DoubleRDDs and basic statistical measures

DoubleRDDs

- Spark provides specific actions for a specific numerical type of RDD called JavaDoubleRDD
- JavaDoubleRDD is an RDD of doubles
  - However, it is different from JavaRDD<Double>
  - Even if they contains the same type of objects
- On JavaDoubleRDDs, the following actions are also available
  - sum(), mean(), stdev(), variance(), max(), min(),..

DoubleRDDs

- A generic JavaRDD<T> containing elements of type T can be transformed in a JavaDoubleRDD by using two specific transformations
  - mapToDouble
  - flatMapToDouble
- mapToDouble and flatMapToDouble operate similarly to map and flatMap, but they return a JavaDoubleRDD

DoubleRDDs

- JavaDoubleRDDs can be created also by using the **JavaDoubleRDD parallelizeDoubles(java.util.List<Double> list)** method of the **JavaSparkContext** class

MapToDouble transformation

MapToDouble transformation

- Goal
  - The mapToDouble transformation is used to create a new DoubleRDD by applying a function on each element of the "input" RDD
  - The new RDD contains one element **y** for each element **x** of the "input" RDD
  - The value of **y** is obtained by applying a user defined function **f** on **x**
    - **y**= **f**(**x**)
  - The data type of **y is always double**

MapToDouble transformation

- Method
  - The mapToDouble transformation is based on the **JavaDoubleRDD mapToDouble(DoubleFunction<T>)** method of the **JavaRDD<T>** class
  - An object of a class implementing the DoubleFunction<T> interface is passed to the mapToDouble method
    - The **public double call(T element)** method of the DoubleFunction<T> interface must be implemented
      - It contains the code that is applied on each element of the "input" RDD to create the double values of the returned DoubleRDD
        - For each element of the "input" RDD one single double is returned by the call method

MapToDouble transformation: Example

- Create an RDD from a textual file containing the surnames of a list of users
  - Each line of the file contains one surname
- Create a new DoubleRDD containing the lengths of the input surnames

MapToDouble transformation: Example

```
// Read the content of the input textual file
JavaRDD<String> surnamesRDD = sc.textFile("surnames.txt");

// Compute the lengths of the surnames
JavaDoubleRDD lenghtsDoubleRDD =
surnamesRDD.mapToDouble(surname  -> (double)surname.length());
```

FlatMapToDouble transformation

FlatMapToDouble transformation

- Goal
  - The flatMapToDouble transformation is used to create a new RDD by applying a function **f** on each element of the "input" RDD
  - The new RDD contains a list of elements obtained by applying **f** on each element **x** of the "input" RDD
  - The function **f** applied on an element **x** of the "input" RDD returns a list of double values **[y]**
    - **[y]**= **f**(**x**)
    - **[y]** can be the empty list

FlatMapToDouble transformation

- The final result is the concatenation of the list of values obtained by applying **f** over all the elements of the "input" RDD
  - i.e., the final RDD contains the merge of the lists obtained by applying **f** over all the elements of the input RDD
- The data type of **y is always double**

FlatMapToDouble transformation

- Method
  - The flatMapToDouble transformation is based on the **JavaRDD<R> flatMapToDouble(FlatMapFunction<T, R>)** method of the **JavaRDD<T>** class
  - An object of a class implementing the FlatMapFunction<T, R> interface is passed to the flatMap method
    - The **public Iterable<Double> call(T element)** method of the DoubleFlatMapFunction<T> interface must be implemented
      - It contains the code that is applied on each element of the "input" RDD and returns a list of Double elements included in the returned RDD
        - For each element of the "input" RDD a list of new elements is returned by the call method
          - The list can be empty

FlatMapToDouble transformation: Example 1

- Create an RDD from a textual file
  - Each line contains a sentence
- Create a new DoubleRDD containing the lengths of the words occurring in the input textual document

FlatMapToDouble transformation: Example 1

```
// Read the content of the input textual file
JavaRDD<String> sentencesRDD = sc.textFile("sentences.txt");

// Create a JavaDoubleRDD with the lengths of words occurring in
// sentencesRDD
JavaDoubleRDD wordLenghtsDoubleRDD =
        sentencesRDD.flatMapToDouble(sentence ->
        {
                String[] words=sentence.split(" ");
                // Compute the length of each word
                ArrayList<Double> lengths=new ArrayList<Double>();
                for (String word: words)  {
                        lengths.add(new Double(word.length()));
                }
                return lengths.iterator();
        });
```

DoubleRDD actions

DoubleRDD actions

- The following actions are applicable only on JavaDoubleRDDs and return a Double value
  - sum(), mean(), stdev(), variance(), max(), min()
- All the examples reported in the following tables are applied on inputRDD that is a DoubleRDD containing the following elements (i.e., values)
  - {1.5, 3.5, 2.0}

DoubleRDD actions: Summary

Action	Purpose	Example	Result
Double sum()	Return the sum over the values of the inputRDD	inputRDD.sum()	7.0
Double mean()	Return the mean value	inputRDD.mean()	2.3333
Double stdev()	Return the standard deviation computed over the values of the inputRDD	inputRDD.stdev()	0.8498
Double variance()	Return the variance computed over the values of the inputRDD	inputRDD. variance()	0.7223
Double max()	Return the maximum value	inputRDD.max()	3.5
Double min()	Return the minimum value	inputRDD.min()	1.5

DoubleRDD actions: example

- Create a DoubleRDD containing the following values
  - {1.5, 3.5, 2.0}
- Print on the standard output the following statistics
  - sum, mean, standard deviation, variance, maximum value, and minimum value

DoubleRDD actions: example

```
// Create a local list of Doubles
List<Double> inputList = Arrays.asList(1.5, 3.5, 2.0);

// Build a DoubleRDD from the local list
JavaDoubleRDD distList = sc.parallelizeDoubles(inputList);

// Compute the statistics and print them on the standard output
System.out.println("sum: "+distList.sum());
System.out.println("mean: "+distList.mean());
System.out.println("stdev: "+distList.stdev());
System.out.println("variance: "+distList.variance());
System.out.println("max: "+distList.max());
System.out.println("min: "+distList.min());
```