WITS UNIVERSITY

School of Computer Science and Applied Mathematics
UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

COMS4040A & COMS7045A Assignment 2 – Report

Abdulkadir Dere - 752817 - Computer Science Hons

14 May 2020

1 Introduction

2 Methodology

2.1 Matrix Transpose

2.2 Vector Addition

2.3 Tiled Matrix Multiplication

3 Experiment

3.1 Experiment Setup

3.2 Experiment Data

Experiments are done using synthetic data. Synthetic data is generated by using a random number generator algorithm for the reference and query matrices. Following sizes have been used for the variables m, n and d in different experiments as shown in results. We have used 100000 for m references and 10 for d dimensions. We have altered q query points between 100 and 200.

3.3 Experiment Results

Experimental results are shown with different distance and sorting algorithms and with varying n query points.

3.4 Summary of Results

Experiment results indicate that parallelising the sequential distance and sorting algorithms improve the computation time. The combination of using manhattan distance and merge sort results in the most efficient computation (table:**??**) followed by euclidean distance and merge sort (table:**??**) combination. This indicates that on large sets of data, merge

sort performs better than quick sort. Comparison between using task construct (table:**??**) and section construct (table:**??**) shows that task construct performs significantly better than the sections construct in the parallel program. Results of parallelising selection sort (table:**??**) shows that any sorting algorithm which does not adopt the divide and conquer approach performs poorly compared to the quick and merge sort algorithms, which do use divide and conquer approach. Euclidean distance calculation performs better when compared to manhattan distance calculation as can be seen in tables **??** and **??** .

4 Conclusion

We have implemented *k*NN algorithm in both sequential and parallel methods. We have focused on parallelising the distance and sorting algorithms for the *k*NN algorithm. We have conducted experiments with different distance and sorting functions and recorded the results of the experiments. As hypothesised, parallelising the distance and sorting algorithms improved the computation time of *k*NN algorithm significantly. Divide and conquer type of sorting algorithms perform computationally better when parallelised compared to other sorting algorithms.

In order to get the best results it is important to use divide and conquer approach algorithms, because of their exceptionally good computational performance when computed in parallel. Using the correct sorting algorithm is key to the computation performance since sorting occupies most of the computation time as can be seen in the experiment result percentages. Implemented methods are sufficient enough to assess the importance of parallelising distance and sorting algorithms.