Data Structures and Algorithms 2
Assessment

Ruth Falconer and Adam Sampson
School of Arts, Media & Computer Games

Introduction

In this assessment, you will demonstrate your achievement of the following learning outcomes through a software project and presentation:

- Be aware of the standard techniques of software performance measurement, including profiling, and apply these techniques to identify performance bottlenecks in real programs.
- Understand the emerging importance of parallel programming in modern software development, and experiment with the performance impact of parallelising parts of an application.
- Describe a variety of application-specific algorithms (sorting/numerical/image processing) and associated data structures in common use, and discuss the benefits and limitations of parallelisation.

You must **work on your own** for this coursework.

If you have any questions, please contact Ruth (r.falconer@abertay.ac.uk) or Adam (a.sampso@abertay.ac.uk)

Requirements

Your application must:

- Implement a parallelised algorithm that solves a problem;
- Make use of appropriate parallelisation strategies;
- Quantify and reason about the (positive and/or negative) impact of your design choices.

Your parallelised application can be for CPU multicore or GPU devices (or both!). Depending on which your application must also demonstrate:

CPU applications

- running at least three threads, with at least two different thread functions;
- sharing resources safely between threads (e.g. using mutexes, atomic operations or barriers);
- signalling between threads (e.g. using semaphores, channels or condition variables).

GPU applications

- Running at least one kernel with appropriate task decomposition;
- Consideration of memory access patterns;
- Sharing resources safely between threads (using barriers or atomic operations).

Choosing an application

You may implement any application you like provided it meets all of the requirements above – see the Module FAQ's for some ideas. For example, you might choose:

- An interactive Mandelbrot set – parallelised using CPU or GPU. You can use the lab exercises as a basis for this, so it's the most straightforward option;
- a complex game, with parallelised collision detection, enemy AI, and/or animation;
- image processing application useful for post processing effects;
- a parallel password cracker or other cryptographic application;
- a parallelised version of a program you've built for a different module. (If you do this, please clearly indicate what you've changed from your original version.)

Your application may run on any operating system or platform. You may use external libraries such as Boost, TBB, SFML - provided these are clearly acknowledged and don't prevent you from demonstrating the required learning outcomes. You do not need to stick with the API's used in the laboratories. If you are unsure whether your algorithm would meet the requirements, please talk to us.

Performance Evaluation

Your application should be constructed so that you can vary the number of threads or thread groups being used and measure the application's performance in an appropriate way. For example, if you built a game with parallel AI, you might choose to measure the time spent on AI in each frame.

Once your application is complete, you should measure its performance across a range of numbers or groupings of threads where appropriate. The presentation must include information on:

- the CPU or GPU specification used for testing depending on your chosen application;
- the results you have measured, including graphs;
- an explanation of these results in terms of the design of your application and your understanding of processor and memory architecture (i.e. a description of why it performs the way it does) of the CPU or GPU.

If it isn't obvious how to compile or use your application, please include a README file with any necessary instructions.

The Presentation

During week 14, week beginning 24th April, you will give a short presentation describing the design of the parallel aspects of your application. Your presentation should last no longer than ten minutes; there will be approximately five minutes for questions and discussion afterwards. Your presentation should cover the following topics:

- the purpose of your application and the problem you're trying to solve through parallel programming (assume we know how standard algorithms work);
- how the parallel parts of the code are structured, and how they are integrated with the rest of your application;
- how your application makes use of threads, and how interactions between threads are managed safely;
- presentation of key results of the performance evaluation (see above for what to include);
- a critical evaluation of the effectiveness of your solution, with reference to your performance evaluation.

You must explicitly justify your technical choices, and quantify their effect during performance evaluation, using the knowledge you've gained from undertaking the Module. Presentations will take

place during the week starting April 24th 2016. Upload your assessment to Blackboard following the submission guidelines below by 23:59 on Tuesday 25th April 2016.

A presentation time will be allocated; if the time suggested isn't possible for you, please get in touch with Ruth (r.falconer@abertay.ac.uk) soon as possible to arrange an alternative.

Some hints

You should aim to make your application perform as well as possible – it shouldn't contain debugging/display code, for example, and you should make use of profiling in order to identify where performance bottlenecks exist.

When presenting your application's performance, you should follow the best practices for performance measurement and comparison described in CMP201 early lectures. In particular, you should use appropriate statistics (do you have a significant difference? what's the effect size?) and try to avoid sources of error in measurement.

Submission

You must submit a ZIP file containing the following:

- the complete source code for your application;
- a ready-to-run executable for your application (e.g. a Windows .exe file);
- your presentation slides, in PDF format.

To reduce the size of your ZIP file, please ensure that you have cleaned out any temporary files from your application's source code before submission – if you've used Visual Studio, then delete any .obj, .ipch, .vsp, .vsps, .suo and .sdf files. For external libraries, tell us a download link rather than including a copy of the library.

You must submit your ZIP file by 23:59 on Tuesday 25th April 2017, using CMP202's “Submit Module Assessment” page on Blackboard.

Feedback will be returned by Tuesday 16th May 2017 (three weeks after the submission date).

Grading criteria

This is a **summative** assessment: your final grade for CMP202 will be determined by your performance at the end of the module, as demonstrated in this assessment. Feedback on your work will be returned electronically 15 working days after the submission deadline.

Grade	Code quality & organisation	Parallel programming & design	Performance evaluation	Presentation
A	Extremely high code quality and structure, following generally accepted best practices and demonstrating a high degree of professionalism	Application demonstrates comprehensive understanding of parallel programming and computer architecture	Outstanding evaluation, considering all appropriate sources of error and providing accurate, insightful explanation	Outstanding quality presentation, giving a clear, correct and critical description of the project's design with appropriate justification of technical choices
B	Very good code quality and structure	Application demonstrates very good understanding of parallel programming and computer architecture	Very good evaluation, considering several appropriate sources of error and explaining results accurately	Very good presentation, giving a clear and correct description of the project's design with some critical evaluation and mostly appropriate justification
C	Good code quality, with some minor inconsistencies	Meets all requirements, demonstrating competence and confidence at parallel	Competent performance evaluation, with some mitigation for error and a plausible	Competent presentation, giving a generally clear and correct

		programming	explanation of the results	description of the project's design
D	Adequate code quality, with substantial scope for improvement	Meets all requirements	Adequate performance evaluation	Adequate-quality presentation
MF	Poor code quality	Fails to meet all requirements	Fails to consider potential sources of error or provides inadequate explanation of results	Presentation fails to convey a sufficiently clear description of the project's design
F	Performance well below the threshold level, with only limited evidence of achievement			
NS	There is no submission, or the submission contains no relevant material			