COMP1680 (2024/25)	**Clouds, Grids and Virtualisation**		**Contribution: 100% of course**
Course Leader: Dr Catherine Tonry	**Coursework**		**Deadline Date: Friday 29th November 2024**

This coursework should take an average student who is up-to-date with tutorial work approximately 40 hours.

Feedback and grades are normally made available within 15 working days of the coursework deadline.

Learning Outcomes:

- Characterise and critically evaluate high performance computing based architectures and their suitability for given applications.
- Implement and execute applications using shared and distributed memory programming paradigms.
- Describe and critically discuss the roles and applications of cloud and grid computing.

Coursework Submission Requirements

- An electronic copy of your work for this coursework must be fully uploaded by 23:30 on the Deadline Date of **Friday 29/11/2024** using the link on the coursework Moodle page for COMP1680.
- For this coursework you must submit a single PDF document. In general, any text in the document must not be an image (i.e. must not be scanned) and would normally be generated from other documents (e.g. MS Office using "Save As .. PDF"). An exception to this is hand written mathematical notation, but when scanning do ensure the file size is not excessive.
- There are limits on the file size (see the relevant course Moodle page).
- Make sure that any files you upload are virus-free and not protected by a password or corrupted otherwise they will be treated as null submissions.
- Your work will not be printed in colour. Please ensure that any pages with colour are acceptable when printed in Black and White.
- You must NOT submit a paper copy of this coursework.
- All coursework must be submitted as above. Under no circumstances can they be accepted by academic staff

The University website has details of the current Coursework Regulations, including details of penalties for late submission, procedures for Extenuating Circumstances, and penalties for Assessment Offences. See http://www2.gre.ac.uk/current-students/regs

Detailed Specification

This Coursework is to be completed individually

Part 1: Parallel processing using cloud computing (40 marks)

A company you work for is considering whether to use Cloud Computing or an onsite HPC for parallel codes. Make a recommendation to which you think they should use and why. Assume the company is a small consultancy with 30 consultants using around 1400 CPU hours each a month.

Your recommendation should include:

1) An analysis of the advantages and disadvantages of the different commercial providers (e.g. Azure, AWS, Google Cloud) over an onsite HPC and compared to each other for multicore workloads using batch processing. Your answer should include referenced examples.
2) A comparison of the financial cost of an onsite HPC vs using cloud storage. You should include numerical examples in your comparison
3) Your recommendations to the company. This should include what platform you think they should use and why. You must justify this using the information in parts 1 and 2.
4) A reference list, please use Harvard Style Referencing, you should reference everything including where you got your costs from. (https://www.scribbr.co.uk/referencing/harvard-style/)

Marks will be awarded based on the following criteria:

15 Marks: Your analysis of the different platforms Vs HPC
10 Marks: Your cost comparison
10 Marks: Your recommendations
5 Marks: Your use of references

This section should be approximately 1500 words.

Part 2: Parallel Programming Exercise (50 Marks)

To complete this assignment you will need the source code provided at the following URL.

https://moodlecurrent.gre.ac.uk/mod/resource/view.php?id=2163909

You are provided with a C program code (called jacobi2d.c) that solves a rectangular 2 dimensional heat conductivity problem using the Jacobi iterative method.

This code can be compiled and linked to produce a conventional executable file called jacobiSerial by using the following commands:

```
gcc jacobi2d.c –o jacobiSerial
```

To run the executable type in the executable name: jacobiSerial

As you implement each of the following 4 steps make sure that you retain and do not overwrite previous versions of your solutions.

Compile and execute the codes using the University HPC. Note this is a shared resource with a queue may become busy near the hand in date so make sure you give plenty of time to run your code and don't leave it to the last minute. If you are unsure how to use the HPC please check the lab notes and the instructions on Moodle.

Please ensure you follow these steps carefully and with the code provided, work based on other Jacobi codes will not be marked and receive 01 mark and not be marked further.

Optimisation an incorrect parallelisation can break your code so check it still gives the same answer at every step.

Step 1 (10 Marks)

You are required to compute a temperature distribution for a rectangular 2D problem with boundary conditions set at top 15°C, bottom 60°C, left 47°C and right 100°C with a range of problem sizes. To do this you are required to modify the codes to:

- reflect the boundary conditions described above
- report the execution time Record the run-time of your code under a range of problem sizes greater than 100x100 using different levels of compiler optimization.

Be advised that:

- it is possible that aggressive optimization will break the code
- you will need to stop the results from printing if you are to obtain realistic measurements of the execution time.

Step 2 (20 Marks)

You are then required to modify the applications you created in step 1 to produce a basic parallel version of the codes using OpenMP. The following commands will compile your parallel version on a platform that has OpenMP installed:

```
gcc -fopenmp jacobiOpenmp.c –o jacobiOpenmp
```

The parallel codes must include timers to report the parallel run-time of the code. This version must be tested to establish correct operation using 1, 2, 4 and 8 threads/processors, regardless of performance. (These versions may run on any platform you choose as performance is not an issue at this stage.)

Include in your report, the result for a 20x20 problem size for 1,2,4 and 8 processors to demonstrate the code works correctly. This should be as a screen shot of the outputted grid.

Step 3 (20 Marks)

Using the HPC and the SLURM queue you are to run performance tests with the OpenMP implementation you created in step 2. This will require that you remove most of the print output from the code and increase the problem size to provide sufficient work to demonstrate useful speedup. You are expected to provide speedup results:

• for a range of problem sizes, you are unlikely to see much speedup for small domains, use the same problem size as step 1

• for a range of number of threads (from 2 up to 8 threads) In calculating the speedup of your parallel code you should use the optimized single processor version of your code you produced in step 1 and compare to this. You will need to apply similar compiler optimizations to your parallel code. Please list your runtimes in a suitable unit.

This section is required to provide details of your implementation of steps 1 to 3 as described above. You should include discussion of your solutions and provide a clear description of; the code changes you have implemented including code snippets, your compilation and execution processes and your test cases. For step 3 you are expected to provide tabular and graphical results. Your zip file should provide suitably named source code files for each of your implementations (e.g. step 1, step 2, step 3).

This section should be approximately 1500 words.

A further 10 marks will be awarded for your use of English in the report.

Grading Criteria

Marks allocated according to the following rubric:

	Part 1: Analysis of the different platforms (15 Marks)	Part 1: Cost Comparison (10 Marks)	Part 1: Recommendations (10 marks)	Part 1: Referencing (5 marks)	Part 2: Step 1 10 Marks	Part2: Step 2 20 Marks	Part2: Step 3 20 Marks	Report Layout and Language (10 marks)
0-49%	unsatisfactory analysis that doesn't cover the basics	An unsatisfactory analysis that doesn't cover the basics	An unsatisfactory recommendation that doesn't cover the basics	Unsatisfactory referencing that doesn't use the correct style or sources	An unsatisfactory step, lacking either timings of the code or changes to boundary conditions.	An unsatisfactory step, lacking the required parallelisation steps to the code.	An unsatisfactory step, there is no evidence of timing the code or it is very limited.	Unsatisfactory layout and language that is disjointed and confusing
50-59%	A satisfactory analysis that could be improved by considering more points	A satisfactory analysis that could be improved by considering more points	A satisfactory recommendation that could be improved by considering more points	Satisfactory referencing, but more sources needed	A satisfactory step, however, there are errors in the code or lack of detail in the report.	A satisfactory step, however, there are errors in the code or lack of detail in the report.	A satisfactory step, however, there are errors in the timings or lack of detail in the report.	Satisfactory layout and language that could be improved by rearranging the text and better grammar
60-69%	A good analysis that covers most of the salient points, could be improved however	A good analysis that covers most of the salient points, could be improved however	A good recommendation that covers most of the salient points, could be improved however	Good Referencing, but could be improved with more variety of sources	A good step, however, the code is mostly correct and documented but there is a lack of discussion of the results.	A good step, however, the code is mostly correct and documented but there is a lack of discussion of the results.	A good step, however, the timings are mostly correct and documented but there is a lack of discussion of the results.	Good layout and language, but could be improved by better grammar
70-79%	A very good analysis that covers most of the salient points	A very good analysis that covers most of the salient points	A very good recommendation that covers most of the salient points	Very good Referencing, but could be improved with more variety of sources	A very good step, however, the code and timings are correct but there are a few limitations in the report	A very good step, however, the code is correct but there are a few limitations in the report.	A very good step, however, the timings is correct but there are a few limitations in the report.	A very good layout and language, but could be improved by better grammar
80-89%	An excellent analysis that covers almost everything	An excellent analysis that covers almost everything	An excellent recommendation that covers almost everything	Excellent referencing, may need more care with style or sources	An excellent step, however, the code and timings are correct but there is a lack of true critical analysis in the report	An excellent step, however, the code is correct but there is a lack of true critical analysis in the report	An excellent , however, the timings is correct but there is a lack of true critical analysis in the report	An excellent layout, but could be improved by better grammar
90-100%	An outstanding analysis, very little could be improved.	An outstanding analysis, very little could be improved.	An outstanding recommendation, very little could be improved.	Outstanding referencing, very little could be improved.	An outstanding step, the code and timings are correct, and the report details them fully.	An outstanding step, however, the code is correct, and the report details them fully.	An outstanding however, the timings is correct, and the report details them fully.	Outstanding Layout, very little could be improved.

If you are unsure about any of these instructions, then please email your tutor or make an appointment to see your module leader as soon as possible.