Spark - Java Documentation

Spark 1.6.1 - java 7 & java 8

Sujan Maddumage Don
Index: 985333
Ms in Computer Science
Maharishi University of Management

10/23/2016

Table of content

Solution	Spark - Java documentation		
Description			
Client	*unknown*	**Estimated Effort**	
Author	Sujan Duminda	**Drafted Date**	10/19/2016
Reviewed		**Reviewed Date**	
Version	0.1	**Last Modified Date**	10/19/2016
Status	Draft	**CR/SRS Ref**	

Name	**Date**	**Reason For Changes**	**Ver**
Sujan Duminda	10/19/2016	Initial Version	0.1

Overview

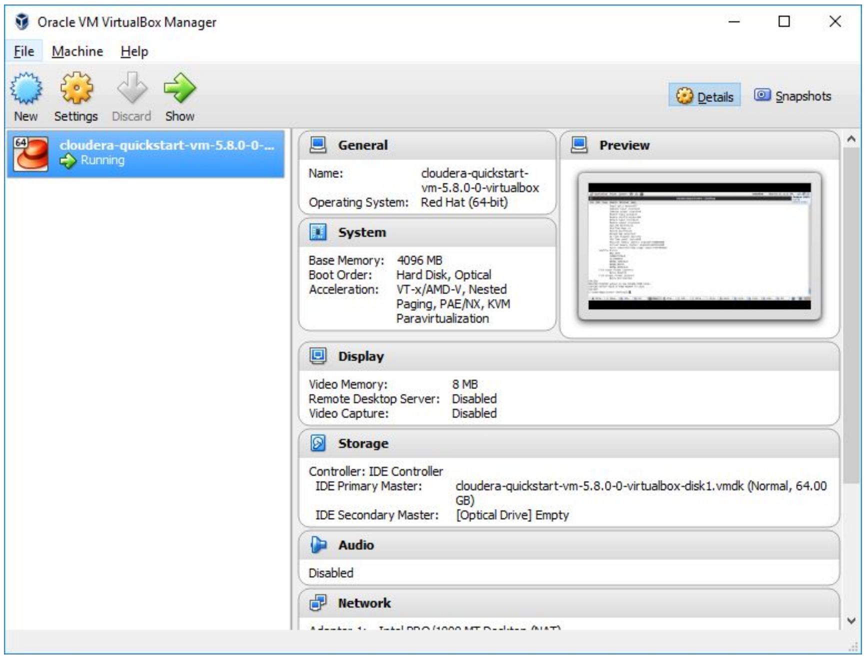
Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3. You can run Spark using its standalone cluster mode, on EC2, on Hadoop YARN, or on Apache Mesos. This documentation explains how to use spark with more than single map reduce program while demonstrating a sample spark project using java 7 and java 8.

Getting Started with Apache Spark

Spark is possible to create more than one map reduce function in same function.

Setup environment

Download cloudera virtual box version from below link
http://www.cloudera.com/downloads/manager/5-8-2.html version 5.8.2
Downloaded and Installed VirtualBox windows version from below link in order to run with linux virtual platform
https://www.virtualbox.org/wiki/Downloads version 5.1.8
Open Oracle virtualbox and drag the cloudera-quickstart-vm-5.8.0-0-virtualbox file into below window.



General
Name: cloudera-quickstart-vm-5.8.0-0-virtualbox
Operating System: Red Hat (64-bit)

System
Base Memory: 4096 MB
Boot Order: Hard Disk, Optical
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Display
Video Memory: 8 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage
Controller: IDE Controller
IDE Primary Master: cloudera-quickstart-vm-5.8.0-0-virtualbox-disk1.vmdk (Normal, 64.00 GB)
IDE Secondary Master: [Optical Drive] Empty

Audio
Disabled

Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT)

Figure 1 oracle virtual box

Run cloudera by clicking show button and virtual cloudera machine will be opened as below.

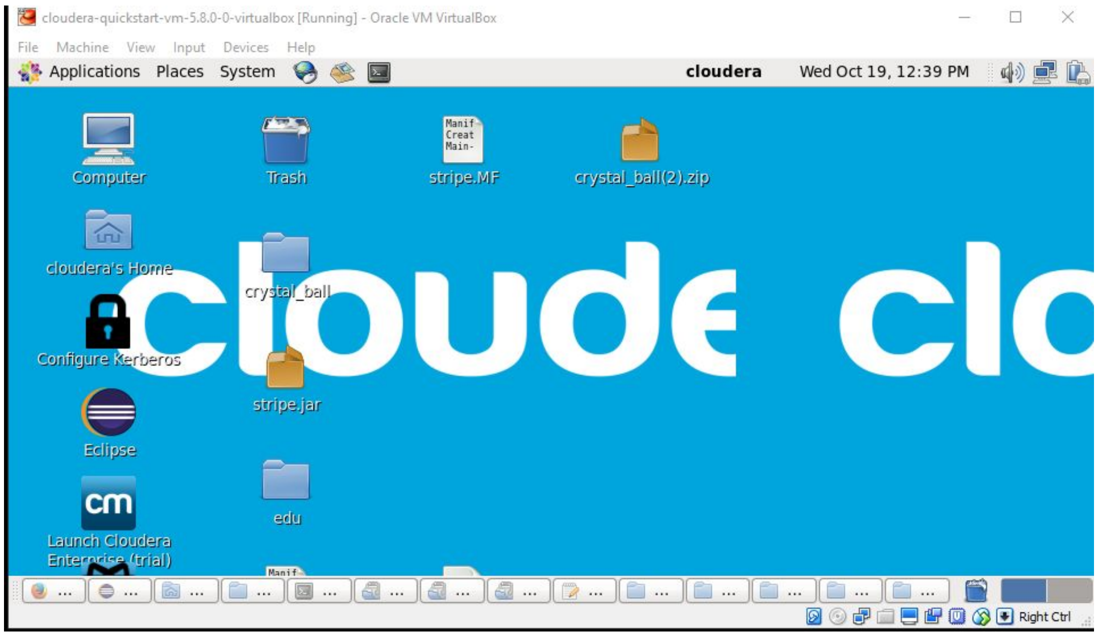
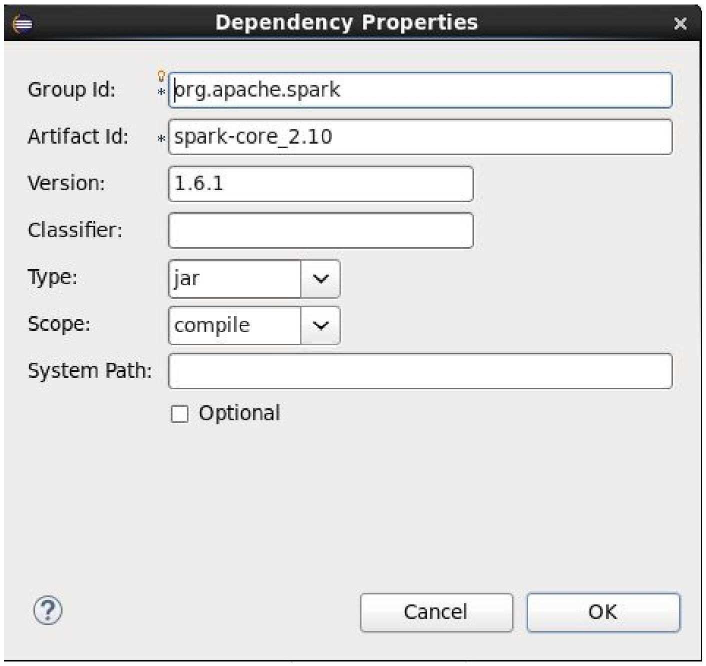

cloudera-quickstart-vm-5.8.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
cloudera
Wed Oct 19, 12:39 PM
Computer
Trash
stripe.MF
crystal_ball(2).zip
cloudera's Home
crystal_ball
Configure Kerberos
stripe.jar
Eclipse
edu
Launch Cloudera Enterprise (trial)
Right Ctrl

Figure 2 cloudera virtual platform

Create new project

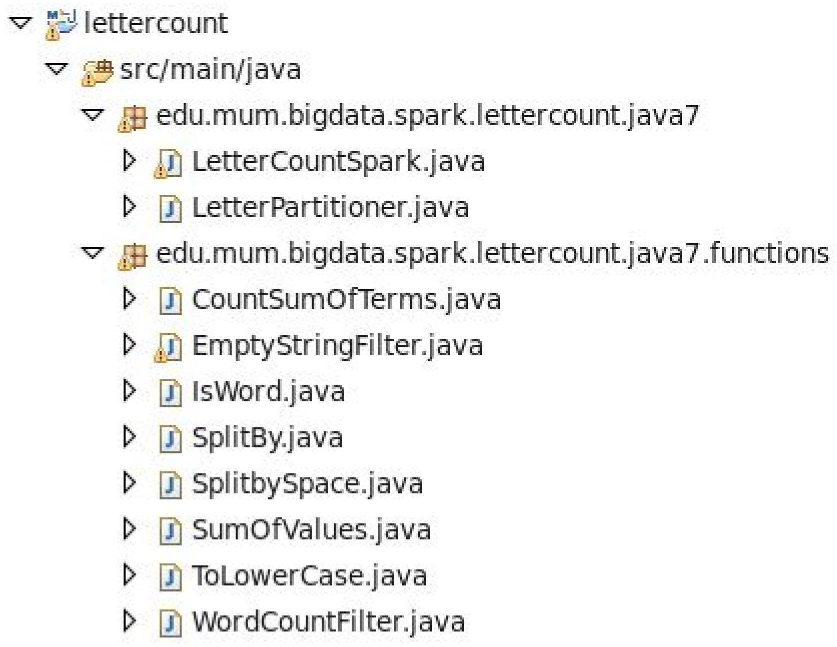
Open eclipse in cloudera and create a new maven project.
Open the pom.xml and add spark jar file detail under dependency.
Use spark jar properties as below.



Dependency Properties

Group Id: org.apache.spark
Artifact Id: spark-core_2.10
Version: 1.6.1
Classifier:
Type: jar
Scope: compile
System Path:
Optional

Project name : lettercount.
Project structure as below :

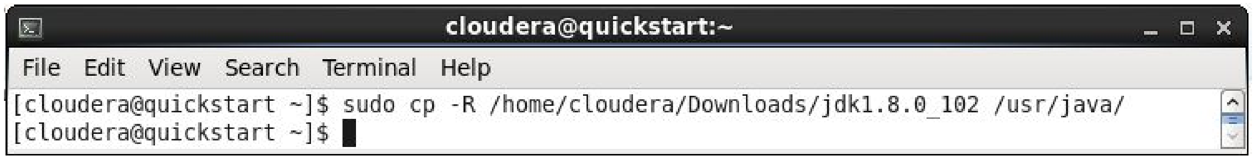


```
lettercount
  src/main/java
    edu.mum.bigdata.spark.lettercount.java7
      LetterCountSpark.java
      LetterPartitioner.java
    edu.mum.bigdata.spark.lettercount.java7.functions
      CountSumOfTerms.java
      EmptyStringFilter.java
      IsWord.java
      SplitBy.java
      SplitbySpace.java
      SumOfValues.java
      ToLowerCase.java
      WordCountFilter.java
```

Figure 3 package structure of the project

This project has been created for java 7 and java 8. If you use java 8 you have to change hadoop default java 7 to java 8. If you use java 7 hadoop is compatible to run with java 7 as default.

In order to instal java 8 below command is used.



```
[cloudera@quickstart ~]$ sudo cp -R /home/cloudera/Downloads/jdk1.8.0_102 /usr/java/
[cloudera@quickstart ~]$ 
```

.

Problem :

- Fine the word count from a log file.
- Filter all words with letters .
  eg : e4e 44 are not allowed
  Dog men are allowed.
- Among of above filtered words find how many unique words has each letter.
- Then output will produces with each letter with count that how many unique words with that letter. Here only count letters and omit all non alphabetics.

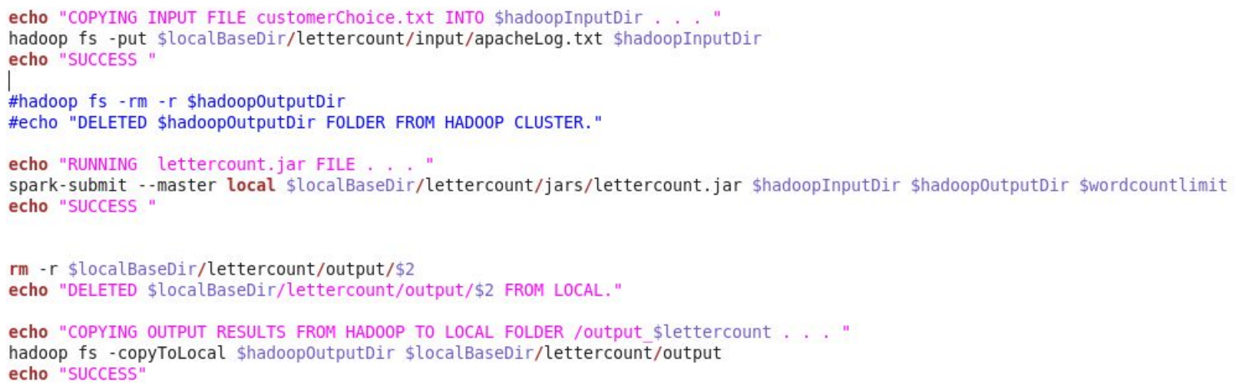
Solution

First I used java 8 and developed complete project. As it does not support with my current cloudera java version,I moved to java 7. There hierarchical logic will demonstrate my solution for letter count problem.

Batch file run

Run the batch file and give three arguments , input_filepath output_filepath wordcount_limit.
Shell command as below. Note : go to the project folder by terminal before run the script.
sh ./lettercount.sh input/logfile.txt output 20
Here argument 3 is wordcount limit that user needs to filter each word occurred less that that count.
Batch file task order is run as below.

```
localBaseDir=$(pwd)
hadoopBaseDir=/user/cloudera/project_spark_lettercount
hadoopOutputDir=$hadoopBaseDir/$2
hadoopInputDir=$hadoopBaseDir/$1
wordcountlimit=$3

echo "removing old jars created by user. . ."

echo "removing old jar file from local"
rm -r $localBaseDir/lettercount/jars/lettercount.jar
classpath=$localBaseDir/lettercount/target/classes/edu.mum.bigdata.spark.lettercount.LetterCountSpark
s
echo "CREATING JAR FILE $lettercount.jar . . ."
cd $localBaseDir/lettercount/target/classes
jar -cvfm ../../jars/lettercount.jar lettercount.MF *
echo "SUCCESS "

echo "CREATING PROJECT FOLDER . . ."
hadoop fs -rm -r $hadoopBaseDir
hadoop fs -mkdir $hadoopBaseDir
hadoop fs -mkdir $hadoopInputDir
echo "CREATED  FOLDER $hadoopInputDir"
```

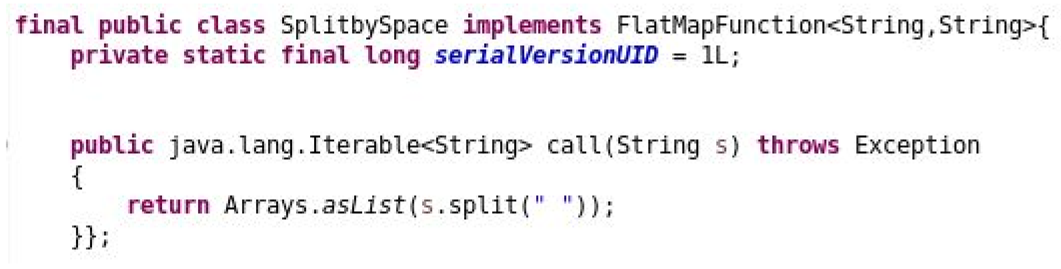


```
echo "COPYING INPUT FILE customerChoice.txt INTO $hadoopInputDir . . . "
hadoop fs -put $localBaseDir/lettercount/input/apacheLog.txt $hadoopInputDir
echo "SUCCESS "
|
#hadoop fs -rm -r $hadoopOutputDir
#echo "DELETED $hadoopOutputDir FOLDER FROM HADOOP CLUSTER."

echo "RUNNING  lettercount.jar FILE . . . "
spark-submit --master local $localBaseDir/lettercount/jars/lettercount.jar $hadoopInputDir $hadoopOutputDir $wordcountlimit
echo "SUCCESS "


rm -r $localBaseDir/lettercount/output/$2
echo "DELETED $localBaseDir/lettercount/output/$2 FROM LOCAL."

echo "COPYING OUTPUT RESULTS FROM HADOOP TO LOCAL FOLDER /output_$lettercount . . . "
hadoop fs -copyToLocal $hadoopOutputDir $localBaseDir/lettercount/output
echo "SUCCESS"
```

Java implementation

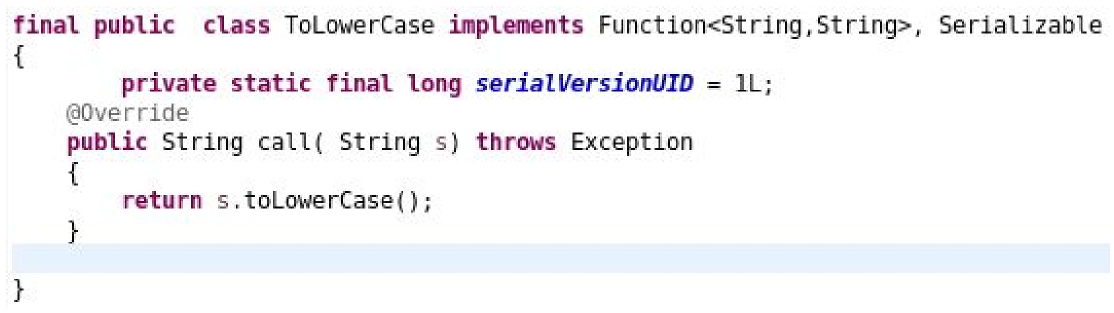
All the algorithms i used for this project can be summarized using below functions.

Step 1 -Read the log file and split all line by space to get words.



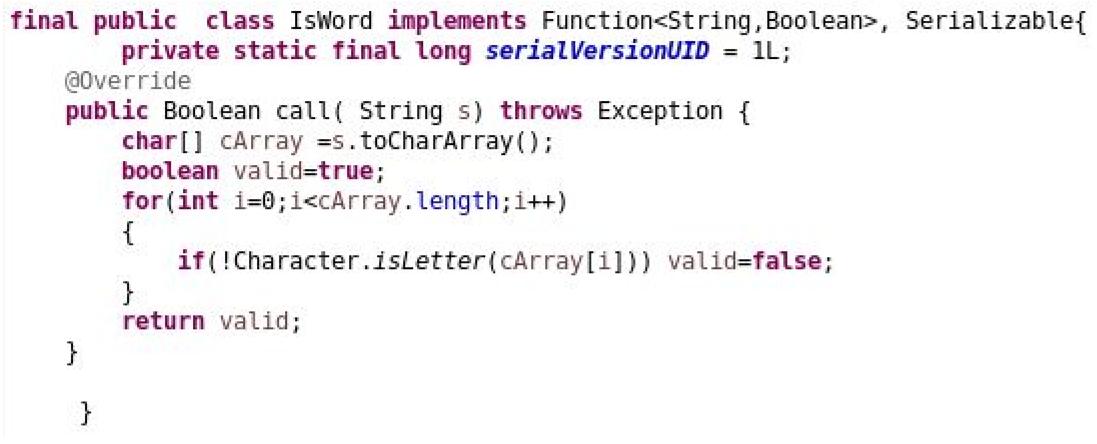
```
final public class SplitbySpace implements FlatMapFunction<String,String>{
    private static final long serialVersionUID = 1L;


    public java.lang.Iterable<String> call(String s) throws Exception
    {
        return Arrays.asList(s.split(" "));
    }};
```

Step 2- Convert all words to lower cases.



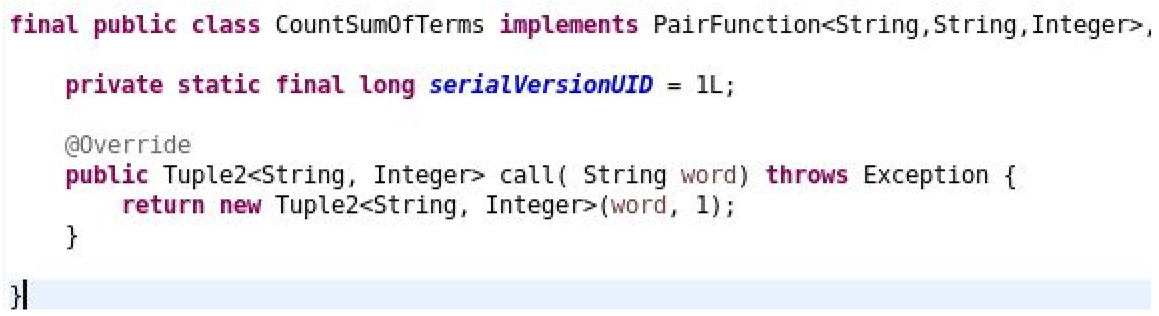
```
final public  class ToLowerCase implements Function<String,String>, Serializable
{
        private static final long serialVersionUID = 1L;
    @Override
    public String call( String s) throws Exception
    {
        return s.toLowerCase();
    }

}
```

Step 3 - Filter all words which includes only alphabets.



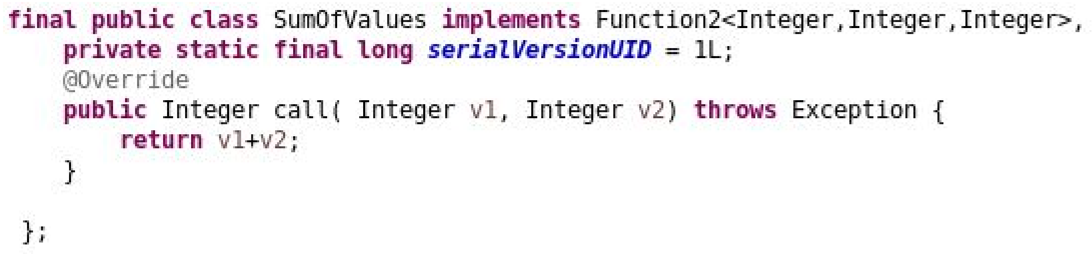
```java
final public  class IsWord implements Function<String,Boolean>, Serializable{
        private static final long serialVersionUID = 1L;
    @Override
    public Boolean call( String s) throws Exception {
        char[] cArray =s.toCharArray();
        boolean valid=true;
        for(int i=0;i<cArray.length;i++)
        {
            if(!Character.isLetter(cArray[i])) valid=false;
        }
        return valid;
    }

     }
```

Step -4 Then emit each word with value 1 from mapper.



```java
final public class CountSumOfTerms implements PairFunction<String,String,Integer>,

    private static final long serialVersionUID = 1L;

    @Override
    public Tuple2<String, Integer> call( String word) throws Exception {
        return new Tuple2<String, Integer>(word, 1);
    }

}
```

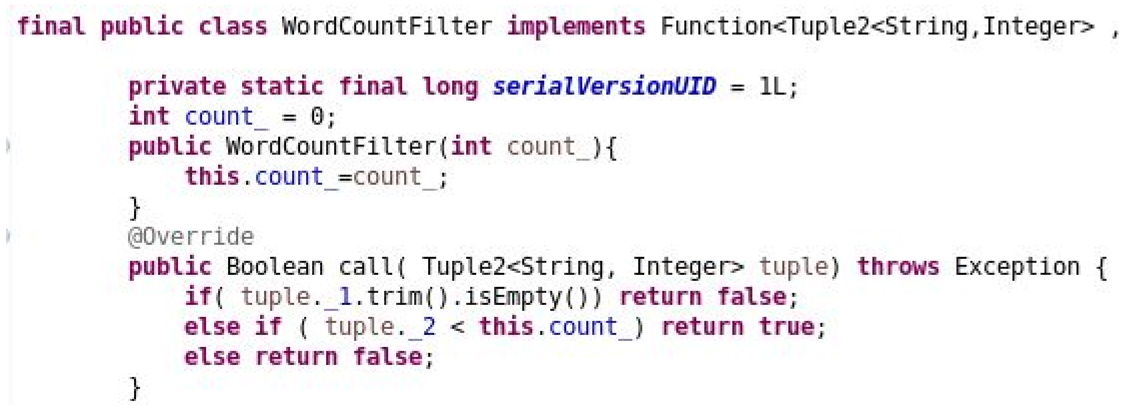
Step -5 Reducer 1 calculate sum of all words counts.



```java
final public class SumOfValues implements Function2<Integer,Integer,Integer>,
    private static final long serialVersionUID = 1L;
    @Override
    public Integer call( Integer v1, Integer v2) throws Exception {
        return v1+v2;
    }

 };
```

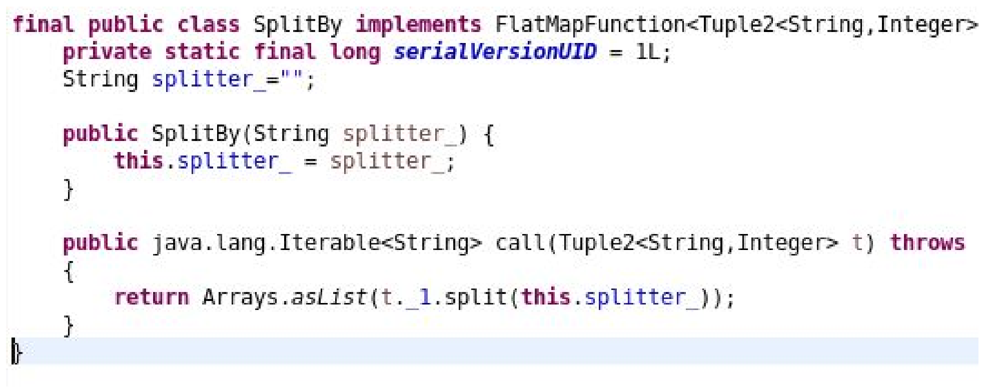
After this function ,System will keep all words counts list. That means one map reduce task is completed . Next it will continue with second map reduce task.

Step -7 Filter all words which less than user entered word count limit.



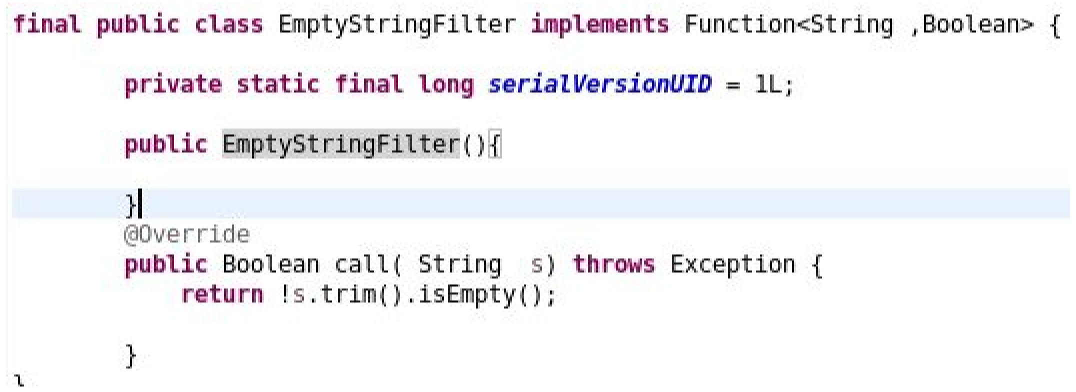
```
final public class WordCountFilter implements Function<Tuple2<String,Integer> ,

        private static final long serialVersionUID = 1L;
        int count_ = 0;
        public WordCountFilter(int count_){
            this.count_=count_;
        }
        @Override
        public Boolean call( Tuple2<String, Integer> tuple) throws Exception {
            if( tuple._1.trim().isEmpty()) return false;
            else if ( tuple._2 < this.count_) return true;
            else return false;
        }
```

Step- 8 Split each words by empty string to splits all letters.



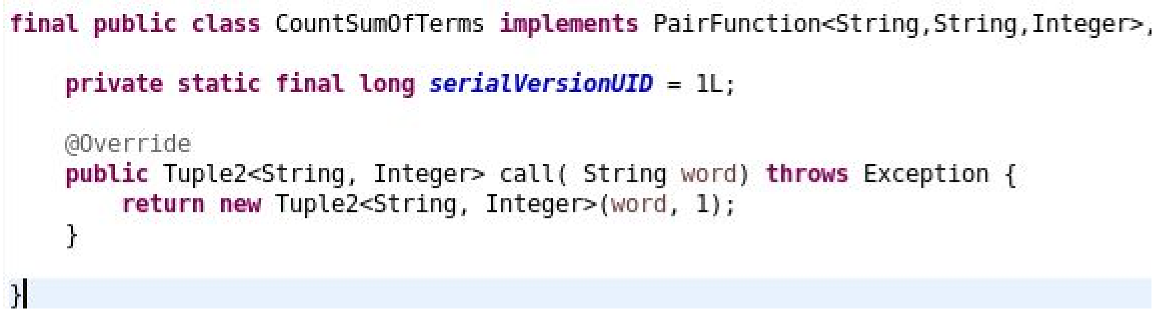
```
final public class SplitBy implements FlatMapFunction<Tuple2<String,Integer>
    private static final long serialVersionUID = 1L;
    String splitter_="";

    public SplitBy(String splitter_) {
        this.splitter_ = splitter_;
    }

    public java.lang.Iterable<String> call(Tuple2<String,Integer> t) throws
    {
        return Arrays.asList(t._1.split(this.splitter_));
    }
}
```

Step - 9Filter all letters except space.



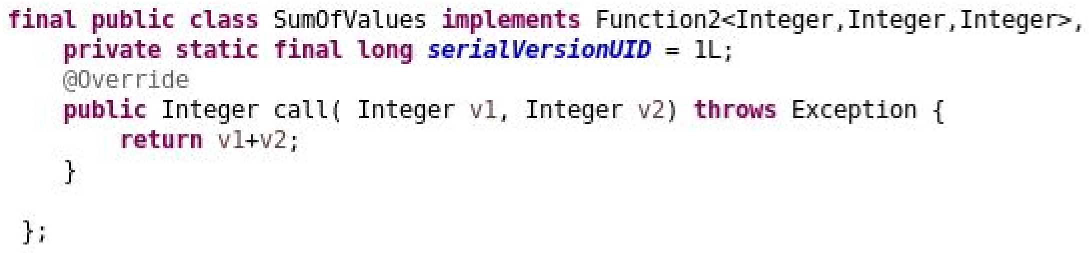
```
final public class EmptyStringFilter implements Function<String ,Boolean> {

        private static final long serialVersionUID = 1L;

        public EmptyStringFilter(){

        }
        @Override
        public Boolean call( String  s) throws Exception {
            return !s.trim().isEmpty();

        }
}
```

Step 10 -Get the count of of each letters from each map.



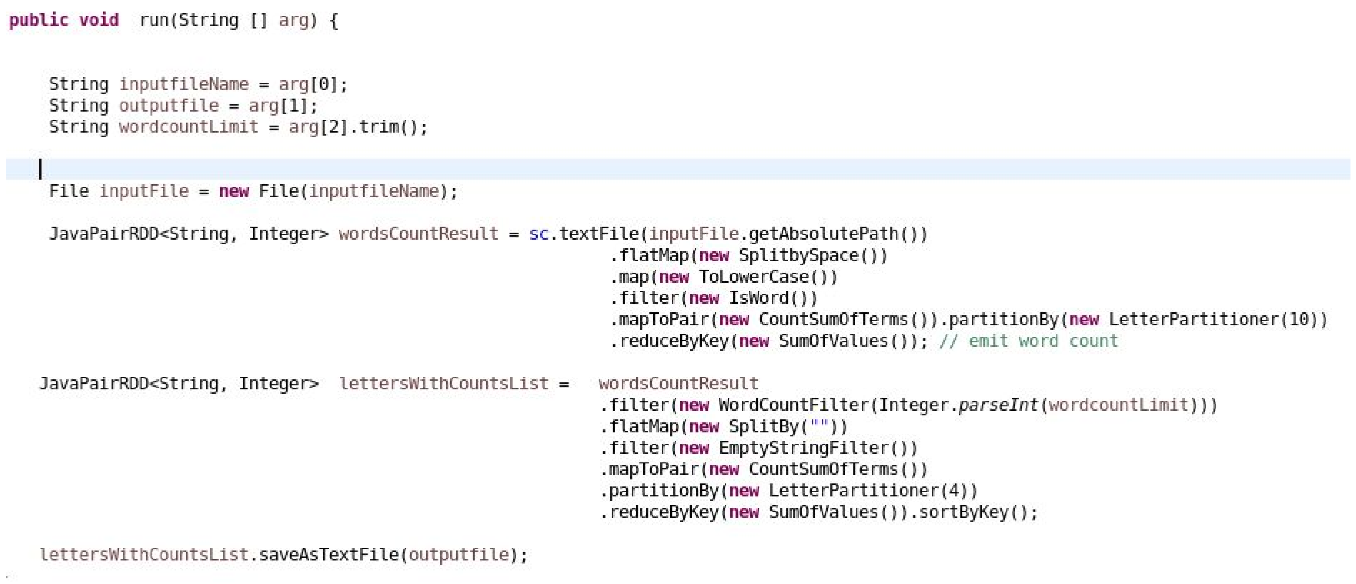
```
final public class CountSumOfTerms implements PairFunction<String,String,Integer>,

    private static final long serialVersionUID = 1L;

    @Override
    public Tuple2<String, Integer> call( String word) throws Exception {
        return new Tuple2<String, Integer>(word, 1);
    }

}
```

Step - 11 Calculate all count s values of each letters. Here i used same reducer which i used before.



```
final public class SumOfValues implements Function2<Integer,Integer,Integer>,
    private static final long serialVersionUID = 1L;
    @Override
    public Integer call( Integer v1, Integer v2) throws Exception {
        return v1+v2;
    }

 };
```

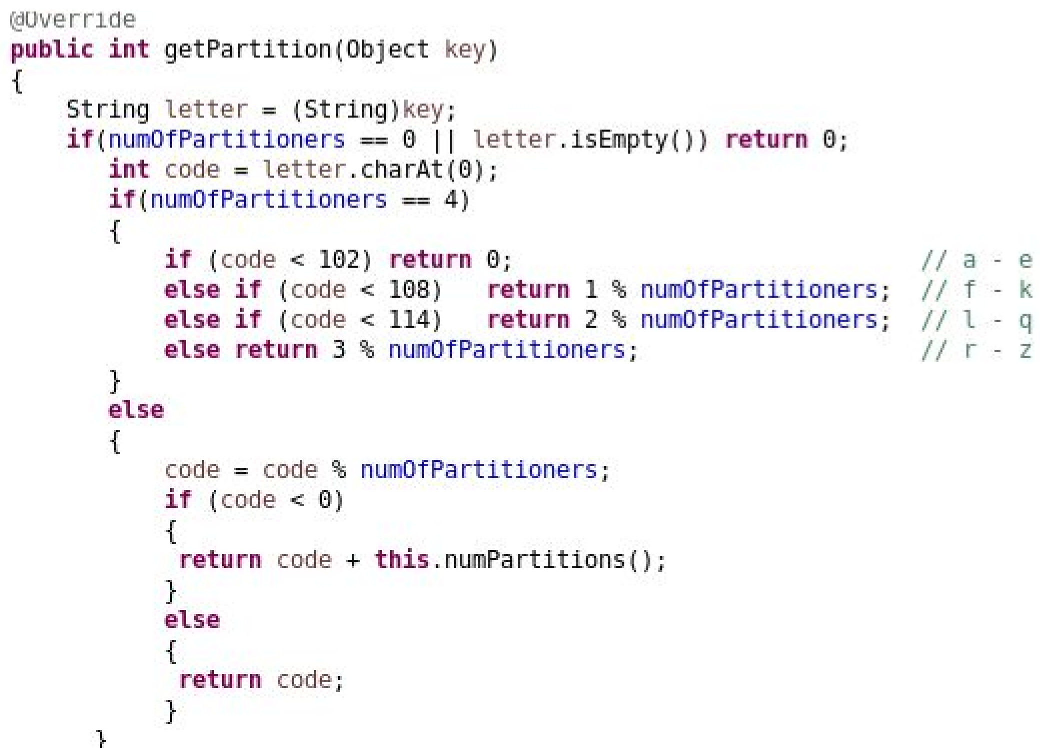
Main function



```
public void  run(String [] arg) {


    String inputfileName = arg[0];
    String outputfile = arg[1];
    String wordcountLimit = arg[2].trim();


    File inputFile = new File(inputfileName);

    JavaPairRDD<String, Integer> wordsCountResult = sc.textFile(inputFile.getAbsolutePath())
                                                        .flatMap(new SplitbySpace())
                                                        .map(new ToLowerCase())
                                                        .filter(new IsWord())
                                                        .mapToPair(new CountSumOfTerms()).partitionBy(new LetterPartitioner(10))
                                                        .reduceByKey(new SumOfValues()); // emit word count

   JavaPairRDD<String, Integer>  lettersWithCountsList =   wordsCountResult
                                                        .filter(new WordCountFilter(Integer.parseInt(wordcountLimit)))
                                                        .flatMap(new SplitBy(""))
                                                        .filter(new EmptyStringFilter())
                                                        .mapToPair(new CountSumOfTerms())
                                                        .partitionBy(new LetterPartitioner(4))
                                                        .reduceByKey(new SumOfValues()).sortByKey();

   lettersWithCountsList.saveAsTextFile(outputfile);
```

Partitioner

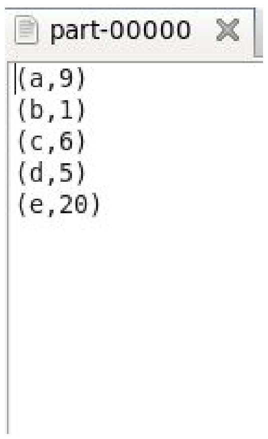
Letterpartitioner is the object which used as main partitioner for this project. Constructor argument is set as number of partitions.
If that value is 4 then it run as custom partitioner and all other values it run as hash partitioner.



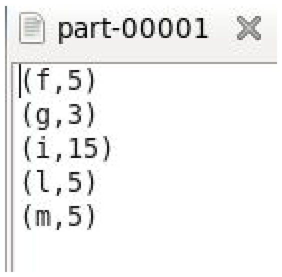
```
@Override
public int getPartition(Object key)
{
    String letter = (String)key;
    if(numOfPartitioners == 0 || letter.isEmpty()) return 0;
       int code = letter.charAt(0);
       if(numOfPartitioners == 4)
       {
           if (code < 102) return 0;                                  // a - e
           else if (code < 108)   return 1 % numOfPartitioners;  // f - k
           else if (code < 114)   return 2 % numOfPartitioners;  // l - q
           else return 3 % numOfPartitioners;                         // r - z
       }
       else
       {
           code = code % numOfPartitioners;
           if (code < 0)
           {
            return code + this.numPartitions();
           }
           else
           {
            return code;
           }
      }
```

Output results sample from spark.

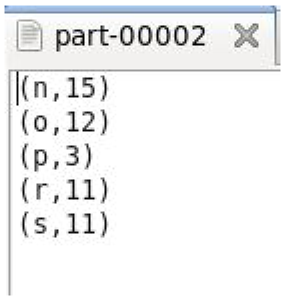
Since there are 4 partitioners , output file also has 4 files as below.


part-00000

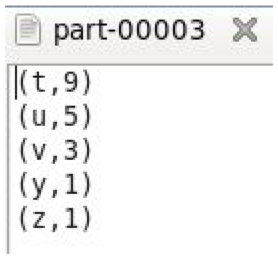
```
(a,9)
(b,1)
(c,6)
(d,5)
(e,20)
```


part-00001

```
(f,5)
(g,3)
(i,15)
(l,5)
(m,5)
```


part-00002

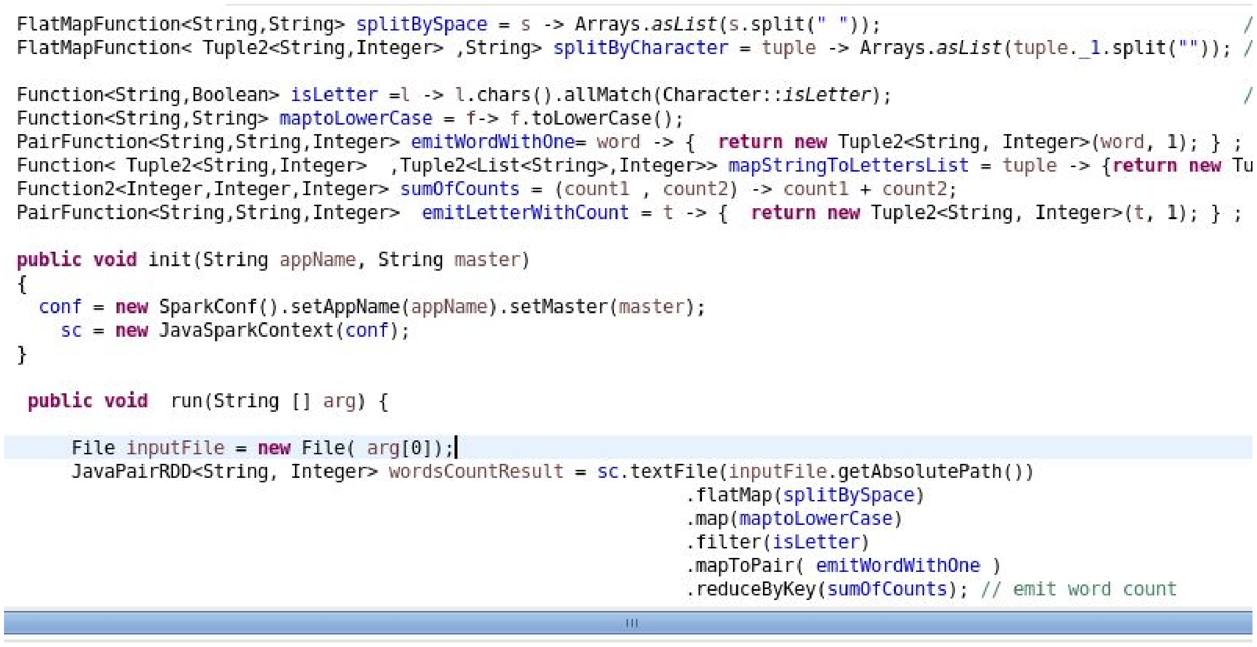
```
(n,15)
(o,12)
(p,3)
(r,11)
(s,11)
```


part-00003

```
(t,9)
(u,5)
(v,3)
(y,1)
(z,1)
```

Java 8

Java 8 functions are defined as below. As java 8 not compatible with hadoop in cloudera ,here i have not created batch file. However project is run on local machine.
All functions are implemented as lambda expression.



```
FlatMapFunction<String,String> splitBySpace = s -> Arrays.asList(s.split(" "));                      /
FlatMapFunction< Tuple2<String,Integer> ,String> splitByCharacter = tuple -> Arrays.asList(tuple._1.split("")); /

Function<String,Boolean> isLetter =l -> l.chars().allMatch(Character::isLetter);                     /
Function<String,String> maptoLowerCase = f-> f.toLowerCase();
PairFunction<String,String,Integer> emitWordWithOne= word -> {  return new Tuple2<String, Integer>(word, 1); } ;
Function< Tuple2<String,Integer>  ,Tuple2<List<String>,Integer>> mapStringToLettersList = tuple -> {return new Tu
Function2<Integer,Integer,Integer> sumOfCounts = (count1 , count2) -> count1 + count2;
PairFunction<String,String,Integer>  emitLetterWithCount = t -> {  return new Tuple2<String, Integer>(t, 1); } ;

public void init(String appName, String master)
{
  conf = new SparkConf().setAppName(appName).setMaster(master);
    sc = new JavaSparkContext(conf);
}

 public void  run(String [] arg) {

     File inputFile = new File( arg[0]);
     JavaPairRDD<String, Integer> wordsCountResult = sc.textFile(inputFile.getAbsolutePath())
                                                         .flatMap(splitBySpace)
                                                         .map(maptoLowerCase)
                                                         .filter(isLetter)
                                                         .mapToPair( emitWordWithOne )
                                                         .reduceByKey(sumOfCounts); // emit word count
```

Technical Details

Required technology

Linux platform cloudera 5.8.2
Oracle virtual box 5.1.8
Java 7
Eclipse platform
Spark 1.6.1 libraries

Review Comments