**Introduction**

1. Background and motivation:
   * Provide a brief overview of the current state of parallel programming education, and explain why it is important.
   * Describe the challenges and barriers that novice learners face when learning parallel programming.
   * Explain the motivation and rationale for the project, and why it is important to develop novice-friendly parallel programming exercises.
2. Research question and objectives:
   * State the research question that the project aims to answer. MC: there doesn’t \*have\* to be a big “research question”, it’s also ok to have an overall goal of designing and building something useful, and evaluating it, or any mixture of this. The key is to be clear.
   * Explain the specific objectives and goals of the project, and how they relate to the research question (MC: or overall goals)
   * Describe the expected outcomes and contributions of the project, and how they will advance the field of parallel programming education.
3. Brief overview of the dissertation structure:
   * Provide a brief overview of the sections and chapters of the dissertation, and how they are organized.

**Literature review**

Done

**Methodology**

MC: it would be good to clearly separate this into two aspects: roughly “Design” and “Implementation”. “Design” will cover your overall design of the work you did. There could be several parts to this: design of the evaluation criteria, and design of the exercises, in terms of the principles used and how these address your project goals. Separately you would discuss “Implementation” issues, including more details on how you went about creating the exercises and the questionnaire.

1. Overview of parallel programming exercises developed:
   * Describe the parallel programming exercises. Include a brief overview of each exercise, what is their characristics, for example, is this exercise intent to be a warm-up or what parallel concept it will show MC yes, good, and these should be motivated by what you have already written about design.
2. Description of the programming languages and frameworks used: MC: yes, good, this is all “implementation”.
   * Explain the programming languages and frameworks that you used to implement the exercises.
   * Describe the features and characteristics of each language and framework, and explain why chose them for the exercises.
   * Include any important technical details or configuration settings that are relevant to the implementation of the exercises.
3. Evaluation criteria and metrics:
   * Describe the criteria and metrics that you used to evaluate the effectiveness and quality of the exercises. MC this is design
   * Explain how you determined these criteria and metrics, and why they are important for evaluating parallel programming exercises. MC so is this
   * Provide specific examples of how you measured each criterion or metric. MC and this is implementation
4. Testing and validation procedures: MC yes, good.
   * Explain how you tested and validated the parallel programming exercises.
   * Describe the testing procedures and methods that is used.
   * Describe the validation procedures and methods that is used, how is the exercises verified the correctness and effectiveness of the exercises.

**Result and analysis**

1. Description of the implementation and performance results:
   * Provide a detailed description of the implementation of the parallel programming exercises. MC this is implementation and belongs above
   * Describe any challenges or obstacles encountered during implementation, and how to overcame them. MC so is this
   * Present performance metrics for each exercise (speedup over different thread number and data size). MC and this is “design” I think
2. Evaluation and analysis of the exercises using the chosen criteria: MC: yes this bit is the real “results” section
   * Evaluate the effectiveness of each exercise using the evaluation criteria and metrics you established in the methodology section.
   * Compare the result from different platform, PC, Dice and MacBook laptop
   * Discuss any interesting or unexpected findings. MC yes, this is analysis
3. Related work reference MC: yes but this doesn’t belong here, I suggest including it with the background (could call that chapter Background and Related Work
   * Try to refer to some related work, including tools or exercises of parallel programming.
4. Limitations and potential for future research: MC: yes but this belongs in the final chapter
   * Identify the limitations and potential weaknesses of the project, including any aspects that could be improved or expanded in future research.
   * Discuss potential avenues for future research that build on the findings and insights of this project.

**Conclusion**

MC maybe call this Reflections and Conclusion and include the material from point 4 above.

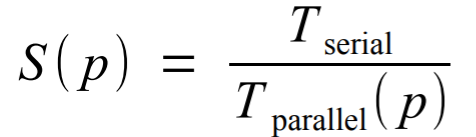
1. Summery of the result and analysis section MC “summary” not “summery”
   * Restate the question and objectives of the project, and explain how they were addressed in the dissertation.
2. Reflection
   * Reflect on the strengths and limitations of the project, including any challenges or obstacles that were encountered and how they were addressed.
   * Discuss any potential improvements or modifications that could be made to the exercises or the evaluation criteria in future research.
   * Self-reflection? MC this is what you have already mention in the “Reflect ,,,” point above.

**References**

**Appendices**

**Criteria:**

1. **Speedup:** can the exercise actually gain speed up?



p = Number of processors / cores

Tserial = Execution time of the serial program

Tparallel (p) = Execution time of the parallel program with p processors / cores

1. **Strong Scaling**

* Efficiency 
* Increase the number of processors p keeping the total problem size fixed result the efficiency remains unchanged
* Goal: reduce the total execution time by adding more processors

1. **Weak Scaling**

* increase the number of processors p and the problem size to let efficiency remains unchanged

1. **Difficulty to Verify**

* Can students easily verify that their implementation of the problem is correct?

1. **Speedup Visualization**

* Is the problem able to show the speedup in visible, sensible way, rather than just output time difference.

1. **Interesting**

* Is the question interesting? When students do the question, does feels funny, and they are happy to proceed

1. **Difficulty**

* Is the exercise easy enough for a beginner to complete?

**Questions**

1. **Scrabble (inf1 OOP tutorial 1)**

* a game where players get pieces with letters that they use to form words in a crossword-like puzzle. Each letter is worth a different amount of points according to this table:

A is worth 1 — B is worth 3 — C is worth 3 — D is worth 2

E is worth 1 — F is worth 4 — G is worth 2 — H is worth 4

I is worth 1 — J is worth 8 — K is worth 5 — L is worth 1

M is worth 3 — N is worth 1 — O is worth 1 — P is worth 3

Q is worth 10 — R is worth 1 — S is worth 1 — T is worth 1

U is worth 1 — V is worth 4 — W is worth 4 — X is worth 8

Y is worth 4 — Z is worth 10

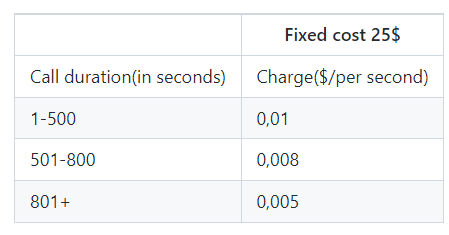
* A word is worth the number of points that its letters add up to. For example, the word ”gear” is worth 2 + 1 + 1 + 1 = 5 points.

1. **Mandelbrot set graph rendering**

1. **Find student that fail the course**

* Student is a class, it contains student’s information, including their course taken and grade
* Knowing grade <40% means fail, given a list of students, find out who failed the course
* To be complicated, add course’s restriction, like student must pass both coursework and exam.

1. **Find energy cost**

* Users use gas and electricity; the total amount of cost is combined of electricity and gas.
* 
* Exemplary cost calculation is like this table, depending on the amounts used, the cost will be different.

1. **Encoding Morse Code**

* Implement a dictionary with morse code and alphabet
* Input a paragraph of English text, get the respect morse code cyphertext. Using the dic.

1. **Find Max Min Sum of an Array**

* Data parallel version
* Task parallel version
* Data parallel and task parallel combined
* Sequential