COMS4040A & COMS7045A Course Project

Hand-out date: 15 April 2024
Due date: 13 May 2024, 23:59; MSc Literature Survey - 23 May 2024, 23:59

General Information

1. You are expected to work individually or in a group of 2 for this project. Group projects are those with their titles underlined.
2. Each group is required to work on:
   (a) one of the projects using both CUDA and MPI, or otherwise specified in the project outline – **for both Honours and Masters students**,
   (b) and a survey on the selected HPC problem specified in Project 7 – **for Masters students only (this has to be done individually).**
3. Note that the descriptions of the project topics are only outlines of the project ideas. For your chosen project, the problem must be set up more specifically.
4. Start early and plan your time effectively. Meet the deadlines to avoid late submission penalties (20% to 40%).

Due Dates

There are two due dates:

1. Wednesday 17 April 2024, 23:59 – he selection of a topic and your group member(s). Please fill in this form https://forms.gle/ZoBX4J5vQi8TqUUr8 of your project choice, and group member(s) where applicable. In the case of a group project, only one group member is requested to fill in the form. You may start working on the project of your choice as soon as you complete this form.
2. **Final Submission** 13 May 2024, 23:59; MSc Literature Survey: 23 May 2024, 23:59

Deliverables and Evaluation

1. Report — must be written in a double-column, 10pt, single line spacing using latex; the page number should not exceed 6 pages in total. The following sections should be included:
   - General problem introduction

- Methodology (including solution, validation, and evaluation methods) and, if required, pseudo code
- Experimental setup (includes data description and performance evaluation approaches etc.)
- Evaluation of results and discussion
- Show evidence in your report that you have run your MPI implementation on the MSL cluster (using more than one node)
- In your report, proper citations and references must be given

2. Source code with `Makefile`, run scripts (`run.sh`), and `README` files. (**Do not submit any code which does not compile!**)

3. Total marks: 30

Project Topics

Note that:

- Each Honours student (or group) is requested to choose **one** problem from Projects 1 – 6 below;
- A Masters student is requested to choose **one** problem in Project 7, plus one problem from Projects 1 – 6. Each problem in Project 7 carries 15% of your coursework for this course. (Masters students must work individually on the Project 7 problem, but can work in a group for the problem from Project 1 – 6).

Project 1: Parallel machine learning algorithms. The goal of this project is to explore parallel computation in selected machine learning algorithms. You may choose an algorithm in supervised, unsupervised, or reinforcement learning. You may choose an algorithm not listed here.

(a) **Clustering.** (If you work in a group for this project, you are requested to implement at least two clustering algorithms.) 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 pattern recognition, data analytics, image processing etc. Your goal in this project is to parallelize one of the clustering algorithms — *k-means* algorithm, *spectral clustering*, or *hierarchical clustering* etc. (Jin and Jaja 2016; Rodriguez *et al.* 2019; Filippone *et al.* 2008).

(b) **Fuzzy c-means clustering.** (If you work in a group for this project, you are requested to implement at least two clustering algorithms.) See Al-Ayyoub *et al.* (2015) for a GPU implementation.

(c) **Clustering.** Items 1a and 1b can be combine as a group project.

(d) **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 the 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 (Mike O'Neill).

(e) **Parallel dimensionality reduction.** In this project, you can work on developing a parallel dimensionality reduction approach based on a relevant method, such as PCA ot t-SNE (`https://opentsne.readthedocs.io/en/latest/`). Ideally, the solution can be incorporated into the visualisation of high dimensional data.

Project 2: Scientific computing. Common problems in scientific computing include linear solvers, approximation such as Chebyshev polynomials, least-squares approximations, eigensolvers, and so on.

(a) **Sparse matrix - dense vector multiplication.** A sparse matrix is a matrix where the majority of the elements are zero. If most of the elements in a matrix are non-zero, then the matrix is dense. The dense matrix multiplication methods are usually considered sub-optimal for sparse matrix multiplication. This project explores implementing sparse matrix - vector multiplication algorithms using proper sparse matrix representation methods (Pinar and Heath 1999; Yang *et al.* 2011; Yuster and Zwick 2005; Buluç *et al.* 2009).

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.

(a) **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 parallelisation 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* (Grama *et al.* 2003, Chapter 10) for undirected graphs with non-negative weights.

(b) **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 Code Project, and a divide and conquer approach is proposed in Park *et al.* (2000).

(c) **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

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)

Labelled							
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

in Figure 1. It is obvious that the solution is not 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 parallelising the 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 Dr. Dobb's; GeeksForGeeks.

(d) **Chess or other games**

Project 4: Simulation.

(a) **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

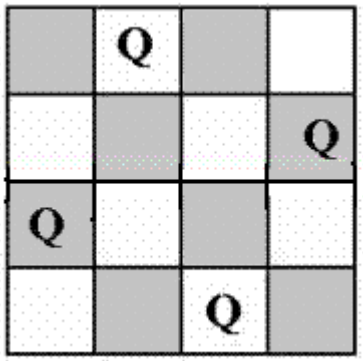

Figure 1: A solution of 4-Queens problem

user-specified time period. In this project, we implement parallel n-body solvers using a selected algorithm, such as Barnes-Hut (Blelloch and Narlikar 1997) or Fast Multipole Method (FMM).

Project 5: Optimisation.

(a) **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 substitutes 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 the 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 minimise 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 Quinn (2003, Chap. 10) and Lazarova (2008).

Project 6: Parallel sorting. Parallel radix sorting (or sample sort) for distributed memory system and GPU, respectively. Implement radix sort, respectively, for distributed memory system using MPI and GPU using CUDA, and compare it with a selected comparison based sorting using corresponding programming models (you may use a library function for the comparison based sorting).

Project 7: A survey on a selected HPC problem (This project is for masters students).

(a) **Reduced Precision Problem.** In many problems and solutions, accuracy and correctness is of vital importance, such as in medical research. In such cases, the variables used in computations are stored in a large amount of memory to capture high precision, and the associated calculations are generally more time-consuming. However, other problems do not require such precise calculations and a degree of approximation is sufficient. In such cases, variables can be made to use less memory and the overall program requires less resources. In this survey, you are required to research the ways in which reduced-precision is included in various computations and applications. Make sure to include discussions on the reliability of such calculations, numerical methods' stability, the parallel overhead patterns (more data takes longer to transfer etc.), and hardware support. Some starting papers are: Abdelfattah *et al.* (2021); Netti *et al.* (2023). (As a guideline, you should include at least 12 papers in your survey).

References

[Abdelfattah *et al.* 2021] Ahmad Abdelfattah, Hartwig Anzt, Erik G Boman, Erin Carson, Terry Cojean, Jack Dongarra, Alyson Fox, Mark Gates, Nicholas J Higham, Xiaoye S Li, et al. A survey of numerical linear algebra methods utilizing mixed-precision arithmetic. *The International Journal of High Performance Computing Applications*, 35(4):344–369, 2021.

[Al-Ayyoub *et al.* 2015] Mahmoud Al-Ayyoub, Ansam M Abu-Dalo, Yaser Jararweh, Moath Jarrah, and Mohammad Al Sa'd. A gpu-based implementations of the fuzzy c-means algorithms for medical image segmentation. *The Journal of Supercomputing*, 71:3149–3162, 2015.

[Blelloch and Narlikar 1997] Guy Blelloch and Girija Narlikar. A practical comparison of n-body algorithms. *Parallel Algorithms*, 30, 1997.

[Buluç *et al.* 2009] 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*, pages 233–244, 2009.

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

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

[Filippone *et al.* 2008] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern recognition*, 41(1):176–190, 2008.

[GeeksForGeeks] GeeksForGeeks. https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/. Accessed on 15 April 2024.

[Grama *et al.* 2003] Ananth Grama, George Karypis, Anshul Gupta, and Vipin Kumar. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Addison-Wesley, 2003.

[Jin and Jaja 2016] Yu Jin and Joseph F Jaja. A high performance implementation of spectral clustering on cpu-gpu platforms. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 825–834. IEEE, 2016.

[Lazarova 2008] Milena Lazarova. Parallel simulated annealing for solving the room assignment problem on shared and distributed memory platforms. In *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, pages II–13, 2008.

[Mike O'Neill] Mike O'Neill. https://www.codeproject.com/Articles/16650/Neural-Network-for-Recognition-of-Handwritten-Digi. Accessed on 15 April 2024.

[Netti *et al.* 2023] Alessio Netti, Yang Peng, Patrik Omland, Michael Paulitsch, Jorge Parra, Gustavo Espinosa, Udit Agarwal, Abraham Chan, and Karthik Pattabiraman. Mixed precision support in hpc applications: What about reliability? *Journal of Parallel and Distributed Computing*, 181:104746, 2023.

[Park *et al.* 2000] June-Me Park, Carl G Looney, and Hui-Chuan Chen. Fast connected component labeling algorithm using a divide and conquer technique. In *CATA*, pages 373–376, 2000.

[Pinar and Heath 1999] Ali Pinar and Michael T Heath. Improving performance of sparse matrix-vector multiplication. In *Proceedings of the 1999 ACM/IEEE conference on Supercomputing*, pages 30–es, 1999.

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

[Rodriguez *et al.* 2019] Mayra Z Rodriguez, Cesar H Comin, Dalcimar Casanova, Odemir M Bruno, Diego R Amancio, Luciano da F Costa, and Francisco A Rodrigues. Clustering algorithms: A comparative approach. *PloS one*, 14(1):e0210236, 2019.

[Yang *et al.* 2011] Xintian Yang, Srinivasan Parthasarathy, and Ponnuswamy Sadayappan. Fast sparse matrix-vector multiplication on gpus: Implications for graph mining. *arXiv preprint arXiv:1103.2405*, 2011.

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