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

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.

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.

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 triangles 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: 8.95 MB, 131, 828 nodes, 84,372 edges]. The first thing we noticed that although the EnumTriangles algorithm was 100% accurate, the running time was quite slow.

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).

Table 1

<table>
<thead>
<tr>
<th>Network</th>
<th>Input Size</th>
<th>Output Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki-Vote</td>
<td>1.04 MB</td>
<td>279 MB</td>
</tr>
<tr>
<td>Slashdot</td>
<td>10.2 MB</td>
<td>1.1 GB</td>
</tr>
<tr>
<td>Epinions</td>
<td>8.95 MB</td>
<td>2.9 GB</td>
</tr>
</tbody>
</table>

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.

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, 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-1158559 under the Research Experience for Undergraduates Program.