School of Computing
6004CEM Parallel and Distributed Programming

Assignment Brief April 2023

Module Title Parallel Distributed Programming	Individual	Cohort - Sept	Module Code **6004CEM**
Coursework Title (e.g. CWK1) Report Portfolio			Hand out date: 3/04/2023
Lecturers Vasuky Mohanan			Due date: **3/07/2023**
	Coursework type Practical work		% of Module Mark 100%

Module Learning Outcomes Met by this assessment

1. Demonstrate good knowledge of the concepts of parallel computing and its use in both commercial and scientific environment.
2. Apply and evaluate parallelism techniques to several practical problems.
3. Compare and contrast Distributed and Parallel Computing.
4. Apply above knowledge and skills to solve a distributed architecture problem.

TASK 1:

Grading

This work is marked out of **100** and represents **30%** of the overall module grade.

Prerequisites

Word Limit 1600 Words,+-**10%,** not including the bibliography. At least ten references are expected, and all references included in the bibliography should be cited in the main text.
Citations and references must use the **APA** referencing system. This is the same one you are required to use in your final year project.

Use current literature to justify your responses and include citations in the body of your text for the answers to each question.

Marking Distribution

Parallel Element – 50%
Distributed Element – 50%

The Topics to cover in this written piece are as follows:

1 – Parallelism (800 words)

Write a technical report that analyses OPENMP and CUDA. Your report should cover the following:

- **a)** Explanation on both technologies
- **b)** Differences and similarities in both technologies
- **c)** Situations and scenarios where each is more suitable
- **d)** Future trends of both technologies

2 – Distribution (800 words)

Write a technical report that analyses OPENMP and MPI Your report should cover the following:

- **a)** Explanation on both technologies
- **b)** Differences and similarities in both technologies
- **c)** Situations and scenarios where each is more suitable
- **d)** Future trends of both technologies

	Essay Content	Citations/Refencing	Layout/Readability	
First ≥70	All points answered to an excellent level of detail.	Excellent use of Citations throughout the work. Hardly any uncited references to external sources	Well laid out and easy to read. Few to no grammatical errors. No unrelated images used.	**First ≥70**
Upper Second 60-69	All points answered to a moderate level of detail.	Reasonable use of Citations throughout the work. More frequent use of uncited references to external sources	Well laid out and easy to read. Some grammatical errors. Possibly some unrelated images used.	**Upper Second 60-69**
Lower Second 50-59	Some or all points answered to a reasonable level of detail.	Inadequate use of Citations throughout the work. Too much reliance on the same few sources. Hardly any cited references to external sources	Issues with either layout or ease of reading. Some grammatical errors. Possibly some unrelated images used.	**Lower Second 50-59**
Third 40-49	Some or all points answered to a poor level of detail.	Citations used poorly, external sources not credited when used.	Bad layout or other problem that interferes with reading. Too many grammatical errors. Possibly some unrelated images used.	**Third 40-49**
Fail <40	Some or all points insufficiently answered.	Citations used inadequately or not present.	Layout is too poor to be acceptable as an academic work.	**Fail <40**
Late	0	0	0	**Late**

TASK 2:

Portfolio

This is Portfolio has two elements, covering both Parallelism and Distribution.

This is marked out of 100 and represents 70% of the marks for this Module. Each portfolio element will be graded separately using a Rubric, and the results from this will be used to decide your final grade.

This Portfolio has two main parts, One for Parallel Computing, and one for Distributed Computing.

These Portfolio elements are split into several tasks to give you the opportunity to stop at a definite finish point if you reach the limit of your abilities, but they will all be graded together in their respective rubrics. Failing to complete all aspects will impact the final grade awarded.

Parallel Element Contribution – 50% of portfolio grade.
Part A Contribution – 5% of portfolio grade.
Part B Contribution – 20% of portfolio grade.
Part C Contribution – 25% of portfolio grade.

Parallel Programming Portfolio Element

Module Learning Outcomes Met by this Portfolio Element:
4. Apply Parallelism to several problems.

Preamble

To have the best chance of a good grade in this element you are strongly advised to complete the exercises for Parallelism. Every aspect must be written in **C/C++** and make use of **OpenMP**.

Parts **A**, **B**, and **C Must** be submitted as **SEPARATE** programs, if they are combined you will lose marks.

Part A - 5% of portfolio grade.

Write a helloworld program using openmp. This program must identify that different threads are printing helloworld. There should be 10 threads.

Then alter the number of threads to 5 by altering the environment variable with a command.

As part of your submission for this you will need a screenshot of the program running ***under your own machine name*** to demonstrate that it is your own work.

If this screenshot is not present, you will lose marks.

Part B – 20% of portfolio grade.

Using a **for** work-sharing construct add the values of two vectors together into a third vector. Write the program with static schedule and then rewrite it with dynamic schedule. What are your conclusions?

As part of your submission for this you will need at least one screenshot of the program running ***under your own machine name***, to demonstrate that it is your own work.

If screenshots of your program running on the server are not present, you will lose marks.

Part C – 25% of portfolio grade.

This MUST be a separate program, not combined with Part B, if you combine it with part B you will lose marks.

Write an OpenMP program that performs matrix calculation. Run your code and collect data based on the following criteria

a) Use 1, 4, 8 and 16 threads on a 50x50 matrix
b) Use 1, 4, 8 and 16 threads on a 500x500 matrix
c) Parallelize only the outer loop for both conditions defined above
d) Parallelize only the inner loop for both conditions defined above

Collect timing data for each case; average the result based on ten test runs each. Make tables and graphs of your average timing data and put them in the submitted report. After you have reported your results, try to explain them as best as possible.

Comments in your code are a good idea, as easy to read code is more likely to raise the amount of marks you are assigned

As part of your submission for this you will need at least one screenshot of the program running ***under your ownmachine name*** on the server, to demonstrate that it is your own work.

If screenshots of your program running on the server are not present, you will lose marks.

Marking Rubric

	Program functionality	Adherence to assignment specification	Code Quality	
First ≥70	Code exceeds expectations and always runs successfully. Demonstration of program running included in submission.	Meets and may exceed all assignment specifications, design matches requirements. Uses OpenMP.	Code quality is excellent, has comments and proper layout. All indenting is correct.	**First ≥70**
Upper Second 60-69	Code is complete and always runs successfully, Demonstration of program running included in submission.	Meets most assignment specifications, design matches requirements. Uses OpenMP.	Code quality is good, has comments and proper layout. Still some indenting errors.	**Upper Second 60-69**
Lower Second 50-59	Code is complete and compiles. May not always run successfully. May not be in C++. Demonstration of program running included in submission.	Meets more assignment specifications, design still not what is required, may not use OpenMP. Code is in separate programs as specified	Code quality is fair, may still lack comments or have improper layout. Indenting should be correct at this level.	**Lower Second 50-59**
Third 40-49	May be complete, but might not compile. May not be in C++.	Meet some assignment specifications but doesn't do so in a satisfactory way. Code is not in separate programs as specified	Code quality is poor, lacks comments, improper layout.	**Third 40-49**
Fail <40	Code is incomplete. May not be in C++.	Fails to meet any assignment specifications.	Quality is poor, or there isn't enough to judge this metric.	**Fail <40**
Late	0	0	0	**Late**

Distributed Programming Portfolio Element

Module Learning Outcomes Met by this Portfolio element:

5: Compare and contrast of the fundamentals of Distributed and Parallel Computing.

Overall Contribution to portfolio grade – 50%.
Broken down into sub elements:
Part A Contribution – 5% of portfolio grade.
Part B Contribution – 25% of portfolio grade.
Part C Contribution – 20% of portfolio grade.

Every aspect must be written in C++ and use MPI.
Parts **A**, **B**, and **C Must** be submitted as **SEPARATE** programs, if they are combined you will lose marks.

Part A – 5% of portfolio grade.

Write a helloworld program using mpi. This program must run with 4 processes.

Comment on the how MPI processes map to processors/cores

As part of your submission for this you will need a screenshot of the program running ***under your own user account*** on the **Cluster.** This is required to demonstrate that it is your own work. If this is not present you will lose marks.

Part B – 25% of portfolio grade.

This MUST be a separate program, not combined with Part A or Part C, if you combine it with either you will lose marks.

a)

Modify the hello.c program so that the slave processes do not issue the print statements, but each sends a message back to the master containing their rank after the master receives each message, it print out a message containing the slave's rank. The program should produce output such as:

Master: Hello slaves give me your messages

Message received from process 1 : Hello back

Message received from process 2 : Hello back

Message received from process 3 : Hello back

Now, all print statements are issued by the master.

Clue: MPI_Comm_rank returns the processes rank using the argument rank..

b)

Alter the code so that each process send a different message back, e.g.:
Master: Hello slaves give me your messages

Message received from process 1 : Hello, I am John

Message received from process 2 : Hello, I am Mary

Message received from process 3 : Hello, I am Susan

Part C – 20% of portfolio grade.

This MUST be a separate program, not combined with Part B. If you combine it with part B you will lose marks.

As part of your submission for this you will need a screenshot of the program running ***under your own machine name*****.** This is required to demonstrate that it is your own work. If this is not present you will lose marks.

In the previous programs, let's say tag of 99 is used in messages. Modify the program from task **b) above** so that the master process sends messages to each slave, but with the tag of 100 and the slave waits for message with a tag of 100. Confirm program works.

Repeat but make the slaves wait for tag 101, and check program hangs. Why?

Marking Rubric

	Program functionality	**Adherence to assignment specification**	**Code Quality**	
First ≥70	Code exceeds expectations and always runs successfully. Distribution aspects excellent. Demonstration of program running included in submission.	Meets and may exceed all assignment specifications, design matches requirements.	Code quality is excellent, has comments and proper layout. All indenting is correct.	**First ≥70**
Upper Second 60-69	Code is satisfactory Distribution aspects fair, may not have all requirements fully implemented, but those that are implemented work. Demonstration of program running included in submission.	Meets most assignment specifications, design matches requirements. MPI used correctly, but incompletely in terms of the brief.	Code quality is good has comments and proper layout. Still some indenting errors.	**Upper Second 60-69**
Lower Second 50-59	Code is complete and compiles. May not always run successfully. Distribution elements poor. May not be in C++. Demonstration of program running included in submission.	Meets more assignment specifications, design still not what is required. MPI used, but incorrectly. Code is in separate programs as specified	Code quality is fair, may still lack comments or have improper layout. Indenting should be correct at this level.	**Lower Second 50-59**
Third 40-49	May be complete, but might not compile. May not be in C++.	Meet some assignment specifications but doesn't do so in a satisfactory way. MPI not used. Code is not in separate programs as specified	Code quality is poor, lacks comments, improper layout.	**Third 40-49**
Fail <40	Code is incomplete. May not be in C++.	Fails to meet any assignment specifications.	Quality is poor, or there isn't enough to judge this metric.	**Fail <40**
Late	0	0	0	**Late**

Submission content requirements for the entire portfolio

All elements, including required files (as specified in each portfolio element), should be placed in folders indicating which aspect of the portfolio they relate to, then these should be placed in a folder named portfolio which is then compressed to create a single file that you will then submit.