# CMP202 Data Structures and Algorithms 2 Assessment

Ruth Falconer, Salma ElSayed, Chris Acornley and Adam Sampson
School of Design and Informatics

## 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)

## 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 demonstrate:

### *CPU applications*

- running at least three threads, with at least two different thread functions (or two different tasks, in a task-based system);
- 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;
- Consider strategies for sharing resources safely between threads (using barriers or atomic operations) if appropriate;
- Consideration of strategies to enhance performance of the GPGPU application (limit thread divergence, memory access patterns etc).

## 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.

## Presentation

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 (e.g. in terms of patterns discussed in the module), 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.

Your presentation should last no longer than ten minutes; there will be approximately five minutes

for questions and discussion afterwards.
We will assign you a presentation time during the week starting Monday 10th May 2021 and your presentation will be in the following week – week commencing Monday 17th May 2021; a list of presentation times will be available on MyLearningSpace. If the time we suggest isn't possible for you, please get in touch with Ruth as soon as possible to arrange an alternative.
You must submit your slides in PDF format to the "Presentation slides" assignment on MyLearningSpace **before your presentation**.

## Submission

1. You must submit a ZIP file to the "Project" assignment on MyLearningSpace by 23.59 on Tuesday 11th May 2021. The ZIP file should contain the following:
   • the complete source code for your application;
   • a ready-to-run executable for your application (e.g. a Windows .exe file);
2. Please also make sure you have submitted your PDF slides to "Presentation slides" assignment before your presentation.

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.

Feedback will be returned by 11th June 2021.

## Grading criteria

This is a **summative** assessment: 80% of your final grade for CMP202 will be determined by your performance at the end of the module as demonstrated in this assessment.

| **Grade** | **Implementation quality** | **Algorithm parallelisation** | **Parallelisation considerations** | **Performance evaluation** |
|---|---|---|---|---|
| **A** | High-quality application that operates flawlessly and follows best practice for code quality | Application demonstrates comprehensive understanding of algorithm parallelisation, with correct implementation and at least three threads with two different thread tasks (CPU) and/or at least one GPU kernel with appropriate thread configuration (GPU) | Entirely appropriate argument for parallelisation strategies and protecting shared resources with clear justification of choice | Outstanding evaluation, considering sources of error and providing appropriate graphical presentation and statistical evaluation of results with reference to expected performance |
| **B** | High-quality application, with appropriate structure and only minor flaws in code quality or operation | Application demonstrates good understanding of algorithm parallelisation with mostly correct implementation and at least three threads with two different thread tasks (CPU) and/or at least one GPU kernel with appropriate thread configuration (GPU) | Mostly appropriate argument for parallelisation strategies and protecting shared resources but some minor weaknesses | Very good evaluation, considering appropriate sources of error and presenting results appropriately |
| **C** | Acceptable-quality application | Application meets all | Parallelisation strategies and protection of shared resources used are | Competent performance |

| | | | | |
|---|---|---|---|---|
| | (lab-exercise quality) with some flaws in code quality or operation | requirements, with some flaws in algorithm implementation and at least three threads with two different thread tasks (CPU) and/or at least one GPU kernel with appropriate thread configuration (GPU) | generally reasonable, but argument contains flaws or omissions | evaluation, with some mitigation for error and a reasonable presentation of the results |
| **D** | Application has substantial flaws in code quality or operation | Meets all requirements but has substantial flaws in implementation of the parallelised algorithm | Adequate choice of parallelisation strategies and protection of shared resources based on application's requirements, with serious flaws in justification | Adequate performance measurement and evaluation |
| **MF** | Unsatisfactory application quality | Fails to meet all requirements | Inappropriate choice of parallelisation strategies and protection of shared resources structures, or inadequate justification of choice | Fails to consider sources of error or provides inadequate presentation of results |
| **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 | | | |