SCHOOL OF ELECTRICAL AND INFORMATION ENGINEERING

University of the Witwatersrand
School of Electrical and Information Engineering

ELEN4020: Data Intensive Computing

Laboratory Exercise 3

Authors:

Kayla-Jade Butkow
714227

Jared Ping
704447

Lara Timm
704157

Matthew van Rooyen
706692

Date Handed In: 5th April, 2018

1. Matrix Multiplication

The input to map function consists of two separate text files containing a matrix formatted as a list of strings. The first row of the file provides the dimensions of the given matrix. The proceeding rows contain three columns, the first two representing the row and column index of the element and the third the value of the element.

Each matrix is read in of the form M_{ij} and N_{jk} where i represents the row elements and j the column elements. The product matrix is therefore represented as $P_{i,j} = M_{ij} * N_{jk}$. This is achieved by taking the first row of M and multiplying it by the first column in N and summing the values. This function is performed for every column in B before repeating the process for every row in A.

2. Map Reduce

In order to perform matrix multiplication for large matrices, a MapReduce algorithm was implemented. The algorithm consists of two key features, namely the `map()` function and `reduce()` function.

The input matrix to the map function is of the form (row, column, value). This data is then assigned a key (i, j, k) within the may function in order to produce a key-value format at the output. The function is performed on each matrix separately before combining the values of each matrix according to the key value producing key-value pairs.

The algorithm consists of two reduce functions. The reduce function performs calculations on the key-value output pairs produced by the map function. `reducer_multiply` simply multiplies the values found at each key (i, k) and prepares the data for summation. The reduce function only processes the data one key at a time. `reducer_sum` sums the output values from the multiplication at each key performing the final step in the matrix multiplication process.

3. MrJob

MrJob is a python-based MapReduce framework. The steps function represent the jobs handled within the script. This also allows for multiple one-step jobs to be executed within the program. The mapper function performs allows for the map function to be presented as a single job. The input data is read in one line at a time after which the data is broken down into single values and given a key. Each yeild function within the mapper is outputted as a single line of code. The reducer performs the multiplication and sums the output results before emitting them. MrJob allows for the inclusion of two separate files as input streams to be included in the program.

MrJob is preferable in it's application as the code has no dependencies with Hadoop in comparison to the other frameworks. the framework was also chosen due to documentation being more readily available and easier to implement.

4. Multiplication

4.1 Algorithm A

Algorithm one consists of two iterations of the Mapreduce procedure resulting in two mapper and reducer functions. The first iteration of the mapper function assigns the key value pair of j which is common in both matrices M_{ij} and N_{jk}. The reducer function outputs the key-value pair associated with j produced by either matrix. This is used as the input to second mapper function which creates a set of key values (i,k) containing the values associated with the j pairs. The values associated with the same (i,k) key are the summed and emitted by the reducer function in order to get the final result.

Main difference, algorithm 1 uses two steps. Two mappers, two reducers. Simplifies process of mapping to a key. Values associated with matrices, value 0 equals matrix 1. Ensures correct matrice is being multiplied. Split values in reducer, yields new key and new value.

4.2 Algorithm B

Single map and reducer stage. Key has an index. Sorts them by j.Multiply corresponding entries with the same j value.

5. Pairs of Nodes Connected by Paths of Length 3

5.1 Overview

According to [1], the total number of paths of length 3 from j to i in G is:

$$N_{ij}^3 = \sum_{k,l=1}^{n} A_{ik}A_{kl}A_{ij} = [G^3]_{ij} \tag{1}$$

Where: $G = (v, e)$ is an unweighted directed graph of $\mid v \mid$ nodes and $\mid e \mid$ edges

From Equation 1, it is evident that the most efficient way to represent G is in matrix form. Since the majority of the values in the G matrix are 0, all zero value entires should be emitted from the input file.

From Equation 1, in order to calculate the pairs of nodes that are connected by paths of length 3, the input matrix *File2ForLab3.txt* must be cubed.

In order to perform the calculation, multiplication algorithm A was used. First, the input text file was multiplied with itself to result in A^2. Then, the resulting matrix was multiplied with the input text file to result in A^3. In order to count the number of paths, the resulting matrix entries were summed, and this value indicates the total number of paths of length 3.

5.2 Implementation

In order to implement the above mentioned procedure, two multiplication classes were required. The first (*lab3-multiplicationPart1.py*) makes use of Algorithm A as discussed in *Section 4.1.* The result of the first multiplication is written to the text file *partialOutputQ6.txt.* The second class (*lab3-multiplicationPart2.py*) makes use of Algorithm A, but takes two in input files. This is necessary for the second multiplication in which the resultant matrix is multiplied by the input matrix. In this class, the text files are differentiated by name using the MRJob `os.environ['mapreduce_map_input_file']` function which gives the program access to the input file name. By using the file name, all data that originates from the file *File2ForLab3.txt* is assigned a 0 as an identifier, and all data from the other file is assigned a 1.

To calculate the total number of paths of length 3, a third mapper and reducer stage was added to the second class. In the mapper stage, all the entries of the resulting matrix were assigned the key *Number of paths:* . This ensures that all the values have the same key and can thus be summed in the reducer. Within the reducer stage, the values are summed and the final count is written to the output file.

In order to run the two classes, a bash script was created. Within this script, the two classes are called with their respective input files. The final resulting matrix is written to the text file *outputQ6.txt.*

REFERENCES

[1] M. Newman. *Networks: An Introduction*, p. 136. Oxford, 2010.