WITS UNIVERSITY

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

COMS4040A & COMS7045A Assignment 1

Hand-out date: 10:00 am, March 6, 2020
Due date for Hons students: 24:00 pm, March 17, 2020
Due date for MSc students: 24:00 pm, March 19, 2020

Objectives

1. Design parallel algorithms using Foster's parallel algorithm design approach.
2. Implement parallel algorithms using OpenMP with a focus on achieving scalable performance.
3. Evaluate the performance of a parallel algorithm implementation.
4. Writing a short scientific report.

Instructions

1. This is an individual assignment.
2. The due date is strictly applied. Late submissions will be penalized by 10% to 50% of the total marks.
3. Hand in the electronic files or source codes on Sakai.
4. Hons students are requested to solve Problem 1, and MSc students to solve both Problems 1 and 2.
5. The total marks for this assignment is 48 for Hons students, and 58 for MSc students. The weight of this assignment towards the final mark is 12%.

Problem 1

K-nearest neighbour (kNN) is a widely used algorithm for the tasks of classification and regression in pattern recognition, data mining, machine learning, and so on.

Let $\mathscr{P} = \{p_0, p_1, \ldots, p_{m-1}\}$ be a set of m reference points with values in $\mathbb{R}^d$, and $\mathscr{Q} = \{q_0, q_1, \ldots, q_{n-1}\}$ be a set of n query points in the same vector space. The kNN search problem refers to finding the k nearest neighbours of each query point $q_i \in \mathscr{Q}$ in the reference set $\mathscr{P}$ given a distance metric. Commonly used distance metrics include Euclidean or Manhattan distances. Other distance metrics, such as cosine similarity and Mahalanobis distance , can also be applied.

For example, when the data points p_i or q_i represent complex objects, such as an image, the dimension d could be very high, as each d-dimensional vector represents a feature extracted from the object. Regarding the distance metric, Euclidean and Manhattan distances are simply the ℓ_2- and ℓ_1-norms, respectively. Efficient algorithms are proposed for kNN search problem, including the one using kd-trees to partition the search space which leads to $\mathrm{O}(\log \mathrm{m})$ query time. However, our focus here is a brute force algorithm.

The brute force (BF) algorithm to find kNN is as follows. For each query point q_i, $0 \leq i < n-1$,

1. compute all the distances between q_i and $p_j, 0 \leq j \leq m$;
2. sort the computed distances;
3. select the k reference points corresponding to the k smallest distances;

4. repeat steps 1 to 3 for q_{i+1}.

The complexity of this algorithm is O(nmd) for distance computation and O(m^2) for sorting (using quicksort). Clearly, the computational complexity is high, however, there is plenty of parallelism we could exploit in this algorithm.

In this assignment, you are to

1. **design a parallel algorithm for kNN based on the BF approach using Foster's parallel algorithm design methodology;** [8/48]
2. **implement the BF algorithm for kNN search problem using OpenMP.**

The implementation should be in the following manner.

1. Implement a serial version of the BF algorithm, which consist of two compute intensive components - distance calculations and sorting;
   - Choose a specific distance metric. Here, I encourage you to implement different types of metrics, such as Euclidean and Manhattan distances.
   - At least three sorting algorithms, including *quicksort*, should be implemented. The other two is your own choice. It could be any two of the insertion sort, bitonic sort, merge sort, and others.
2. Parallelize the two components using OpenMP.
   - For the sorting component, apply both `sections/section` construct and `task` construct, respectively, where applicable.
   - Benchmark the performance of your implementation. Once your implementation is complete, run a series of tests that evaluate the performance of your implementation.
     - Test data. You may use synthetic data, in which case you need to generate sets of reference data and query data, respectively. Depending on the machine you run your test on, choose a sufficiently large number for the reference data set $\mathscr{P}$.
     - Tests should run on varying m,n,d. In this case, you can choose two large numbers for m, $d = 32,64,128,256,512$, and $n = 200,400,800,1600$ etc. (Avoid choosing data size that is too large for your machine, which could result in long running time, or too small data size which is meaningless in this context.)
     - Evaluate your solutions
       * by comparing the performances on serial and parallel implementations; [20/48]
       * on using various algorithms for the sorting component; [6/48]
       * on using varying task parallelism approaches (i.e., `sections\section` and `task` constructs); [6/48]
       * and by testing your parallel solution for varying parameters of m,n,d. [8/48]

       The performance should be recorded in a clear manner using tables or plots in your report. By performance, we consider only running time. The overall running time should be decomposed into distance calculation time and sorting time. The two running times (as percentage of the overall time), as well as the overall time are then recorded as functions of different parameters including m,n,d. (You don't need to include the file reading time in your overall time if you read your data from a file.) An example of table reflecting some imaginary results is illustrated in Table 1.

n	200	400	800	1600
Distance	30%	40%	50%	60%
Sorting	60%	50%	40%	30%
Overall	0.08s	0.16s	0.32s	0.64s

Table 1: The overall time and the decomposition percentages as a function of n, where $m = 100000, d = 8$.

- Report. This document will be the main source of evaluation of the quality of your work. Thus, produce a decent report, which should include the following.

- A general introduction to the problem, algorithm, and its various applications. Cite the sources properly when applicable.
- The solution to the problem, i.e., methodology.
- Experiment setup – the description of data being used; the specifications of machine you run your tests on; the details of the tests you run; and most importantly, the results. **Note:** Do not just put your results in a table or a graph without giving any explanations or discussions on those in the text.
- A summary of your work, plus some discussions on the challenges you encountered in the current solutions and other potential problems.

Problem 2

(Only MSc_CWRR students are requested to solve this problem.)

1. Implement OpenMP parallel algorithms for the following two standard matrix operations:

   - Kronecker product of matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$, which is

   $$\mathbf{A} \otimes \mathbf{B} \equiv \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{IK \times JL} \tag{1}$$

   - Khatri-Rao product of matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$, which is

   $$\mathbf{A} \odot \mathbf{B} \equiv \begin{bmatrix} \mathbf{a_1} \otimes \mathbf{b_1} & \mathbf{a_2} \otimes \mathbf{b_2} & \ldots & \mathbf{a_K} \otimes \mathbf{b_K} \end{bmatrix} \in \mathbb{R}^{IJ \times K}. \tag{2}$$

   Benchmark your implementation using increasing values of matrix sizes, and compare the performance of both serial and parallel implementations. **[10/58]**

Hand-Ins

A compressed file (.tar.gz) named by your student number is to be submitted on Sakai. This file when extracted should include: a 2 to 4 pages of report in PDF format (name your report using your student number as well) and your source code folder.

1. A `Makefile` must be present in your source code folder for the compilation.
2. It is recommended that the report should be written using latex. In particular, use 10pt, single line spacing for your report. Template tex files are provided in this regard. You just need to edit `assignment1_report.tex` file and compile `template.tex` using latex.