CAB401:

High Performance and Parallel Computing

Assignment

Due: 22/10/2021 (with proposal due by 06/09/2021)

Worth: 50%

Individual

Overview

You are to select a real world software application and manually *parallelize* it. That is, take any application that is not written in an explicitly parallel fashion and transform it so that it executes as efficiently as possible on a particular parallel computer. The software application can be whatever you like but you will obviously need access to its source code. It can for example be an open source application or an application that you have developed yourself, or perhaps one from your workplace. To be amenable to parallelization it will need to be relatively computationally intensive, i.e. it will need to perform sufficient computation so that parallelizing that computation will potentially produce a noticeable difference in perceived execution time. For example, a word processing application would probably not be a good candidate as such applications are already normally adequately responsive to user interaction. Note, that some applications are more amenable to parallelization than others. It is not expected that a perfect linear speedup will be achieved for all applications – simply that your parallelization achieves as much performance improvement as is available.

Hardware

You can use any parallel hardware that you have access to. You can make use of parallel computers provided by QUT or any other parallel computers you personally you have access to. It can be any form of parallel computer, e.g. multi-core, cluster, SMP, shared memory, distributed memory, GPU, etc. See criteria below regarding scalable parallelism.

Software

Again you can use whatever software that you have access to. This includes compilers, profilers, debuggers, libraries, etc. Some such software is available through QUT. You may use whatever programming language you wish and whatever parallel frameworks and libraries that you have access to.

Submission

Project Proposal, due 03/09/2021:

Submit online form, describing:

1. A brief description of the sequential application that you have selected to parallelize. What does it do? Where did you find it? (1 paragraph max)
2. Discuss whether you think the proposed application performs sufficient computation so that parallelizing it will potentially produce a noticeable difference in perceived execution time. (1 paragraph max).
3. What parallel hardware and parallelization language/framework are you considering? E.g. targeting NVidia GPU programmed using CUDA. (1 paragraph max).

The project proposal is designed to give you constructive feedback and to ensure you are on a productive path prior to final submission.

Final Submission, due 22/10/2021, a zip file including both:

1. A report of 10-15 pages (not including appendices) describing your outcomes. The report should address the following criteria:
    a. An explanation of the original sequential application being parallelized, what it does (black box) and how it works (a high level description of software's design/architecture). This might include call graphs, class diagrams, etc – whatever you find useful to describe the structure of the original sequential application.
    b. Your analysis of potential parallelism within the application. This might include identification of existing loops or control flow constructs where parallelism might be found. Explanation of the data and control dependences that you analysed to determine which sections of code were safe to parallelize. Which of these is likely to be of sufficient granularity to be worth exploiting? Is it scalable parallelism? A discussion of changes required to expose parallelism, such as replacing algorithms or code restructuring transformations.
    c. How did you map computation and/or data to processors? Which parallelism abstractions or programming language constructs did you use to perform synchronization?
    d. Timing and profiling results, both before and after parallelization and a speedup graph.
    e. How did you test that the parallel version produced the exact same results as the original sequential version?
    f. A description of the compilers, software, tools, and techniques you used to parallelize the application.
    g. The story of how you overcame performance problems/barriers (e.g. load imbalance, memory contention, granularity, data dependencies, etc) to improving parallel performance.
    h. An explanation of the code that you added or modified to parallelize the application (including source code line count).
    i. Reflect on your outcome – What have you learnt? How successful was your attempt? Do you think you've done as well as is possible? What might you have done differently?

2. Your source code (both before and after versions) together with instructions for compiling, running, hardware requirements and realistic input data sets.

Assessment Criteria

Criteria	Standards			
	Unsatisfactory	**Satisfactory** (50%)	**Good**	**Excellent** (100%)
Analysis of original application (10 marks)				Demonstrates a deep understanding of the original application, its structure and performance issues/bottlenecks. (Must include identification and discussion of data and control dependencies and detailed before and after detailed profiling results).
Use of tools and techniques (10 marks)				Demonstrates advanced use of a wide variety of parallel programming software, tools and technologies.
Optimal Speedup (10 marks)				Obtained very close to the best possible performance improvement for the application (*must be more than 4 cores for excellent*). (Must include a correctly constructed speed-up graph).
Overcoming Barriers (10 marks)				Demonstrated great skill and effort to achieve this outcome and overcome significant barriers to improved performance. (Include interesting before and after code snippets)
Report (10 marks)				Report is well structured, easy to read, reflective and insightful.