High Performance Computing
Assignment 1 Report

Simbarashe Mhlanga-1325952

March 2020

UNIVERSITY OF THE WITWATERSRAND

SCIENTIA ET LABORE

JOHANNESBURG

1 Introduction

Searching is one of the fundamental problems in computer science field and is the field we computers excel.With the advancements in the information retrieval systems, the focus of searching has been changed to similarity search.Similarity search, also called proximity search, is searching for data points that are close to a given query point based on a distance measure [Berkay Aydin,2014].For high dimensional data, because dimensions are very high (as name suggests), volume of the space for data points increases tremendously, as they become more sparse with more dimensions [Berchtold et al.1997].Further more there is no ordering in the dimensions where we could search using classical methods.Under these conditions, similarity based search became more popular, since equality-based or range queries do not suffice the needs of different fields such as data mining, computational geometry, bioinformatics etc. In this report the presented algorithm is the K-nearest neighbour (kNN).Which can be described as:*Given a d-dimensional space in R^d,query points of $q \in R^d$ and a set of data points P.The algorithm returns the data point or points in P that is closest to our query point with respect to a defined distance D.* Knn nearest neighbour algorithm is used in machine learning applications for clustering that is when there is need to get the sense of data the algorithm can easily recognise patterns and cluster given data with similar features as one group.Information retrieval systems, with the focus on high dimensional data is the main application area for nearest neighbor search. A wide range of search engines either for text, image or audio retrieval uses nearest neighbor queries for retrieving the data as well as ranking them.

2 Assignment question 1

1.design a parallel algorithm for kNN based on the BF approach using Foster's parallel algorithm design methodology.

Answer:
Foster's parallel algorithm design methodology Summary
Its a four stage design process whose input is the problem statement,the four stages are described as Partitioning the first stage of design methodology that is the process of dividing the computation and the data into pieces.Communication this process of determining how tasks will communicate with each other distinguishing between local communication and global communication.Agglomeration The process of grouping tasks into larger tasks to improve performance.Mapping The processing of assigning tasks to physical processors.

Implementation to K-nearest neighbour (kNN)

Algorithm

The method is labour intensive when given training set.K-nearest neighbour (kNN) classifiers are based on learning by analogy that is by comparing a given test tuple with training tuples that are similar to it.The training tuples are described by n attributes ,where each tuple represents a point in the n-dimensional space.When given unknown tuple a KNN classifier searches the pattern spaces for the k-training tuple.These training tuples are the K nearest neighbours of the unknown tuples.The closeness is defined by distance metrics such as Euclidean distance,Manhattan distances ,Mahalonobis distance and so on.

Foster's parallel algorithm design methodology implementation into the K-nearest neighbour

First Stage :Partitioning

In our algorithm, the input data partitioning is the natural decomposition technique because the output (the computed distances) is not clearly known a-priori. It divides the data set equally according to the number of worker processes by sending a one data partition for each of them.

Second Stage:Mapping

Once a problem has been decomposed into concurrent tasks, these must be mapped to processes (that can be executed on a parallel platform).In our algorithm, we use the static mapping technique that distributes the tasks among processes prior to the execution of the algorithm.The scheme for this static mapping is mapped based on data partitioning because our data represented in a two-dimensional array. So, the most suitable scheme used for distributing the two-dimensional array among processes is the row-wise 1-D block array distribution that distributes the array and assign uniform contiguous portions of the array to different processes.According to the previous selected decomposition and mapping techniques, the suitable parallel algorithm model is the master-slave model in which the master process generates the work and assigns it to worker processes.

Third Stage: Communication

In most parallel algorithms, processes need to exchange data with other processes.In our algorithm the master process divide the data set according to the number of workers and sending a one data partition for each of them with the k and query-instance values. Also, the worker processes send the k-th ordered list to the master which include the k-th distances and classes.The suitable communication operation for our algorithm is the scatter operation (one-to-all personalized communication), in which the master process sends a unique part of the divided data to each of the worker processes.The dual of one-to-all personalized communication or the scatter operation is the gather operation, or concatenation, in which the master process collects a unique k-th ordered list from each other worker processes.

3 Assignment question 2

This section presents the algorithm implementation methodology,experiment setup and the summary of the work presented in this report.

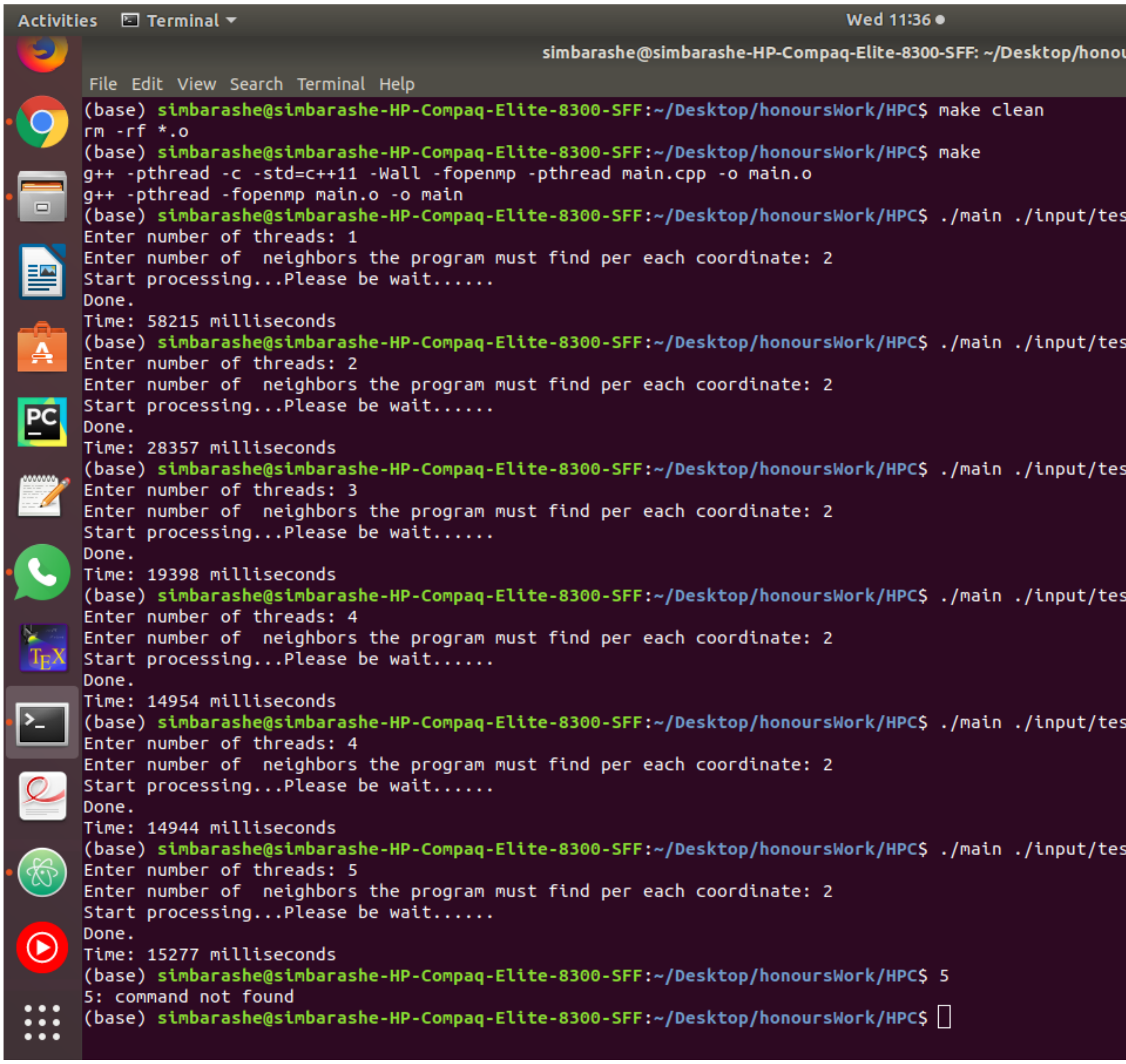
3.1 Methodology

It is a c++ parallel implementation of k-nearest neighbour algorithm using OpenMP and c++ threads.As discussed during design in question 1.The input data partitioning is the natural decomposition technique because the output(the computed distances) is not clearly known a-priori.During the design of this algorithm the main decision was how is the data going to be distributed to different threads that is how are we going to fully exploit the threads supported by any machine.The data will be evenly split between n threads that is each thread computes the k-th neighbour for its part of data.Since the task is to compare how serial computation compares to parallel computation.Serial computation is when the program runs or execute on a single thread on a given computer while parallel computation is when the program run or execute on more than one thread on a given computer a the same time.In this program the advantages provided by threads are fully exploited.When the user run this program the program asks the user to input the number of threads the program should run on and the advantages of parallel computing a realised instantly by experimenting with different number of threads.

3.2 Experiment

Data description
The input data to this program is the data that contains 9271 3D world space coordinates.
Machine Specification
This program was coded and tested on HP desktop with the following features:
Memory:3.7 GiB
Processor:Intel® Core^{TM} i5-3470 CPU @ 3.20GHz 4
Graphics:Intel® Ivybridge Desktop
OS type:64-bit

The details of the test
This program has the ability to scale up its performance depending on the machine it execute on.The user is able to input the number of threads the program must run on and the data to this is spread over the number of threads.It must be noted that on this machine the program was performing best using 4 threads an increase to 5 threads showed a slightly decrease in performance of the program when using time to complete the task as the base of measurement.The bellow picture is a screenshot of performance of the program when running on 1,2,3,4,5 threads respectively.More work is done when the program has to find a larger number of neighbors per each coordinate.


Activities
Terminal
Wed 11:36
simbarashe@simbarashe-HP-Compaq-Elite-8300-SFF: ~/Desktop/honou
File Edit View Search Terminal Help

```
(base) simbarashe@simbarashe-HP-Compaq-Elite-8300-SFF:~/Desktop/honoursWork/HPC$ make clean
rm -rf *.o
(base) simbarashe@simbarashe-HP-Compaq-Elite-8300-SFF:~/Desktop/honoursWork/HPC$ make
g++ -pthread -c -std=c++11 -Wall -fopenmp -pthread main.cpp -o main.o
g++ -pthread -fopenmp main.o -o main
(base) simbarashe@simbarashe-HP-Compaq-Elite-8300-SFF:~/Desktop/honoursWork/HPC$ ./main ./input/tes
Enter number of threads: 1
Enter number of  neighbors the program must find per each coordinate: 2
Start processing...Please be wait......
Done.
Time: 58215 milliseconds
(base) simbarashe@simbarashe-HP-Compaq-Elite-8300-SFF:~/Desktop/honoursWork/HPC$ ./main ./input/tes
Enter number of threads: 2
Enter number of  neighbors the program must find per each coordinate: 2
Start processing...Please be wait......
Done.
Time: 28357 milliseconds
(base) simbarashe@simbarashe-HP-Compaq-Elite-8300-SFF:~/Desktop/honoursWork/HPC$ ./main ./input/tes
Enter number of threads: 3
Enter number of  neighbors the program must find per each coordinate: 2
Start processing...Please be wait......
Done.
Time: 19398 milliseconds
(base) simbarashe@simbarashe-HP-Compaq-Elite-8300-SFF:~/Desktop/honoursWork/HPC$ ./main ./input/tes
Enter number of threads: 4
Enter number of  neighbors the program must find per each coordinate: 2
Start processing...Please be wait......
Done.
Time: 14954 milliseconds
(base) simbarashe@simbarashe-HP-Compaq-Elite-8300-SFF:~/Desktop/honoursWork/HPC$ ./main ./input/tes
Enter number of threads: 4
Enter number of  neighbors the program must find per each coordinate: 2
Start processing...Please be wait......
Done.
Time: 14944 milliseconds
(base) simbarashe@simbarashe-HP-Compaq-Elite-8300-SFF:~/Desktop/honoursWork/HPC$ ./main ./input/tes
Enter number of threads: 5
Enter number of  neighbors the program must find per each coordinate: 2
Start processing...Please be wait......
Done.
Time: 15277 milliseconds
(base) simbarashe@simbarashe-HP-Compaq-Elite-8300-SFF:~/Desktop/honoursWork/HPC$ 5
5: command not found
(base) simbarashe@simbarashe-HP-Compaq-Elite-8300-SFF:~/Desktop/honoursWork/HPC$
```

4 Summary

Indeed there are gains realised when the program is run on multiple threads.These gains have a certain maxima which are brought about by the communication between threads after completing the tasks.

Problems faced during this Assignment Due to poor processing power of my computer the running of the program was very slow for test purposes.

References

[1] B. Aydin, "Parallel algorithms on nearest neighbor search," *CSC8530-Survey Paper for Parallel Algorithms-Georgia State University*, p. 2, April 2014.

[2] A. S. J. A. Hammad, "Implementation of a parallel k-nearest neighbor algorithm using mpi," *Central European Researchers Journal*, 2019.