below steps are included in for loop on `DStream`

1. Make string out of every `RDD` , separated with space
2. Call the above function which will get the sum of numbers in that line
3. Check if sum is Odd if yes print the line with its sum
4. Else if sum is not we have to add the sum to variable which should give us the sum of entire batch where line's sum is Even
 - a. To achieve this we have declare the accumulator

Some brief about accumulators in Spark

- Accumulator is a shared variable in Apache Spark, used to aggregating information across the cluster.
- In other words, aggregating information / values from worker nodes back to the driver program
- Accumulators are variables that are used for aggregating information across the executors. For example, this information can pertain to data or API diagnosis like how many records are corrupted or how many times a particular library API was called.

```

//Below line will read the text entered after running netcat utility
val lines = ssc.socketTextStream("localhost",
9999,StorageLevel.MEMORY_AND_DISK_SER)
val sumOfEven = sc.accumulator(0,"sumOfEven")
lines.foreachRDD(x=>{
    val ln = x.collect().mkString(" ").toString
    if(getLineSum(ln)%2!=0)
    {
        println("Entered line is having odd sum. Line was==> " + ln + "
& Sum is ==> " + getLineSum(ln))
    }
    else
    {
        sumOfEven.add(getLineSum(ln))
        println("Sum of numbers belonging to Line having even Sum (As of

```

```
now)==> "+ sumOfEven)
    }
  })
```

Complete Code

```
package Assignment_26_Streaming

import org.apache.spark.storage.StorageLevel
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.{SparkConf, SparkContext}

object Assignment_26_Streaming {

  def main(args: Array[String]): Unit = {
    println("hey Scala")

    //Created Spark Context Object
    val conf = new
SparkConf().setMaster("local[*]").setAppName("Assignment_26")
    val sc = new SparkContext(conf)

    sc.setLogLevel("WARN")
    println("Spark Context Created")

    //Defined a function which get the line from streaming and
    converts the words to numbers
    def getLineSum(line:String):Int={
      var number:Int=0;
      val lineWords = line.split(" ");
      for(x <- lineWords) {
        try {
          //If words can be converted to number it toInt won't throw a
          exception
          val f = x.toInt
          number = number + f
        }
        catch {
          case ex: Exception => {}
        }
      }
      return number;
    }

    //Create the streaming Object
```

```

val ssc = new StreamingContext(sc, Seconds(5))

//Below line will read the text entered after running netcat
utility
val lines = ssc.socketTextStream("localhost",
9999,StorageLevel.MEMORY_AND_DISK_SER)
val sumOfEven = sc.accumulator(0,"sumOfEven")
lines.foreachRDD(x=>{
    val ln = x.collect().mkString(" ").toString
    if(getLineSum(ln)%2!=0)
    {
        println("Entered line is having odd sum. Line was==> " + ln
+ " & Sum is ==> " + getLineSum(ln))
    }
    else
    {
        sumOfEven.add(getLineSum(ln))
        println("Sum of numbers belonging to Line having even Sum
(As of now)==> "+ sumOfEven)
    }
})
ssc.start()
ssc.awaitTermination()

}
}

```

Output

Streaming Window

operable program or batch file.

```

C:\netcat-1.11>nc64.exe -l -p
nc64.exe: option requires an argument -- p
nc -h for help

```

```

C:\netcat-1.11>nc64.exe -l -p 9999
This is Example 1 of assignment 26
I liked session 4 as well
I liked Spark SQL session 1 and 2 as well
I am waiting for case study 4

```

————— This line is having numbers 1 , 26 sum = 27, which is odd , so display this line in output
————— This line has number 4 which is even so total sum of remaining no.s = 4 as of now
————— This line has numbers as 1 and 2 so sum is 3 , which is odd, display this line.
└—————┘ This line has number 4, which is even so total sum of remaining no.s is 4+4 = 8 (as of now)


```

Sum of numbers belonging to Line having even Sum (As of now)==> 0
Sum of numbers belonging to Line having even Sum (As of now)==> 0
Sum of numbers belonging to Line having even Sum (As of now)==> 0
Sum of numbers belonging to Line having even Sum (As of now)==> 0
Entered line is having odd sum. Line was==> This is Example 1 of assignment 26 & Sum is ==> 27
Sum of numbers belonging to Line having even Sum (As of now)==> 4
Sum of numbers belonging to Line having even Sum (As of now)==> 4 These two line has sum of numbers which is
Sum of numbers belonging to Line having even Sum (As of now)==> 4 odd. So displaying two lines
Entered line is having odd sum. Line was==> I liked Spark SQL session 1 and 2 as well & Sum is ==> 3
Sum of numbers belonging to Line having even Sum (As of now)==> 4
Sum of numbers belonging to Line having even Sum (As of now)==> 4
Sum of numbers belonging to Line having even Sum (As of now)==> 8
Sum of numbers belonging to Line having even Sum (As of now)==> 8 Sum of remaining numbers from the
Sum of numbers belonging to Line having even Sum (As of now)==> 8 other lines is 4+4 = 8
Sum of numbers belonging to Line having even Sum (As of now)==> 8
Sum of numbers belonging to Line having even Sum (As of now)==> 8
Sum of numbers belonging to Line having even Sum (As of now)==> 8
Sum of numbers belonging to Line having even Sum (As of now)==> 8
Sum of numbers belonging to Line having even Sum (As of now)==> 8

```

Task 2

Read two streams

1. List of strings input by user

2. Real-time set of offensive words

Find the word count of the offensive words inputted by the user as per the real-time set of offensive words

Solution Approach –

As we have to read the two stream we can either read from Textfile stream or having another socket streaming with different portal.

Step 1:

Added code to read the real time offensive words.

As these words should stay in memory we need to add a variable which will load these words in memory as below

```

//ArrayBuffer to store list of offensive words in memory
val wordList: ArrayBuffer[String] = ArrayBuffer.empty[String];

```

We mapped offensive words as word split by ‘,’

```

//Below line will read the text entered after running netcat utility
val offensivelines = ssc.socketTextStream("localhost",
9000,StorageLevel.MEMORY_AND_DISK_SER)
//Need to Read these lines and keep those in broad case variable to
send it to all
val offensiveWords= offensivelines.flatMap(_.split(",")).map(x=>(x))
// storing offensive words in ArrayBuffer
offensiveWords.foreachRDD(a => { a.foreach(f => {wordList += f})});

```

Step 2:

Read the second stream with lines entered by user which may have offensive words
Those lines are split and mapped as word and counter which then reduced to get the word count on basis of word as key.

```
//Reading lines entered by user
val lines= ssc.socketTextStream("localhost",
9999,StorageLevel.MEMORY_AND_DISK_SER)
//If word is duplicate then summing them on key (i.e. word)
val wordCount = lines.flatMap(line => line.split(" ")).map(word =>
(word, 1)).reduceByKey(_+_)
```

Once this is done wrote code to match the words with offensive words and print the count

```
/Getting word count of offensive words only
val offensiveWordsRDD = wordCount.filter {x => matchWord(x._1)%2==1 };
offensiveWordsRDD.print();
```

```
/** * Filter Method for offensive words */
def matchWord(ln : String): Double=
{
    val lineWords = ln.trim.toLowerCase();
    var num: Double = 0;
    for(y<-wordList) {
        if(y.toLowerCase() == lineWords)
        { num = 1; return num; }
    }
    return num;
}
```

Complete Code

```
package Assignment_26_Streaming
import org.apache.spark.storage.StorageLevel
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.{SparkConf, SparkContext}
import scala.collection.mutable.ArrayBuffer
object Assignment_26_2 {

    //ArrayBuffer to store list of offensive words in memory
    val wordList: ArrayBuffer[String] = ArrayBuffer.empty[String];

    def main(args: Array[String]): Unit = {
        println("hey Scala")

        //Create spark context object
```

```

    val conf = new
SparkConf().setMaster("local[*]").setAppName("Assignment_26_2")
    val sc = new SparkContext(conf)

    //Setting the log level
    sc.setLogLevel("WARN")
    printf("Spark context Created")

    //Define a function which returns the Real time set of Offensive
Words
    //Create a streaming context
    val ssc = new StreamingContext(sc,Seconds(10))
    //Below line will read the text entered after running netcat
utility
    val offensivelines = ssc.socketTextStream("localhost",
9000,StorageLevel.MEMORY_AND_DISK_SER)
    //Need to Read these lines and keep those in broad case variable
to send it to all
    val offensiveWords=
offensivelines.flatMap(_.split(",")).map(x=>(x))
    // storing offensive words in ArrayBuffer
    offensiveWords.foreachRDD(a => { a.foreach(f => {wordList +=
f}}}));

    //Reading lines entered by user
    val lines= ssc.socketTextStream("localhost",
9999,StorageLevel.MEMORY_AND_DISK_SER)
    //If word is duplicate then summing them on key (i.e. word)
    val wordCount = lines.flatMap(line => line.split(" ")).map(word =>
(word, 1)).reduceByKey(_+_ )

    //Getting word count of offensive words only
    val offensiveWordsRDD = wordCount.filter {x =>
matchWord(x._1)%2==1 };
    offensiveWordsRDD.print();

    ssc.start()
    ssc.awaitTermination()

}
/** * Filter Method for offensive words */
def matchWord(ln : String): Double=
{
    val lineWords = ln.trim.toLowerCase();
    var num: Double = 0;
    for(y<-wordList) {

```

```

        if(y.toLowerCase() == lineWords)
        { num = 1; return num; }
    }
    return num;
}
}

```

Output

List of Offensive Words on Port 9000

```

C:\netcat-1.11>nc64.exe -l -p 9000
dumb, idiot, lazy, horrible, irresponsible
bad, negative

```

Lines entered by user on another port 9999

```

C:\netcat-1.11>nc64.exe -l -p 9999
^C
C:\netcat-1.11>nc64.exe -l -p 9999
i dont ^C
C:\netcat-1.11>nc64.exe -l -p 9999
i dont like people with negative thinking
i dont want bad behavior
she is lazy
he is dump and lazy
she is horrible and irresponsible
he is also horrible

```

Now displaying the count of offensive words

Negative – 1 , Bad – 1 , Lazy – 2, Horrible – 2, Irresponsible - 1

Output after execution

```

-----
Time: 1528968270000 ms
-----

```

```

-----
Time: 1528968300000 ms
-----

```

```

(lazy,2)
(horrible,2)
(irresponsible,1)
(negative,1)
(bad,1)

```

```

-----
Time: 1528968330000 ms
-----

```