Graph Processing using MapReduce

Chan-Ching Hsu

Purpose

Use Hadoop MapReduce for analyzing large graphs. A graph (sometimes called a network) is a fundamental structure used for modeling relationships between entities, for example, hyperlinks between webpages, or friendship between people in a social network. In this exercise, I processed a large graph that is presented as a set of edge, and computed local properties of graphs, such as neighborhoods and triangles.

Along with this write-up, the same directory includes code for the program.

Experiment

The dataset is the U.S. patent citation data, which is maintained by the National Bureau of Economic Research. In the graph that is considered, the vertex set is the set of all patents issued between 1975 and 1999, for a total of nearly 4 million patents. For each citation, say, from patent A to patent B, there is an edge from vertex representing A the vertex representing B, in the citation graph. Hence this citation graph is a directed graph.

More information about the data can be obtained from its source: http://snap.stanford.edu/data/cit-Patents.html

The graph is in the form of an edge list. Every line of the file has information about a single edge. A line contains information in the format <from vertex> <to vertex>, which means that patent <from vertex> has a citation to paten <to vertex>.

Experiment 1

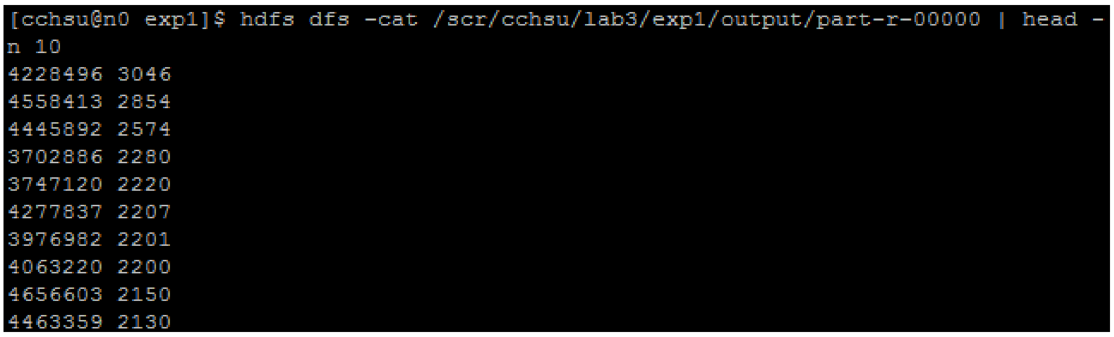
Find significant patents, defined as follows. We say that there is a one-hop citation from patent X to patent Y if X cites Y directly, and we say that there is a two-hop citation from X to Y if there is a patent Z such that X cites a patent Z and Z cites Y. For the purpose of this experiment, we define the **significant of a patent X as the number of distinct patents Y such that there is either a one-hop citation or two-hop citation**

from Y to X. It is possible that a patent X has a direct citation to X itself. There might also be a two-hop citation from X to X. Such self-citations are ignored, either 1-hop or 2-hop.

The task is the following: Write a MapReduce program to extract the ten patents with the largest significance. If there is a tie in choosing the winners, then they can be broken arbitrarily.

To check out what these patents do at http://patft.uspto.gov/netahtml/PTO/srchnum.htm

Snap shot of result



```
[cchsu@n0 exp1]$ hdfs dfs -cat /scr/cchsu/lab3/exp1/output/part-r-00000 | head -
n 10
4228496 3046
4558413 2854
4445892 2574
3702886 2280
3747120 2220
4277837 2207
3976982 2201
4063220 2200
4656603 2150
4463359 2130
```

As shown in the snap shot, the top ten significant nodes are listed in the first column, followed with the numbers of the citations in the second column.

Analysis of the communication complexity

First mapper to reducer

For each key (node), there are two times as many emission as its neighbors (both directions); therefore, the total output that is transferred could be up to 2*N, where N is the total number of edges (lines in the file). The one-hop self-loop situation is dealt with in the mapper so there will be no such output in the transferred output.

Second mapper to reducer

Here the mapper basically reads in the output of the first reducer and emit each pair of key and value for grouping; what the first reducer emit is for each node, emitting all the nodes that can reach it within 1 or 2 hops. Hence, there will be M nodes, with their

1-hop or 2- hop neighbors, where M is the number of nodes in the graph. The 2-hop self-loop situation is deal with in the second mapper. The communication then can be in an order of M; the worst case is each node can reach every others in 2 hops. This will have a complexity of M*M.

THIRD MAPPER TO REDUCER

The mapper emits for each node, the number of distinct nodes that can reach it in 1 or 2 hops; if all nodes can be reached this way, then there will be 2*M entries in transmission.

TOTAL COMMUNICATION COMPLEXITY

With the analysis above, the complexity is in the order of 2N+M*M+2M, resulting in an order of M*M in worst case.

Experiment 2

Consider the same patent graph as the input, but convert it into an undirected graph by ignoring the direction on an edge. Thus each edge in the input file represents an undirected edge between the two vertices.

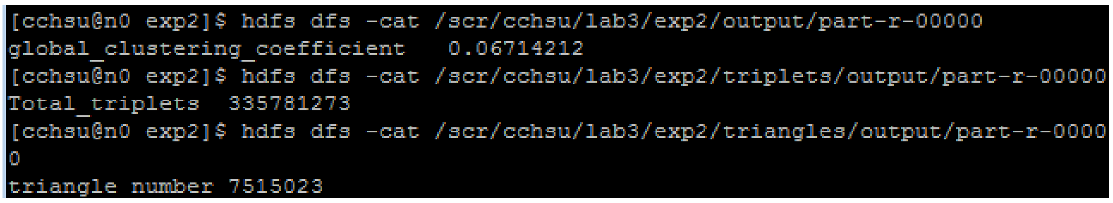
A triangle is a set of three vertices such that all three pairs of vertices are connected to each other. For example a triplet of vertices {4, 7, 9} form a triangle in a graph if and only if the graph has the following edges: {4, 7}, {4, 9}, {7, 9}. The number of triangles in a graph is an important metric of a graph that has applications in several domains including social network analysis. Note that a single vertex can participate is multiple triangles in the graph.

The global clustering coefficient (GCC), which is a measure of "connectedness" in the network, is based on the numbers of different types of triplets of nodes. A triplet of nodes is called an open triplet if the three nodes are connected with two links. A triplet is called "connected" if it is either open or is a triangle. The global clustering coefficient is calculated as

$$GCC = \frac{3 * Number\ of\ triangles}{Number\ of\ connected\ triplets}$$

Write a MapReduce program to compute the global clustering coefficient of the input undirected graph.

Snap shot of result



```
[cchsu@n0 exp2]$ hdfs dfs -cat /scr/cchsu/lab3/exp2/output/part-r-00000
global_clustering_coefficient   0.06714212
[cchsu@n0 exp2]$ hdfs dfs -cat /scr/cchsu/lab3/exp2/triplets/output/part-r-00000
Total_triplets  335781273
[cchsu@n0 exp2]$ hdfs dfs -cat /scr/cchsu/lab3/exp2/triangles/output/part-r-0000
0
triangle number 7515023
```

The numbers of triplets and triangles are 7515023 and 335781273, respectively. This gives us a global clustering coefficient of the graph 0.067 roughly.

Analysis of the communication complexity

FIRST MAPPER TO REDUCER

For each key (node), there are two times as many emissions as the number of its neighbors; therefore, the total output that is transferred could be up to 2*N, where N is the total number of edges (lines in the file).

SECOND MAPPER TO REDUCER

The second mapper basically reads in for each key(node), the number of triplets it can form, so if there are M nodes in the graph, the output will have up to 2*M entries in the value that gets emitted by this mapper.

THIRD MAPPER TO REDUCER

The mapper emits for each key node, the neighbors and the neighbors' neighbors. In the worst case, where each node is a neighbor of each other, there will be M*(M-1)*(M-1) entities emitted, resulting in an order of M*M*M.

FOURTH MAPPER TO REDUCER

Here, for each found triangles, the nodes are read in for the reducer to count, so the communication complexity is related to the number of triangles in the graph. The worst case is that all nodes are connected to each other, meaning it can be the case where for every three nodes out of N, they form a triangle. Then the number is roughly M*M*M when M is large.

Fifth mapper to reducer

Unusually, this mapper just reads in the results that are stored for the number of triplets and triangles and forwards the information to the reducer. Hence, there fundamentally are just two numbers being transferred, leading to a constant complexity.

Total complexity

Based on the analysis for each MapReduce job, the combined complexity is 2N+2M+M*M*M+M*M*M; consequently, the total complexity is in an order of M*M*M.