Assignment 7: Graph Processing in Spark with Scala

Arash Vahdat

CMPT 733

Fall 2015

Readings You are highly recommended to go through the following readings while doing this assignment:

- **Scala**: Chapter 2, "Advanced Analytics with Spark", Ryza et al., O'Reilly Media, 2015.
- **Spark GraphX**: Chapter 7, "Advanced Analytics with Spark", Ryza et al., O'Reilly Media, 2015.
- **Personalized PageRank**: `http://www.r-bloggers.com/from-random-walks-to-personalized-pagerank/`.

Provided Data In this assignment, we will use a co-authorship network[1]. The data is available at the following location on the SFU RCG network:

`/cs/vml2/avahdat/CMPT733_Data_Sets/Assignment7`

1 Introduction

GraphX is the new Spark API for graphs and graph-parallel computation (e.g. PageRank). At a high level, GraphX extends Spark RDD by introducing a new Graph abstraction: a directed multigraph with properties attached to each vertex and edge. To support graph computation, GraphX exposes a set of fundamental operators (e.g., subgraph, joinVertices, and aggregateMessages) as well as an optimized variant of the Pregel API. In addition, GraphX includes a growing collection of graph algorithms and builders to simplify graph analysis tasks.

This assignment aims at introducing graph processing basics in Scala and GraphX. We will process a relatively small social network to extract different common statistics related to vertices' connectivity. This assignment is based on the Chapter 7 of "Advanced Analytics with Spark" by Ryza et al.. You are encouraged to read this chapter as you are doing this assignment.

The data directory provided with this assignment contains the following items:

- `co_author_graph.txt`: is the co-authorship dataset. It has 36097 lines, in the form "`author_id1 author_id2 year`" which indicates that `author_id1` and `author_id2` had a joint paper in "year".
- `ExtractStat.scala`: is an incomplete script that will be used in Section 2.
- `PR.scala`: used in the lectures for computing PageRank. You can use this script in Section 3.

[1] The network we will use is a variant of the co-authorshop network at https://snap.stanford.edu/data/ca-HepTh.html

2 Setup

Running Spark Shell on Lab Machines If you are working on lab machines, you can easily run Spark shell by typing the following commands in terminal:

```
$ setenv PATH "$PATH":/Users/YOUR_ID/cmpt733/spark-1.5.0-bin-hadoop2.6/bin
$ spark-shell
```

This command will open the Scala version of Spark shell. You can execute your Scala code in the shell easily.

Running Spark Shell on Remotely If you are working remotely, you can easily run Spark shell by typing the following commands in terminal:

```
$ ssh your_id@linux.cs.sfu.ca
$ ssh hadoop
$ spark-shell
```

Writing Scala Code You can use Eclipse as editor for writing Scala code. Follow the instruction in the following page to configure the Eclipse IDE for Scala. Note that Eclipse is already installed on the lab machines.

```
https://nosqlnocry.wordpress.com/2015/03/05/setup-eclipse-to-start-developing-in-spark-scala/
```

Creating Self-contained Applications For this assignment you can simply use the Spark shell to run your scripts. If you are interested in creating self-contained applications that can be run using the spark-submit command check the following page:

```
http://spark.apache.org/docs/latest/quick-start.html#self-contained-applications
```

3 Extract Graph Properties

In this assignment you are provided with an incomplete script named `ExtractStat.scala`. This script starts by reading a graph from a fixed location on the RCG network. If you are working on your personal machine, you will need to change the variable that indicates the location of the input network file. Answer all the following questions in this section by modifying the `ExtractStat.scala` script.

Question 1: (5 marks) The original network file provided with this assignment contains co-authorship links between two authors for each year. In this network, two authors can have multiple common papers published in different years. This creates multiple edges between two nodes in the graph. We would like to know whether two authors have published a paper together or not, regardless of the publication year. Modify the provided code to remove all redundant edges and report the number of vertices and edges using the graph attributes.

Question 2: (5 marks) Use the connectedcomponents() method available in the Graph class, to find connected components in the input network. How many connected components exist in the co-authorship graph? How many nodes belong to the largest connected component?

Question 3: (5 marks) Visualize the degree histogram of your network. Use log scales for both x and y axes in your figure. Does the degree histogram look like a power law probability distribution?

Hint: To compute the degree histogram, you can create two scripts. First, save the count of vertices with a given degree. Then, process the saved count with another scripts (e.g. python) to visualize the histogram.

Question 4: (5 marks) Compute the average clustering coefficient for the given graph.

Question 5: (10 marks) Compute the average shortest distance between every pair of nodes in the graph using the Pregel API. Instead of processing the full co-authorship graph, use the sampled graph extracted in the template code to estimate this measure. Try different sampling rates and report the average shortest distance. In addition to the average shortest distance, report the largest shortest distance in your sampled graph which represents the diameter of the graph.

4 Personalized PageRank

The personalized PageRank algorithm is an extension of the original PageRank algorithm that defines a user-oriented measure for the importance of each vertex in a graph. This algorithm and its extensions have been applied for example for link predictions in social networks.

Question 6: (15 marks) During lectures, we reviewed the Pregel implementation of the PageRank algorithm. Modify this implementation to compute personalized PageRank values for a given set of users. You should assume that you are given a graph and a list of vertex IDs that will be used as the source of your personalized PageRank algorithm. Your algorithm should return personalized PageRank values for all vertices in the graph with respect to each source vertex. Run personalized PageRank on the co-authorship graph for vertex 1 and 2. Report the top 10 nodes with highest personalized PageRanks with respect to each one of these two vertices. Set the damping factor (d, alpha or 1 - resetProb) to 0.9.

Hint: To debug your code, you can create a small graph and compare your results using your implementation in Pregel API with the results obtained by GraphX's PageRank implementation. Note that in Spark's PageRank implementation, you will need to specify only one node as source node using the `srcId` argument:

```
import org.apache.spark.graphx.lib.PageRank
val pr = PageRank.runWithOptions(graph, 100, 0.1, Some(1L))
pr.vertices.collect
```

5 Submission

Your assignment should be submitted online at the following URL: `https://courses.cs.sfu.ca/`. Your submission should include two files:

1. **Report:** Create your report in PDF format including answers to all questions.
2. **Code:** Create an archive file containing all scripts you have used during this assignment. You should create your scripts such that your results can be reproduced. Please include a small readme.txt file in your archive briefly explaining which script corresponds to your answer for each question.