LAB 5 ACTIVITY REPORT

Student:

Haoyu Guo #50087555 <u>haoyuguo@buffalo.edu</u>

Enze Qian #50091378enzeqian@buffalo.edu

Introduction:

According to the requirement of activity, we implemented the map reduce process of Latin word co-occurrence by Spark in the case of 2-gram and 3-gram. In the attached file, there are two sample outputs included in the file folders named as "2-gram_output" and "3-gram output".

For example, a output pairs looks like the fallowing:

"('contra,nescius,prophetauit', '<ambrose apdavidaltera 3>')('errauit,nouit,soluere', '<ambrose apdavidaltera 3><ambrose apdavidaltera 3>')"

In the Unix-like terminal, it may format as:

('contra,nescius,prophetauit', '<ambrose apdavidaltera 3>')

('errauit,nouit,soluere', '<ambrose apdavidaltera 3><ambrose apdavidaltera 3><ambrose apdavidaltera 3>')

In this sample, the words, contra, nescius, prophetauit, are

Analyze

This graph only record the runtime of map reduce process in Spark. As the graph shown, the increase of files as input brought essential effect on the runtime performance. And the different of 3-gram and 2-gram is the fatal factor of performance. For example, in a sentence with 5 words, there are actually 5*4 2-gram pairs need to calculate. However, this number for 3-gram is 5*4*3. This increasing can be nightmare when the amount of words gets large.

Other interesting thing for Spark is the performance of RDD operation such as take(), saveAsTextfile(), collect() will affected by the change of gram tempestuously. Different with the Hadoop MR, the map reduce function takes consistent time to finish if we don't generate output file. The same output for one file in 2-gram will take 1 min to generate, but by 3-gram the time extends to several hours. In my thought, the reason of the great difference is because the data get skewed heavily and Spark only load, record the operation and run function when we actually need output.

Light blue line : 3-gram

Dark blue line: 2-gram

