Pattern Detection in Real-World Social Networks using MapReduce in Hadoop

Auromita Nagchaudhuri, Undergraduate, University of Maryland Baltimore County
Department of Information Systems
Joshua Schultz, Student Mentor, Salisbury University
Department of Math & Computer Science
Dr. Enyue Lu, Faculty Mentor, Salisbury University
Department of Math & Computer Science

Project Description

Pattern analysis on large-scale graphs, such as social networks is a very important part of various applications such as blog analysis, spamming detection, community detection, and many more. This project continues from last summer's endeavors of the REU students. The objective of this study is to test the MapReduce algorithm, in Hadoop framework, called EnumTriangles to see if there can be any improvements regarding the running times, while maintaining 100% accuracy. We are using the real-world datasets from SNAP Stanford with our testing.

MapReduce Programming Model

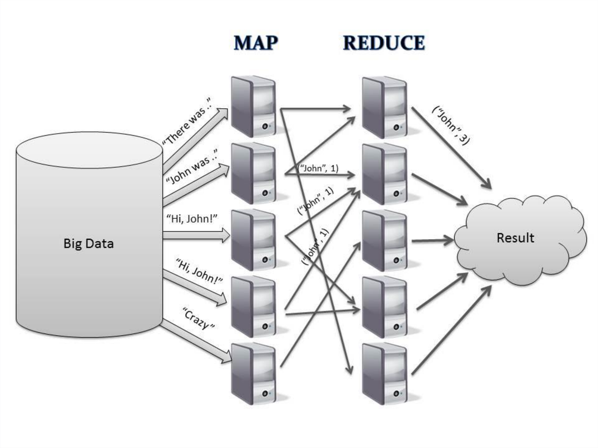

Figure 1

Big Data processing is a complicated procedure. This was made less complex with Google's new programming tool called MapReduce(Fig 1). MapReduce has two procedures:

1. The map phase takes in inputs and divides them into key-value pairs then distributes them to the worker nodes. The worker nodes process the inputs and send it to the master node.
2. In the reduce phase the master node works on collecting all of the data from the given key-value pairs and organizes it into an output file.

Hadoop is an open source library that was used to implement the MapReduce algorithms.

Improving EnumTriangles Algorithm

Enumerating Triangles is a MapReduce algorithm which takes input data in as raw edges. It has 2 Map and 2 Reduce phases. First, it maps out all of the open triads and finally reduces it down to every possible closed triangle. In the output it shows all of the vertices from the given datasets that make up triangles. We used three of the smaller datasets from SNAP Stanford which are Wiki-Vote [size: 1.04MB; 7,115 nodes, 103,689 edges], Slashdot[size: 10.2MB; 77,360 nodes, 905,468 edges] and Epinions[size: 8.99 MB; 131, 828 nodes, 841,372 edges]. The first thing we noticed that although the EnumTriangles algorithm was 100% accuracy, the running time was quite slow.

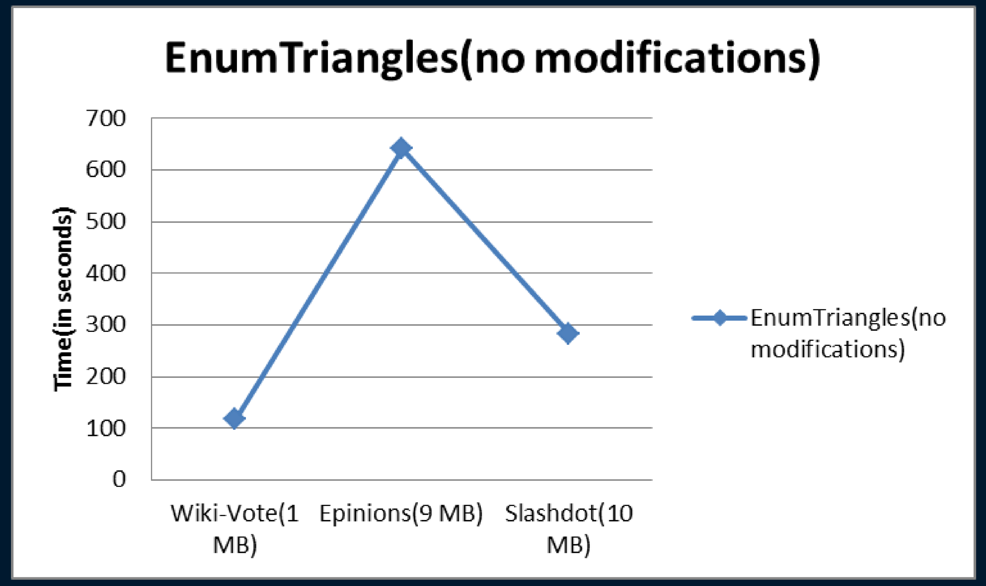

Figure 2

In order to improve the algorithm (see Figure 2 for running time graph), we decided to take out a method called deleteDir(). This method deletes intermediate files that are unnecessary. However, we found that deleting this method only improves running time on a single computer and not on the cloud.

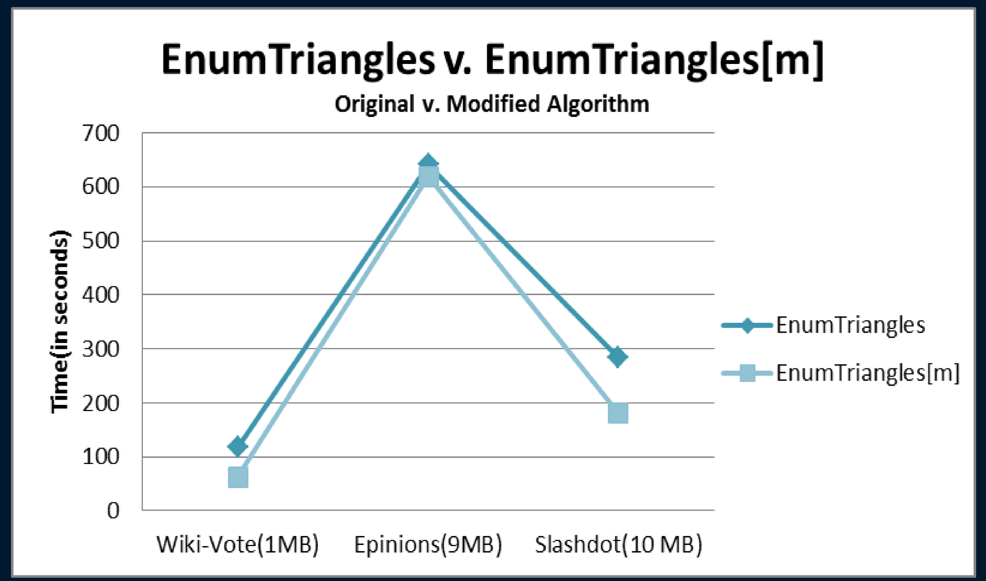

Figure 3

Then, we timed each of the jobs and noticed that the final job (job 3) in the last Reduce() phase of the program took the longest. After that it was much easier to figure out what part of the code we were to focus on. We found the reduce phase of job 3 was the slowest. In the reducer there is a "for loop" that could be parallelized. Using java's ExecutorService, we were able to distribute each iteration of the "for loop" to different cores and improve our running time. Figure 3 shows the slight improvement in running time.

ThreeCompany- Another Implementation for Triangle Counting

In light of enumerating triangles we tested another algorithm on the three datasets with an iterative program called ThreeCompany. It has 1 Map and 1 Reduce phase. The goal of ThreeCompany is to find various groups of threes in a social circle. It takes in inputs as linked lists and outputs them according to whether or not there are closed or open triads. All of the closed triangles have a "3" next to them in the output. After counting the number of "3s" for each we matched the numbers to SNAP Stanford. It was around 99% accurate. All of the other open triangles have a "1" next to them. The small datasets that it took in outputted very large files. Due to it outputting every open and closed triad, ThreeCompany outputs very large files. (See Table 1)

Network	Input Size	Output Size
Wiki-Vote	1.04 MB	270 MB
Slashdot	10.2 MB	1.1 GB
Epinions	8.99 MB	2.9 GB

Table 1

After counting the number of "3s" for each we matched the numbers to SNAP Stanford. It was around 99% accurate. We expected a 100% accuracy , but did not get it. We have not been able to figure out why.

In ThreeCompany algorithm, all of the closed triangles have a "3" next to them in the output. After counting the number of "3s" for each we matched the numbers to SNAP Stanford. It was around 99% accurate. All of the other open triangles have a "1" next to them. Due to the larger output sizes, the running time in comparison to EnumTriangles was slower (see figure 4).

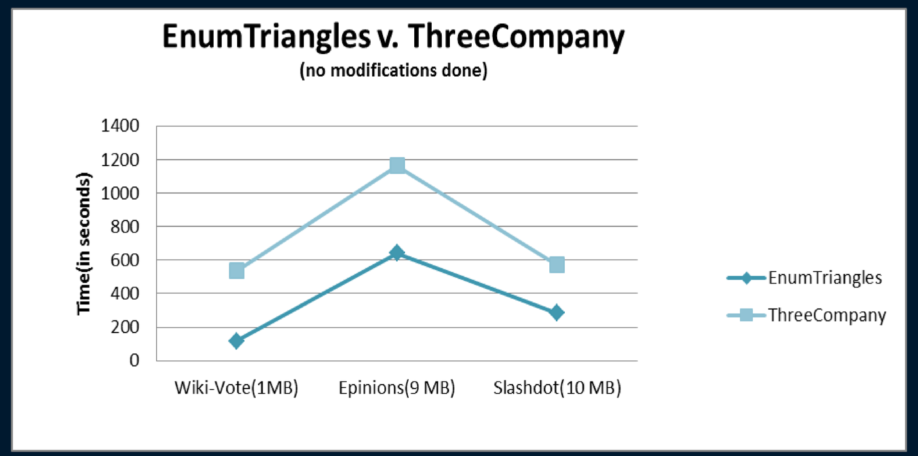

Figure 4

Future Work

- To figure out why ThreeCompany doesn't have a 100% accuracy in determining the number of triangles.
- To reduce the running time for different graph pattern detection algorithms: EnumRectangles, BaryCentricClusters, Trusses, Components, and ThreeCompany.
- To use compression of the mapper output: compression would be done before the data is sent across the network therefore reducing the amount of data being distributed. Doing this will also take longer to pack and unpack so it will be interesting to see whether or not it would be beneficial.
- To use combiners instead of reducers: combiners, although they can perform the same tasks as reducers, however the data is combined on the same computer as the mapper which therefore decreases the amount of data being distributed. This should reduce the running time. However, this only works in specific cases so it may or may not be applicable to all phases.
- To process larger datasets of larger sizes on SNAP Stanford (>1 GB, 1,000,000 nodes, 1,000,000 edges, etc.) such as Facebook, Twitter, Reddit, etc.

This research is funded by National Science Foundation CCF-1156509 under the Research Experience for Undergraduates Program.