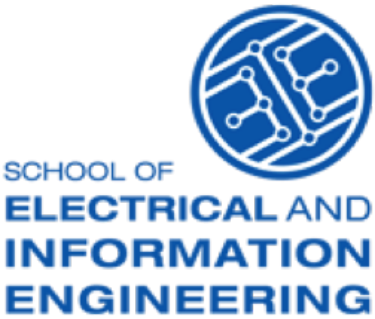

University of the Witwatersrand
School of Electrical and Information Engineering

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 the map function is a text file containing a input matrix written in Matrix-Market format. The first row of the file provides the dimensions of the matrix. The following rows contain three columns, with the first two representing the row and column index of the element and the third the value of the element.

The product matrix is represented as $P_{ik} = M_{ij} * N_{jk}$, where M_{ij} and N_{jk} are the input matrices. In order to perform the multiplication, the first row of M is multiplied by the first column in N and the resultant values are summed. This process is performed for every combination of rows in M and columns in N.

2. MapReduce & MrJob

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

The `map` function takes as an input a key and a value and emits key-value pairs. The `reduce` function then makes use of the output from the `map` function. It takes as inputs a key and a list of values associated with the key and performs an action of the values. It also emits key-value pairs.

MrJob is a MapReduce framework which provides an environment which allows multi-step MapReduce jobs to be written in Python and run on a variety platforms [1]. A job is defined by class which inherits from `MRJob`, and which contains the functions which define the `steps` of the job. Each `step` consists of a `mapper`, a `combiner` and a `reducer`, of which you must use at least one [1].

The input data is read in one line at a time. The `mapper` function performs the role of map in MapReduce allocating key-value pairs to each line of text input. Similarly, the `reducer()` performs the reduce task in order to output a single value for each key-value pair. Each yield function found at the end of the mapper and reducer is responsible for a single line of output code.

MrJob is intended as an easy way to write Python programs that run on Hadoop. Using MrJob, one is able to test the code locally, without installing Hadoop, before you deploy the code on your cluster. For the purpose of this laboratory, this local approach was used. Additionally, MrJob was preferred over other Python MapReduce libraries owing to its relative ease of implementation and availability of documentation.

3. Multiplication

Two algorithms for performing matrix multiplication with MapReduce were implemented. Since the matrix is to be multiplied with itself, only one matrix is taken as an input to the job.

In both the algorithms, the `map` function transforms the input file into a format which can be used by the reducer to perform the necessary calculations. The input data to the map function is a line from the text file in the form *row column value*. The data is then assigned a key so that a key-value pair can be yielded from the function.

The `reduce` function performs computations on the key-value pairs produced by the `map` function in order to produce the product matrix.

3.1 Algorithm A

In Algorithm A, two MapReduce steps are used, meaning that a mapper and reducer stage are used, followed by a second mapper and reducer stage.

In the first iteration, the algorithm takes in one matrix

in two matrices as input in the form $M(i, j, v)$ and $N(j, k, w)$. In the first iteration, the map performs a join between the two matrices, M_{ij} and N_{jk}. Each matrix elements value within is assigned a key according to it's j value which is common to both matrices. For the first matrix, the pair in the form (M, i, m_{ij}) is created with m_{ij} representing the value of each element in the M matrix. The same is done for the second matrix, producing the j key-pair in the form (N, k, n_{jk}). The M and N values within each pair are assigned 0 and 1 respectively in order to represent which matrix the pair belongs to.

The reduce function in the first iteration combines each element value in M with its corresponding value in N and emits the key value pair $(i, k, m_{ij}n_{jk})$ for the corresponding j key-value pair. This is simply a grouping however meaning an extra step is necessary to find the product of the two values. The output of the reduce function is then used as the input to the map function in the second iteration.

In the next iteration, a key-value pair of the form (i,k) is created for each element and assigned the combined value $m_{ij}n_{jk}$. Each i value will represent the row index of the product matrix and k representing the column index.

The reducer emits each pair and splits the values. The yield then produces a new key and sums the given values assigned with each pair. The new key represents the value of at each row, column index within the product matrix.

3.2 Algorithm B

Algorithm B makes use of a single map and reduce stage. Instead of originally producing key-value pairs according to each j element, the map function creates (i,k) key-value pairs for each matrix containing the common j element. For M, the pair (M, j, m_{ij}) is created for each value of k according to the number of columns contained within the N matrix. The same is done for matrix N, (N, j, n_{jk}), with a pair created for a unique i value according to the number of rows within the M matrix. Once again, The M and N values of each pair are represented as 0 and 1's respectively in order to identify which key belongs to each matrix.

The yeild of the map function is then used as the input to the reducer. The key (i,k) represents the row and column index's of the product matrix. The function sorts the values in matrix M according to j in the key-value pair list. This is then repeated for matrix N. Values m_{ij} and n_{jk} at the j_{th} element of each list are the multiplied together before being summed together to produce the final value at the row, column index.

this product is represented mathematically by the following equation:

$$P_{ik} = \sum_{j=1} m_{ij} * n_{jk} \tag{1}$$

4. Pairs of Nodes Connected by Paths of Length 3

4.1 Overview

According to [2], the total number of paths of length 3 from j to i in G is:

$$N_{ij}^3 = \sum_{k,l=1}^{n} A_{ik}A_{kl}A_{ij} = [G^3]_{ij} \tag{2}$$

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

From Equation 2, 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 2, 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.

4.2 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 3.1*. 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*.

5. Results

From the tables below, it is evident that Algorithm B was found to perform best. This result was obtained by performing a timed benchmark on the multiplication of two 3 x 3 matrices.

Algorithm	**Time taken (s)**
A	0.4670000076
B	0.4359998703

REFERENCES

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