Big Data Systems
Assignment: Apache Spark

Spark Core & Spark SQL Library

by

Dimitrios-Gerasimos Anastasatos, AM: BAFT1702

Athens, May 2018

- Question 1a:

Scala Program:

```
//Create an RDD file from the file airports (already in HDFS: "/user/b-
//analytics/assignment/airports.text"). This is created using the spark
//context ".textFile". This operation is a transformation, so nothing
//actually happens. We're just telling it //that we want to create an
//airRDD. This was an RDD transformation, thus it returned a pointer
//to a RDD, which we have named as airRDD.
val airRDD = sc.textFile("/user/b-analytics/assignment/airports.text")

//Now, we use the filter transformation to return a new RDD with a
//subset of the items in the file. Specifically, we keep only the Greek
//airports
val greekRDD = airRDD.filter(line => line.contains("Greece"))

//To parse out the airport's name, the city's name and the IATA/FAA
//code, we create a new RDD by splitting the lines of the RDD using the
//comma as the delimiter.
val greekParse = greekRDD.map(_.split(","))

//We map each line of the original text to the 3 corresponding columns
//containing the airport's name, the city's name and the IATA/FAA code
val question_1a = greekParse.map(vals=>(vals(1),vals(2), vals(4)))

//Finally, we print an array with the first 3 elements of the previous
//RDD (RDD action: take(n))
question_1a.take(3).foreach(println)
```

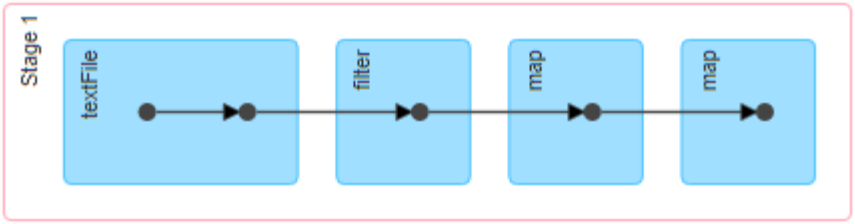
Output:

```
scala> question_1a.take(3).foreach(println)
("Alexion","Porto Heli","PKH")
("Andravida","Andravida","PYR")
("Agrinion","Agrinion","AGQ")
```

Direct Acyclic Graph (DAG):

```
scala> question_1a.toDebugString
res5: String =
(2) MapPartitionsRDD[10] at map at <console>:33 []
 |  MapPartitionsRDD[9] at map at <console>:31 []
 |  MapPartitionsRDD[8] at filter at <console>:29 []
 |  /user/b-analytics/assignment/airports.text MapPartitionsRDD[7] at textFile a
t <console>:27 []
 |  /user/b-analytics/assignment/airports.text HadoopRDD[6] at textFile at <cons
ole>:27 []
```

DAG Visualization:



- Question 1b:

Scala Program:

```
//We use the same transformation to create the airRDD file
val airRDD = sc.textFile("/user/b-analytics/assignment/airports.text")

//We use 2 cascased filter tranformations to reduce the dataset to the
//wanted latitude range
val airRDD2 =
airRDD.filter(_.split(",")(6).toDouble>37).filter(_.split(",")(6).toDou
ble<39)

//We map (RDD transformation) the filtered dataset by splitting its
//lines using the comma as the delimiter and then we use a second
//mapping transformation to the values of the 3 wanted columns.
val question_1b =
airRDD2.map(_.split(",")).map(vals=>(vals(1),vals(2),vals(6)))

//We take the first 3 elements of the last RDD
question_1b.take(3).foreach(println)
```

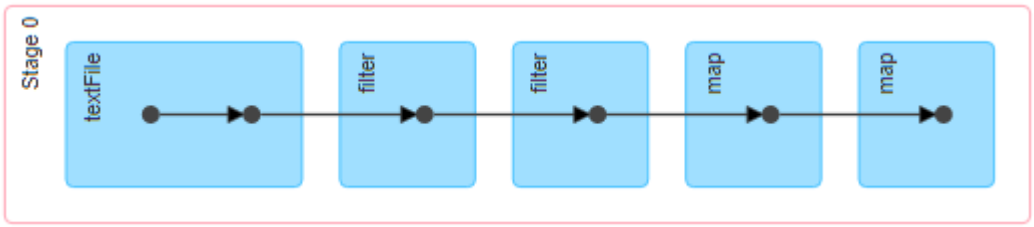
Output:

```
scala> question_1b.take(3).foreach(println)
18/05/13 21:30:50 WARN cluster.YarnScheduler: Initial job has not accepted any r
esources; check your cluster UI to ensure that workers are registered and have s
ufficient resources
("Sidi Ahmed Air Base","Bizerte",37.245447)
("Albacete","Albacete",38.948528)
("Alicante","Alicante",38.282169)
```

DAG:

```
scala> print(question_1b.toDebugString)
(2) MapPartitionsRDD[5] at map at <console>:31 []
 |  MapPartitionsRDD[4] at map at <console>:31 []
 |  MapPartitionsRDD[3] at filter at <console>:29 []
 |  MapPartitionsRDD[2] at filter at <console>:29 []
 |  /user/b-analytics/assignment/airports.text MapPartitionsRDD[1] at textFile a
t <console>:27 []
 |  /user/b-analytics/assignment/airports.text HadoopRDD[0] at textFile at <cons
ole>:27 []
```

DAG Visualization:



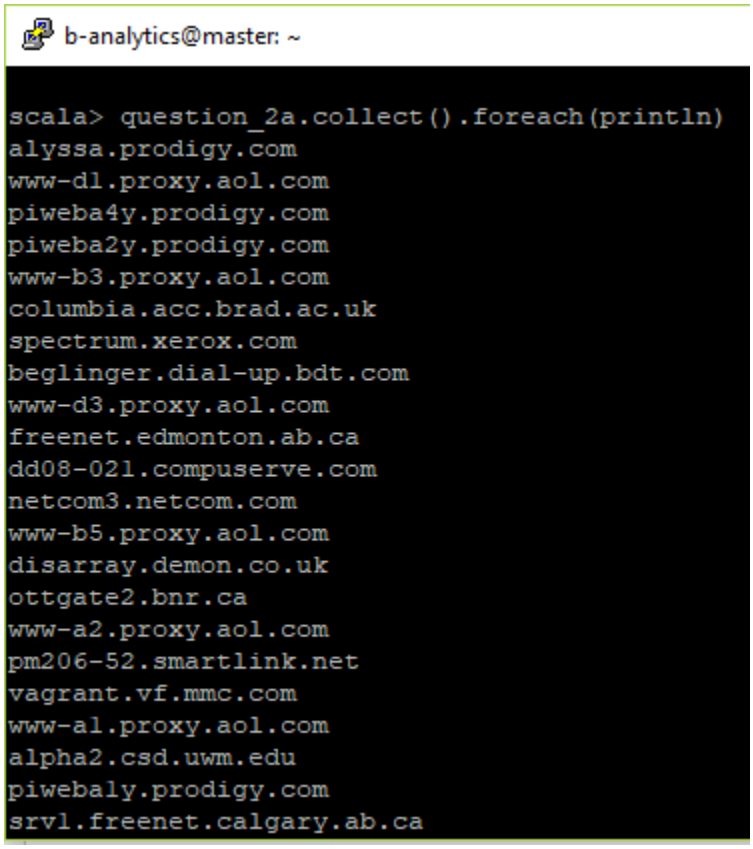
- Question 2a

Scala Program:

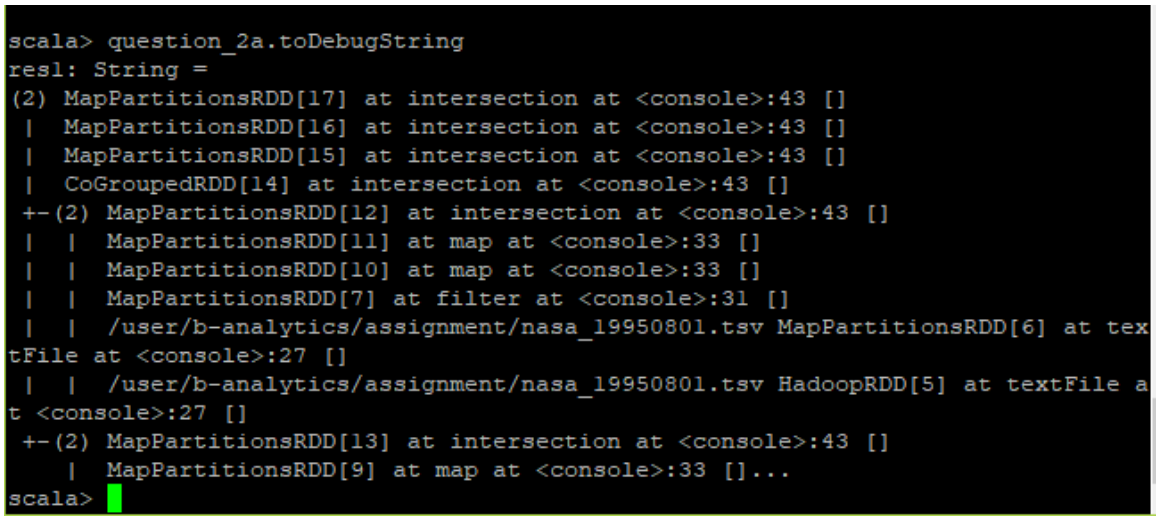
```
//Use transformation to create the RDD file about July
val nasaJuly = sc.textFile("/user/b-
analytics/assignment/nasa_19950701.tsv")

//RDD Action to return the first element of the dataset
val header1 = nasaJuly.first()

//Filter transformation to remove the first element, that is the header
val nasaJuly1 = nasaJuly.filter(row => row != header1)

//Create a new RDD by first splitting the lines of the RDD using the
//tab as the delimiter, and then by mapping the dataset to
//the column containing the hosts
val nasaJuly2 = nasaJuly1.map(_.split( "\t")).map(vals=> vals(0))

//RDD Transformation to create the RDD file about August
val nasaAugust = sc.textFile("/user/b-
analytics/assignment/nasa_19950801.tsv")

//RDD Action to return the first element of the dataset
val header2 = nasaAugust.first()

//Filter transformation to remove the first element, that is the header
val nasaAugust1 = nasaAugust.filter(row => row != header2)

//Create a new RDD about July by first splitting the lines of the RDD
//using the tab as the delimiter, and then by mapping the dataset to
//the column containing the hosts
val nasaJuly2 = nasaJuly1.map(_.split( "\t")).map(vals=> vals(0))

//Create a new RDD about August by first splitting the lines of the RDD
//using the tab as the delimiter, and then by mapping the dataset to
//the column containing the hosts
val nasaAugust2 = nasaAugust1.map(_.split( "\t")).map(vals=> vals(0))

//Use the intersection tranformation to create a new RDD containing
//only the hosts which are accessed on both datasets
val question_2a = nasaAugust2.intersection(nasaJuly2)

//Use the collect Action to return the previous dataset
```

```
question_2a.collect().foreach(println)
```

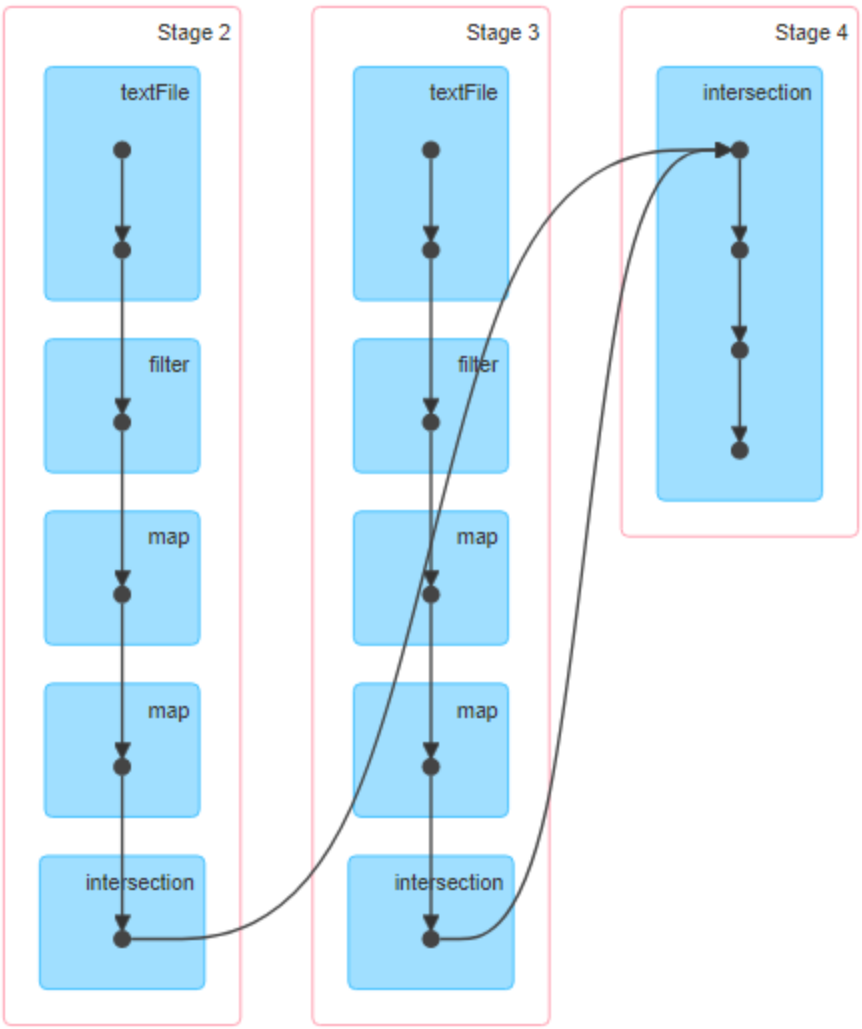
Output:



DAG:



DAG Visualization:



- Question 2b

Scala Program:

```
//RDD transformation to create RDD file
val wordsRDD = sc.textFile("/user/b-
analytics/assignment/word_count.text")

//Use flatMap Transformation  to split the lines into words
val wordsRDD1 = wordsRDD.flatMap(x => x.split(" "))

//Chain together map transformation and reduceByKey tranformation in
//order to count the occurrences for each word in file
val wordCounts = wordsRDD1.map(word => (word, 1)).reduceByKey(_+_)

//RDD tranformation to sort the words by the number of occurrence(value
//in the key-value pair) and by descending order.
val sortedRdd = wordCounts.sortBy(_._2, false)

//Use the collect Action to return the previous dataset
```

```
sortedRdd.collect().foreach(println)
```

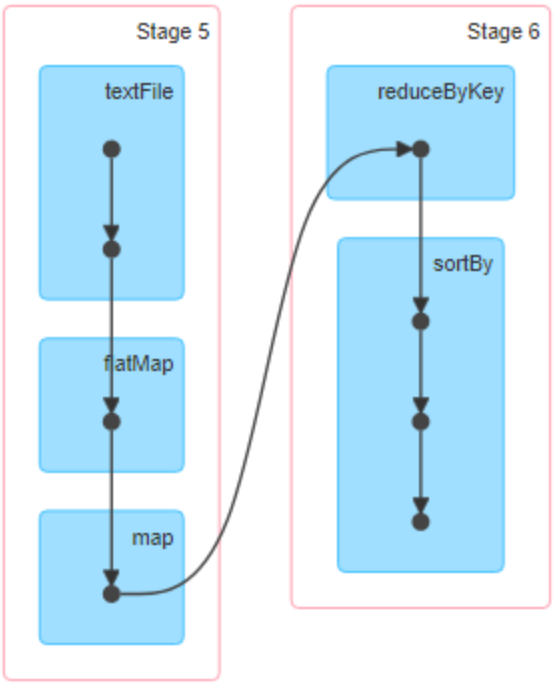
Output:

```
scala> sortedRdd.collect().foreach(println)
(the,71)
(of,33)
(in,21)
(and,21)
(New,20)
(York,17)
(to,17)
(a,11)
(The,10)
(from,8)
(was,8)
(,7)
(City,7)
(became,6)
(first,5)
(state,5)
(as,4)
(around,4)
```

DAG:

```
scala> sortedRdd.toDebugString
res3: String =
(2) MapPartitionsRDD[27] at sortBy at <console>:33 []
 |  ShuffledRDD[26] at sortBy at <console>:33 []
 +-(2) MapPartitionsRDD[23] at sortBy at <console>:33 []
    |  ShuffledRDD[22] at reduceByKey at <console>:31 []
    +-(2) MapPartitionsRDD[21] at map at <console>:31 []
       |  MapPartitionsRDD[20] at flatMap at <console>:29 []
       |  /user/b-analytics/assignment/word_count.text MapPartitionsRDD[19] at t
extFile at <console>:27 []
       |  /user/b-analytics/assignment/word_count.text HadoopRDD[18] at textFile
 at <console>:27 []
```

DAG Visualization:



- Question 3a

Scala Program:

```
//RDD transformation to create RDD file
val estate = sc.textFile("/user/b-analytics/assignment/Real.csv")

//RDD Action to return the first element of the dataset
val header = estate.first()

//Filter transformation to remove the first element, that is the header
//and then map transformation by splitting the lines of the RDD using
//the comma as the delimiter
val estate1 = estate.filter(row => row != header).map(_.split(","))

//Map (RDD tranformation) the lines to key-value pairs. Key is the
//values of the 4th column (Bedrooms) and the value is a second key-
//value pair with key 1 and value the values of the 3rd column (Price)
val estate2 =
estate1.map(vals=>(vals(3).trim.toInt,(1,vals(2).toDouble.toInt)))

//Create an RDD that aggregates the contents of the previous RDD into
//records of the form: (word, (number_of_houses, total_price_of_house))
val estate3 = estate2.aggregateByKey((0,0))(
(x,y) => (x._1 + y._1, x._2 + y._2),
(rdd1,rdd2) => (rdd1._1+rdd2._1 , rdd1._2+rdd2._2))

//Compute the average price per bedroom by mapping the lines of the
//previous RDD to a key-value pair with key the number of bedrooms and
//value the average price
```

```
val estate4 = estate3.map(x=>(x._1, (x._2)._2/(x._2)._1))

//Use the collect Action to return the previous dataset
estate4.collect().foreach(println)
```

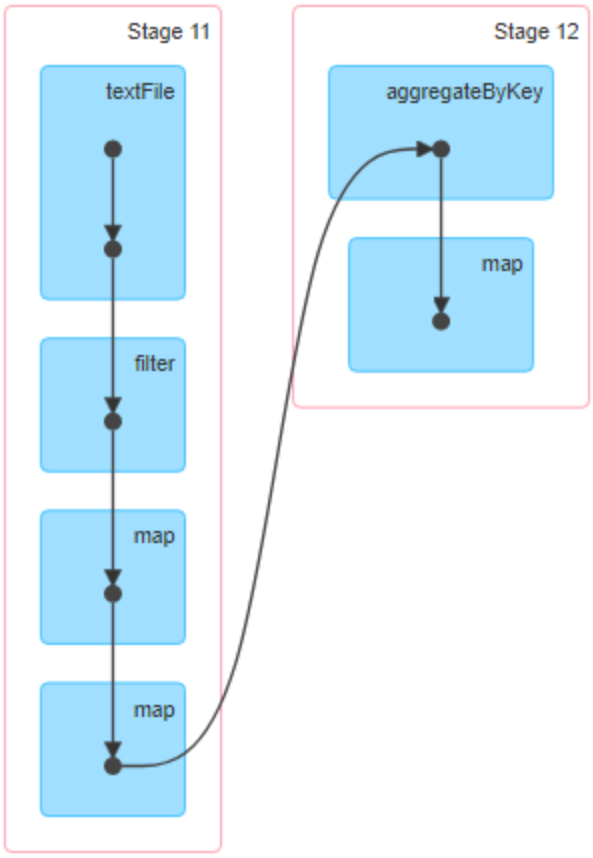
Output:

```
scala> estate4.collect().foreach(println)
(4,483475)
(0,293450)
(6,603225)
(10,699000)
(2,266356)
(1,169981)
(3,359062)
(7,325000)
(5,657858)
```

DAG:

```
scala> estate4.toDebugString
res5: String =
(2) MapPartitionsRDD[34] at map at <console>:37 []
 |  ShuffledRDD[33] at aggregateByKey at <console>:35 []
 +-(2) MapPartitionsRDD[32] at map at <console>:33 []
    |  MapPartitionsRDD[31] at map at <console>:31 []
    |  MapPartitionsRDD[30] at filter at <console>:31 []
    |  /user/b-analytics/assignment/Real.csv MapPartitionsRDD[29] at textFile at
 <console>:27 []
    |  /user/b-analytics/assignment/Real.csv HadoopRDD[28] at textFile at <conso
le>:27 []
```

DAG Visualization:



- Question 3b

Scala Program:

```
//Define the SparkSQL context. Do so by creating it from an existing
//SparkContext
val sqlContext = new org.apache.spark.sql.SQLContext(sc)

//Import a library for creating a SchemaRDD
import sqlContext.implicits._

//Define the schema of the table; The arguments of the case class
//become the names of the columns
case class house(Location: String, Price: Double, PriceSQ: Double)

//Create the RDD of the house object; estate1 RDD object is from the
//previous question
val houseRDD = estate1.map(vals=>house(vals(1), vals(2).toDouble,
vals(6).toDouble))

//Register the RDD as a table
houseRDD.toDF().registerTempTable("houseRDD")

//Run SQL statement using the sql method provided by the SQLContext
```

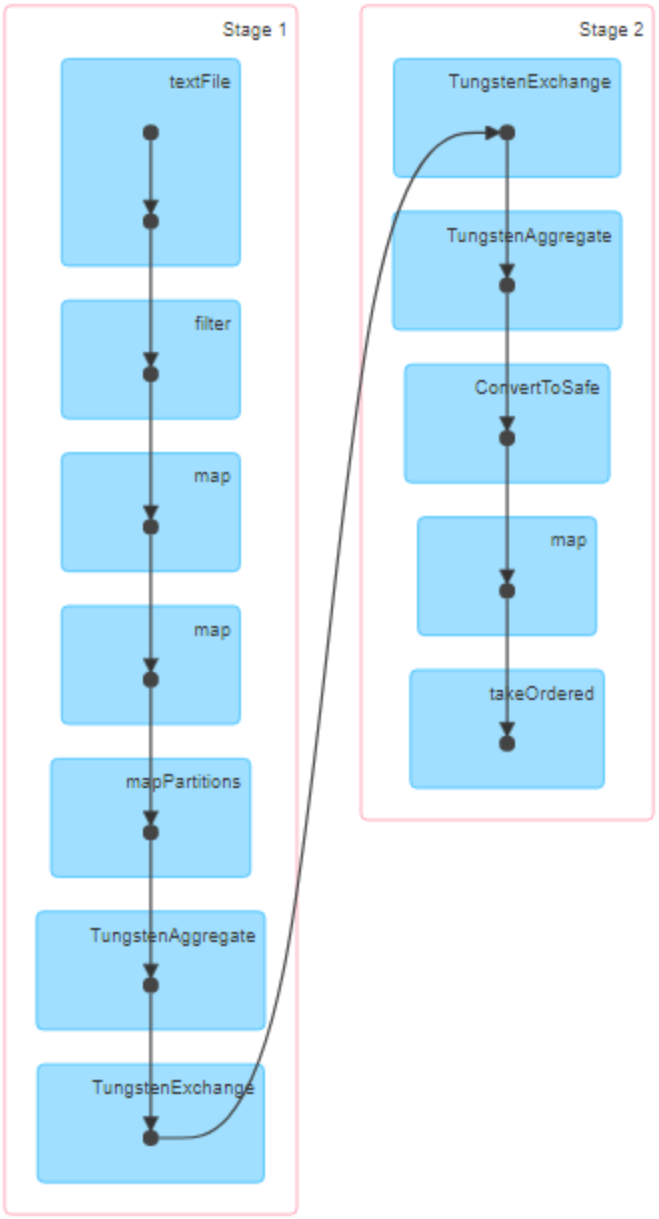
```
val question_3b = sqlContext.sql("SELECT Location, AVG(PriceSQ) AS
Average_PriceSQ, MAX(Price) AS Max_Price FROM houseRDD GROUP BY
Location ORDER BY Average_PriceSQ DESC")

//Displays the top 20 rows of DataFrame in a tabular form
question_3b.show()
```

Output:

b-analytics@master: ~

```
scala> question_3b.show()
+---------------+------------------+---------+
|       Location|   Average_PriceSQ|Max_Price|
+---------------+------------------+---------+
|         Oceano|           1144.64|1195000.0|
|        Bradley|            606.06|1600000.0|
|    Avila Beach| 566.5500000000001|1999000.0|
|        Cambria| 491.9558333333334|2995000.0|
|    Pismo Beach|462.28416666666664|1799000.0|
|San Luis Obispo|458.91333333333336|2369000.0|
|     Santa Ynez|391.33000000000004|1395000.0|
|        Cayucos|             386.6|1500000.0|
|        Cayucos|            385.11| 695000.0|
|      Morro Bay|374.13750000000005| 982800.0|
|  Arroyo Grande| 361.4208333333333|5499000.0|
|    Pismo Beach|         357.01125| 920000.0|
|        Cambria| 347.5544444444444| 699900.0|
|      Morro Bay|345.90538461538466|1100000.0|
|        Creston|            322.75| 549000.0|
|    Out Of Area|            314.47|1195000.0|
|   Grover Beach|            308.78| 999000.0|
|San Luis Obispo|291.09357142857147| 890000.0|
|     Atascadero|            285.76| 995000.0|
|       Lockwood|            283.33| 425000.0|
+---------------+------------------+---------+
only showing top 20 rows
```

DAG Visualization:


Stage 1
textFile
filter
map
map
mapPartitions
TungstenAggregate
TungstenExchange
Stage 2
TungstenExchange
TungstenAggregate
ConvertToSafe
map
takeOrdered