Hands On: Building a Graph

In this activity, we are going to build our first graph using Spark's Graphx library. You can consult the datasets to be used within the *big-data-5/graphx/EOADATA* directory. The data has already been placed inside the Docker containers that we are going to use.

Instructions

Step 1. Set up. We will start by setting up *Graphx*. First, make sure to open Docker Desktop to have the Docker engine running. Then, open your local terminal shell, and run the following command to download the Docker image for this module.

```
docker pull pramonettivega/graphx-coursera
```

After pulling the image, build your docker container by running the following command:

```
docker run -it --name graphx-coursera -p 4040:4040 pramonettivega/graphx-coursera
```

After running command, you should be immediately redirectered to the container's shell.

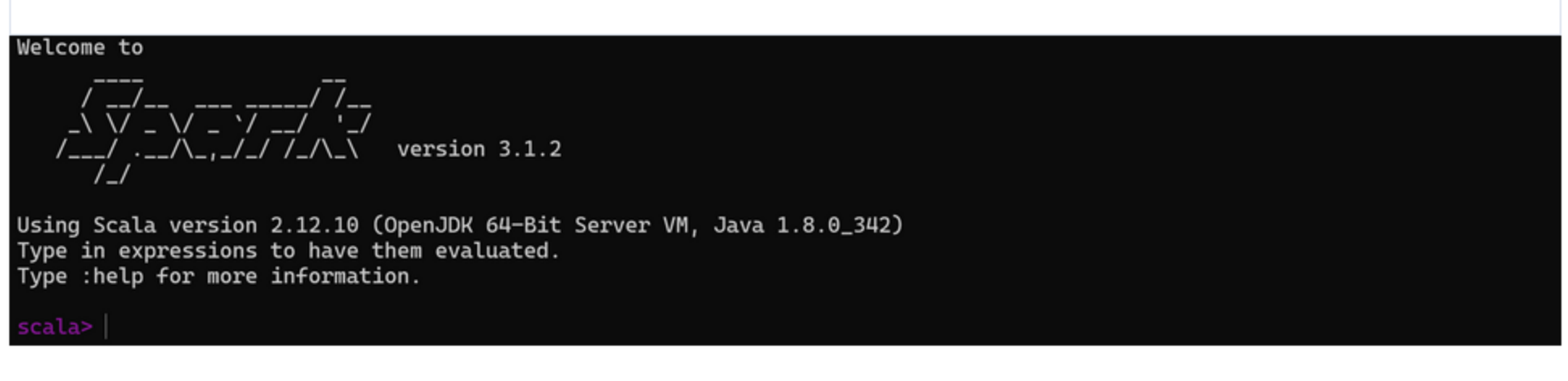
```
root@42f5b4ca0e37:/#
```

You can also confirm that the container is running.

graphx-coursera 42f5b4ca0e37	pramonettivega/graphx-coursera	Running	0%	4040:4040	5 minutes ago

In the container's shell, start the Spark's shell:

```
spark-shell --jars lib/gs-core-1.2.jar,lib/gs-ui-1.2.jar,lib/jcommon1.0.16.jar,lib/jfreecha
```



In the spark shell, set log level to error, suppress info and warn messages.

```
import org.apache.log4j.Logger
import org.apache.log4j.Level
Logger.getLogger("org").setLevel(Level.ERROR)
Logger.getLogger("akka").setLevel(Level.ERROR)

```

Import the Spark's GraphX and RDD libraries along with Scala's source library.

```
import org.apache.spark.graphx._
import org.apache.spark.rdd._
import scala.io.Source
```

Step 2. Import the vertices.

Before importing any datasets, let view what the files contain. Print the first 5 lines of each CSV file.

Input:

```
Source.fromFile("./EOADATA/metro.csv").getLines().take(5).foreach(println)
```

Output:

```
#metro_id,name,population
 1,Tokyo,36923000
 2,Seoul,25620000
 3,Shanghai,24750000
 4,Guangzhou,23900000

```

Input:

```
Source.fromFile("./EOADATA/country.csv").getLines().take(5).foreach(println)
```

Output:

```
#country_id,name
1,Japan
2,South Korea
3,China
4,India
```

Input:

```
Source.fromFile("./EOADATA/metro_country.csv").getLines().take(5).foreach(println)
```

Output:

```
#metro_id,country_id
1,1
2,2
3,3
4,3

```

Now, create case classes for the places (metros and countries).

Input:

```
class PlaceNode(val name: String) extends Serializable
```

Output:

```
defined class PlaceNode
```

Input:

```
case class Metro(override val name: String, population: Int) extends PlaceNode(name)
```

Output:

```
defined class Metro
```

Input:

```
case class Country(override val name: String) extends PlaceNode(name)
```

Output:

```
defined class Country
```

Read the CSV file *metros.csv* into an RDD of Metro vertices, ignore lines that start with # and map the columns to: id, Metro(name, population).

Input:

```
val metros: RDD[(VertexId, PlaceNode)] =
  sc.textFile("./EOADATA/metro.csv").
    filter(! _.startsWith("#")).
    map {line =>
      val row = line split ','
      (0L + row(0).toInt, Metro(row(1), row(2).toInt))
    }
```

Output:

```
metros: org.apache.spark.rdd.RDD[(org.apache.spark.graphx.VertexId, PlaceNode)] = MapPartit
```

Read the CSV file *country.csv* into an RDD of Country vertices, ignore lines that start with # and map the columns to: id, Country(name). Add 100 to the country indexes so they are unique from the metro indexes.

Input:

```
val countries: RDD[(VertexId, PlaceNode)] =
  sc.textFile("./EOADATA/country.csv").
    filter(! _.startsWith("#")).
    map {line =>
      val row = line split ','
      (100L + row(0).toInt, Country(row(1)))
    }
```

Output:

```
countries: org.apache.spark.rdd.RDD[(org.apache.spark.graphx.VertexId, PlaceNode)] = MapPar
```

Step 3. Import the edges.

Read the comma delimited text file metro_country.tsv into an RDD[Edge[Int]] collection. Remember to add 100 to the countries' vertex id.

Input:

```
val mclinks: RDD[Edge[Int]] =
  sc.textFile("./EOADATA/metro_country.csv").
    filter(! _.startsWith("#")).
    map {line =>
      val row = line split ','
      Edge(0L + row(0).toInt, 100L + row(1).toInt, 1)
    }

```

Output:

```
mclinks: org.apache.spark.rdd.RDD[org.apache.spark.graphx.Edge[Int]] = MapPartitionsRDD[11]
```

Step 4. Create a graph.

Concatenate the two sets of nodes into a single RDD.

Input:

```
val nodes = metros ++ countries
```

Output:

```
nodes: org.apache.spark.rdd.RDD[(org.apache.spark.graphx.VertexId, PlaceNode)] = UnionRDD[1
```

Pass the concatenated RDD to the Graph() factory method along with the RDD link

Input:

```
val metrosGraph = Graph(nodes, mclinks)
```

Output:

```
metrosGraph: org.apache.spark.graphx.Graph[PlaceNode,Int] = org.apache.spark.graphx.impl.Gr
```

Print the first 5 vertices and edges.

Input:

```
metrosGraph.vertices.take(5)
```

Output:

```
res5: Array[(org.apache.spark.graphx.VertexId, PlaceNode)] = Array((52,Metro(Ankara,5150072
(56,Metro(Boston,4732161)), (4,Metro(Guangzhou,23900000)), (112,Country(United Kingdom)),
(120,Country(Colombia)))
```

Input:

```
metrosGraph.edges.take(5)
```

Output:

```
res6: Array[org.apache.spark.graphx.Edge[Int]] = Array(Edge(1,101,1), Edge(2,102,1),
Edge(3,103,1), Edge(4,103,1), Edge(5,104,1))
```

Step 5. Using Spark's filter method to return Vertices in the graph.

Filter all of the edges in *metrosGraph* that have a source vertex Id of 1 and create a map of destination vertex Ids.

Input:

```
metrosGraph.edges.filter(_.srcId == 1).map(_.dstId).collect()
```

Output:

```
res7: Array[org.apache.spark.graphx.VertexId] = Array(101)
```

Similarly, filter all of the edges in metrosGraph where the destination vertexId is 103 and create a map of all of the source Ids.

Input:

```
metrosGraph.edges.filter(_.dstId == 103).map(_.srcId).collect()
```

Output:

```
res11: Array[org.apache.spark.graphx.VertexId] = Array(3, 4, 7, 24, 34)
```

We recommend you to go directly to the next activity *Hands-On: Building a Degree Histogram*, because that activity will use the graph you created in this activity.

Mark as completed

Like Dislike Report an issue