UCT CS Honours 2020

High Performance Computing Course Assignment:

Parallelizing a genetic algorithm

(version 1)

Lecturer: Michelle Kuttel

Due date: Wednesday 11 November 2020, **10 pm**

1 Aim

This project will give you experience in coding genetic algorithms (GAs), parallel programming using both OpenMP and MPI, as well as validation and profiling of parallel codes.

You are provided with a basic C program, which you will first optimize. You will then parallelize this code with OpenMP; and then with MPI. You will then evaluate the performance of your parallel algorithms experimentally, measuring the run times of your parallel methods across a range of input sizes and numbers of cores and (for MPI) numbers of nodes and calculating the speedup of the algorithms with respect to each other and the serial version. You will be testing on UCT's hpc cluster as well as your laptop.

Finally, you will document the results in a concise report that presents and explains your optimizations, parallelization strategy and results.

2 2D particle location optimization

You are provided with a C program that uses a genetic algorithm to optimize the arrangement of interacting particles on a 2D grid. The particles can only be positioned at grid points (i.e. have x,y locations that are integers). The particles interact via a Lennard-Jones type potential, which is represented in the fitness function. The Lennard-Jones potential is the most widely used potential in molecular simulations - it describes the essential features of interactions between atoms where two interacting particles repel each other at very close distance, attract each other at moderate distance, and do not interact at infinite distance. The Lennard-Jones potential is a pair potential, i.e. no three- or multi body interactions are covered by the potential.

The program takes as input the size of the solution population, the width and length of the 2D world, the number of particles to be placed and the number of times the optimization should be done. **Do NOT change the input to the program.**

The program them outputs a file of the top solutions from each optimization. You may output additional information but **do not alter this file output.**

The code you are provided with has clear inefficiencies in the genetic algorithm, which you should correct before parallelizing it. The main issue is that you should find a better termination criterion (or criteria) for the GA. Do not alter the basic fitenss function (although it can be fun to play around with this yourself).

After optimizing the serial version, you must then parallelize it; first with OpenMP and then a separate version with MPI.

You should time the execution of all the algorithms with different input and demonstrate whether you have good speedup.

3 Code requirements

You must submit a code archive including running/installation instructions in a README file. You will need to generate suitable test files of different sizes for testing. You must also submit sample output files.

4 Report

You must also hand in a report containing the following sections.

a) Introduction

b) Serial GA

Briefly explain the genetic algorithm, including any optimizations or corrections that you made to the original code.

c) Parallel GA: OpenMP

Describe and justify how you parallelized the GA for OpenMP. Your description must explain any changes you made to the algorithm for parallelization. You must explain how you avoided data races in the OpenMP code as well as how you ensured parallel efficiency.

d) Parallel GA: MPI

Describe and justify how you parallelized the GA for MPI. Your description must explain any changes you made to the algorithm. You must explain how you avoided data races in the OpenMP code as well as how you ensured parallel efficiency.

e) Benchmarking

Contrast and compare your results for the 3 algorithms on the two computers. Explain how you timed your algorithms (with different input, cores etc.) and how you measured speedup. Make sure that you do this properly, for different input sizes and consider the different components of the application. Is there evidence of strong scaling or weak scaling? You must include speedup graphs, but do not include graphs of raw run times. All table data must be graphed – do not expect the reader to process tables in their head. Do not include voluminous tables in the text – add them as appendices and graph the results. Graphs should be clear and labelled (title and axes). You should compare and contrast OpenMP and MPI and discuss any problems or difficulties encountered.

f) Conclusions

Summarize the main results and findings. Discuss what you achieved, where the issues were and come to some conclusions about the assignment. Reflections and observations would be interesting to read here.

Please do NOT ask the lecturer for the recommended numbers of pages for this report. Say what you need to say: no more, no less.

5 Assignment submission requirements and assessment rules

- **You may do the assignment alone or in a group of 2 (maximum).**
- Your submission archive must consist a technical report and your solution code archive (including a **Makefile** for compilation/running and **sample output files**).
- Submit your report as a separate (from the source code) file **IN PDF FORMAT** named **HPC**_*STDNUM001*.**pdf,** replacing STDNUM001 with your student number (or both student numbers if a group). **Incorrect formats will incur a 5% penalty.**
- Upload the files and **then check that they are uploaded.** It is your responsibility to check that the uploaded file is correct, as mistakes cannot be corrected after the due date.
- A late penalty of 5% a day (or part thereof) applies to this assignment.
- The deadline for marking **queries** on your assignment is **one week after the return of your mark.** After this time, you may not query your mark.
- Note well: submitted code that does not run or does not pass standard test cases will result in a mark of zero. **Any plagiarism, academic dishonesty, or falsifying of results reported will get a mark of 0 and be submitted to the university court**.

I hope you enjoy this assignment.