COMS4040A & COMS7045A Course Project

Hand-out date: April 16, 2018
Due date: 24:00 pm, May 21, 2018

General Guidelines

1. You are expected to work in groups of 1 - 2 persons for this project. Some of the projects are suitable for those who are keen on working alone.
2. Each group is requested to work on one of the projects using programming models
   (a) CUDA and MPI - for Honours students,
   (b) OpenMP, CUDA and MPI - for Masters students.
3. In your report, proper citations and references must be given where necessary.
4. Note that the following descriptions of the problems are only outlines. Further elaborations are necessary.
5. Formulating your own project is allowed. It can be a particular problem given in other courses, or a problem close to your current Honours or Masters research, where you would like to bring in high performance computing. However, your final report to this project should have minimal overlap with your Honours or Masters research report. Send me a brief problem statement if this is the case for you.
6. Start early and plan your time effectively. Meet the deadlines to avoid late submission penalties (20% to 40%).

Due Dates

There are three due dates:

1. Friday, April 20, 2018 — the selection of a topic and your group member(s); send me an email regarding this information by the due date.
2. Friday, May 18, 2018 — the project preparation and a draft of your report; the draft is to help you for your presentation; it can be in the form of presentation slides.
3. Monday, 24:00 pm, May 21, 2018 — the submission of final report and source codes.

Deliverables and Evaluation

1. Presentation — Each group will be given a 15-20 minutes to present their project to the instructor.
2. Report —

- General problem introduction
- Methodology and pseudo codes
- Experimental setup
- Evaluation results and discussions

3. Source codes with `Makefile`, run scripts (`run.sh`), and `readme` files.

4. Total mark 20: Presentation (20%) + Report and Source code (80%).

Problems

Projects with a '*' are suitable for students who prefer working alone.

Project 1: Clustering*. What *clustering* algorithms do is to form groups, given a set of data points in d-dimensional space. Clustering is used in diverse fields, such as data analysis, image processing etc. Your task in this project is to parallelize one of the clustering algorithms — *k-means* algorithm. See [7] for an efficient algorithm for k-means. Choose one of the data sets from UCI machine learning repository [12] to evaluate your implementations.

Project 2: Sparse matrix multiplication*. Sparse matrix is a matrix where the majority of the elements are zeros. If most of the elements in a matrix are non-zeros, then the matrix is dense. The dense matrix multiplication methods are usually considered not optimal for sparse matrix multiplication. This project explores implementing sparse matrix multiplication algorithms using proper sparse matrix representation methods. Read [13, 2] to understand the problem further.

Project 3: Parallel graph algorithms. Graph representations are common in many scientific and engineering applications, and the problems requiring graph data analytics are growing rapidly. The following projects explores the challenges and limitations of parallel graph algorithms for shared memory and distributed memory systems. (You only need to choose **ONE** of the following problems.)

.1 **Single-source shortest paths***. Many problems can be expressed in terms of graphs, and can be solved using standard graph algorithms. This project will focus on parallelization of one of these algorithms. For a weighted graph $G = (V, E, w)$, the *single-source shortest paths* problem is to find the shortest paths from a vertex $v \in V$ to all other vertices in V. A *shortest path* from u to v is a minimum-weight path. In this project, you are to explore parallel formulation of *Dijkstra's single-source shortest paths algorithm* [6, Chapter 10] for undirected graphs with non-negative weights.

.2 **Connected component labeling.** Write both serial and parallel programs to solve the *connected component labelling problem*. Connected component labeling is used in computer vision to detect connected regions in binary digital images. A binary image is stored as an $n \times n$ array of 0s and 1s. The 1s represent objects, while the 0s represent empty space between objects. The connected component labelling problem is to associate a unique positive integer with every object. When the program completes, every 1-pixel will have a positive integer label. A pair of 1-pixels have the same label if and only if they are in the same component, where they are linked by a path of 1-pixels. Two 1-pixels are contiguous if they are adjacent to each other, either horizontally or vertically. For example, given the input in Table 1, (a), a valid output is shown in Table 1, (b). Note that a 0 in a particular position of the input image results in a 0 in the same position in the output image. If 2 positions in the output image have the same integer value, it means there is a path of 1s between the two positions in the input image. An easy to follow explanation can be found at [3], and a divide and conquer approach is proposed in [10].

.3 **N-queens problem.** The N-Queens problem is to place N queens on an $N \times N$ chessboard so that no two queens attack each other. If two queens are on the same row or column, they attack each other. For example, a solution for 4-queens problem is shown in Figure 1. It is obvious that the solution is not

Original

1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
0	1	1	0	1	1	1	1
0	0	0	0	1	0	0	1
1	1	1	0	1	1	0	1
1	1	1	1	0	1	1	1
0	0	0	0	0	0	1	1

(a)

Labeled

1	0	0	0	0	0	0	0
0	2	0	2	0	0	0	0
0	2	2	2	0	0	0	0
0	2	2	0	10	10	10	10
0	0	0	0	10	0	0	10
31	31	31	0	10	10	0	10
31	31	31	31	0	10	10	10
0	0	0	0	0	0	10	10

(b)

Table 1: An example of connected component labelling

unique. Then the question is in how many different ways they can be placed. A naive algorithm for N-queens is to use brute force enumeration with pruning. It tries every possible arrangements of N-queens and checks if any of them satisfies the criteria. On a $N \times N$ board, there are N^2 locations. There will be N^2 possible locations for the first queen, $N^2 - 1$ for the second one, $N^2 - 2$ for the third , and so on. Thus, in a naive approach, we have to choose from

$$\binom{N^2}{N} = \frac{N^2!}{(N^2 - N)!N!} \tag{1}$$

possible solutions. For $N = 10$, this number is $1.73e13$, and for $N = 21$, there are 314,666,222,712 possible solutions! Alternatively, we can place the queens one by one in different columns (or rows), starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes then we backtrack and return false — *backtracking algorithm*. The purpose of this project is for you to explore parallelizing backtracking algorithm. That is, solving the N-queens problem in parallel. There are numerous discussions about implementing N-queens problem. A good place to start with is [4, 5].

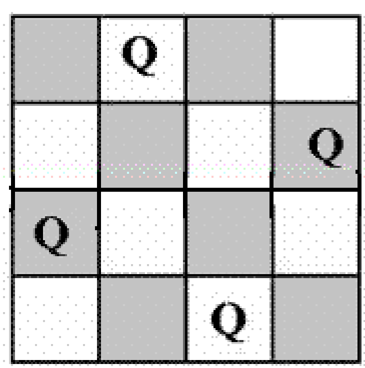

Figure 1: A solution of 4-Queens problem

Project 4: **Feed-forward fully connected neural network** An artificial neural network is an information processing method that was inspired by the way biological nervous systems function, such as brain, to process information. A neural network consists of two kinds of elements, neurons and connections, and it often has multiple layers. The connections between the neurons are assigned with weight values. These weight values need to be learned by training a neural network with sample inputs and predefined loss functions. This project explores parallel implementations of fully connected neural network for different parallel computing systems [9].

Project 5: **N-body problem solver** In an n-body problem, we need to find the positions and velocities of a collection of interacting particles over a period of time. For example, in astronomy, the problem can be about a collection of stars, while in chemistry, it can be a collection of molecules or atoms. An n-body solver is a program that finds the solution to an n-body problem by simulating the behaviour of the particles. The input to the problem is the mass, position, and velocity of each particle at the start of the simulation, and the output is typically the position and velocity of each particle at a sequence of user-specified times, or simply the position and velocity of each particle at the end of a user-specified time period.

In this project, we implement parallel n-body solvers using a selected algorithm, such as Barnes-Hut [1].

Project 6: **Parallel simulated annealing for solving the room assignment problem** Simulated annealing is a general purpose optimization technique used to find an optimal or near-optimal solution in various applications. It can be used for combinatorial optimization problems such as the travelling salesman problem. Simulated annealing is an iterative procedure that requires large amount of computing resources and is time consuming. In each iteration, the algorithm finds a new solution by a random modification to the current solution. If the cost of the new solution is less than that of the current solution, then the current solution is replaced by the new one. On the other hand, if the cost of the new solution is greater than the current one, the new solution substitute the current solution with a probability $e^{\Delta/T}$, where Δ is the difference between the values of the cost function, and T is the current 'temperature'.

In this project, we aim to investigate parallel solutions of the simulated annealing for the case study of room assignment problem. The room assignment problem solves the assignment of N (N is an even number) number of students to $N/2$ rooms. Such an assignment should minimizes the cost function defined as the sum of the conflict measures between each pair of roommates. Using simulated annealing, the room assignment problem can be solved by representing the problem as randomly assigning (Monte Carlo method) each student to a given room and calculating the cost function. See more details in [11, Chap. 10] and [8].

Project 7: **Parallel image segmentation*** In this project, we develop parallel segmentation solutions for images. You may work on any type of images, and preferably, you can work on multiple segmentation algorithms.

Project 8: **Parallel dimensionality reduction*** In this project, you can work on developing a parallel dimensionality reduction approach based on a relevant method, such as PCA. Ideally, the solution can be incorporated into the visualization of high dimensional data.

References

[1] Guy Blelloch and Girija Narlikar. A practical comparison of n-body algorithms. In *Parallel Algorithms*, Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1997.

[2] Aydin Buluç, Jeremy T. Fineman, Matteo Frigo, John R. Gilbert, and Charles E. Leiserson. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In *Proceedings of the Twenty-first Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA '09, pages 233–244, New York, NY, USA, 2009. ACM.

[3] Code Project. http://www.codeproject.com/Articles/336915/Connected-Component-Labeling-Algorithm. Accessed on 2018-04-12.

[4] Dr.Dobb's. http://www.drdobbs.com/multicore-enabling-the-n-queens-problem/221600649?queryText=N-queens Accessed on 2018-04-12.

[5] GeeksforGeeks. http://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/. Accessed on 2018-04-12.

[6] A. Grama, A. Gupta, G. Karypis, and V. Kumar. *Introduction to Parallel Computing*. Addison Wesley, 2003.

[7] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, July 2002.

[8] Milena Lazarova. Parallel simulated annealing for solving the room assignment problem on shared and distributed memory platforms. In *International conference on computer systems and technologies - CompSysTech'08*, 2008.

[9] Mike O'Neill. https://www.codeproject.com/kb/library/neuralnetrecognition.aspx. Accessed on 2018-04-12.

[10] JM Park, CG Looney, and HC Chen. Fast connected componenet labeling algorithm using a divide and conquer technique. In *Conference on computers and their applications*, 2000.

[11] Michael J. Quinn. *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill Education Group, 2003.

[12] UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/Iris. Accessed on 2018-4-12.

[13] Raphael Yuster and Uri Zwick. Fast sparse matrix multiplication. *ACM Trans. Algorithms*, 1(1):2–13, July 2005.