



Xi'an Jiaotong-Liverpool University

西交利物浦大學

XJTLU Entrepreneur College (Taicang) Cover Sheet

Module code and Title	DTS202TC Foundation of Parallel Computing
School Title	School of AI and Advanced Computing
Assignment Title	Individual Assessment 2
Submission Deadline	Saturday Dec. 23rd, 2023 @ 11:59pm
Final Word Count	N/A
If you agree to let the university use your work anonymously for teaching and learning purposes, please type "yes" here.	

I certify that I have read and understood the University's Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy I certify that:

- My work does not contain any instances of plagiarism and/or collusion.
- My work does not contain any fabricated data.

By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.

Scoring – For Tutor Use					
Student ID					
Stage of Marking	Marker Code	Learning Outcomes Achieved (F/P/M/D) (please modify as appropriate)			Final Score
		A	B	C	
1 st Marker – red pen					
Moderation – green pen	IM Initials	The original mark has been accepted by the moderator (please circle as appropriate):			Y / N
		Data entry and score calculation have been checked by another tutor (please circle):			Y
2 nd Marker if needed – green pen					
For Academic Office Use		Possible Academic Infringement (please tick as appropriate)			
Date Received	Days late	Late Penalty	<input type="checkbox"/> Category A <input type="checkbox"/> Category B <input type="checkbox"/> Category C <input type="checkbox"/> Category D <input type="checkbox"/> Category E		
			Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary) _____		

DTS202TC Foundation of Parallel Computing

Individual Assignment 2

Due: Saturday Dec. 23rd, 2023 @ 11:59pm

Weight: 70%

Maximum score: 100 points

Assessment Overview

The purpose of this assignment is to gain experience parallel implementation and analysis. You are expected to implement two C programs to rotate a grayscale image using two different parallelism techniques.

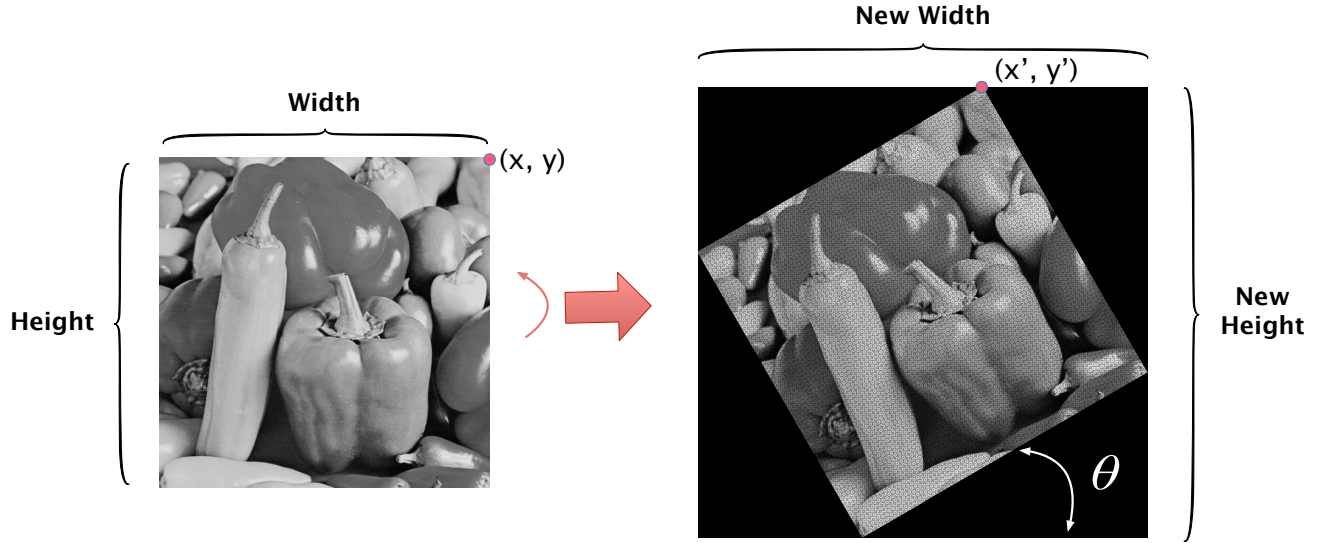
Learning Outcomes

- C. Devise and implement parallel algorithms
- D. Acquire basic software development skill using MPI
- E. Analyze and implement common parallel algorithm patterns in a parallel programming model such as CUDA.
- F. Design experiments to analyze the performance bottlenecks in their parallel code.
- G. Apply common parallel techniques to improve performance given hardware constraints.
- H. Use a parallel debugger to identify and repair code defects; Use a parallel profiler to identify performance bottlenecks in their code.
- J. Demonstrate understanding of the major types of hardware limitations that limit parallel program performance.

Avoid Plagiarism

- Do **not** submit work from others.
- Do **not** share code or work to other students.
- Do **not** read code or work from others, discussions between teams should be limited to high level only.
- Do **not** use open-source code.

Tasks



Algorithm 1

```

1:  $new\_height \leftarrow |height * cosine(\theta)| + |width * sine(\theta)|$ 
2:  $new\_width \leftarrow |width * cosine(\theta)| + |height * sine(\theta)|$      $\triangleright$  Define the height and width of the
   new image rotated
3:
4:  $ori\_centre_w \leftarrow width/2$ 
5:  $ori\_centre_h \leftarrow height/2$ 
6:  $new\_centre_w \leftarrow new\_width/2$ 
7:  $new\_centre_h \leftarrow new\_height/2$ 
8: for  $i = 0, 1, 2, \dots, height - 1$  do
9:   for  $j = 1, 2, \dots, width - 1$  do
10:     $x \leftarrow width - ori\_centre_w - j$ 
11:     $y \leftarrow height - ori\_centre_h - i$   $\triangleright$  coordinates of pixel with respect to the centre of original
       image
12:
13:     $new\_x \leftarrow x * cosine(\theta) + y * sine(\theta)$ 
14:     $new\_y \leftarrow y * cosine(\theta) - x * sine(\theta)$ 
15:     $new\_x \leftarrow new\_centre_w - new\_x - 1$ 
16:     $new\_y \leftarrow new\_centre_h - new\_y - 1$   $\triangleright$  coordinates of pixel with respect to the centre of
       new image
17:     $img\_out[new\_y][new\_x] \leftarrow img\_in[i][j]$   $\triangleright$  make sure new_x and new_y are integers, and
       don't forget to check the boundaries
18:   end for
19: end for

```

0.1 Image Rotation

Image rotation is one of the most common image processing algorithms. To complete the rotation we need to write the pixel value for each pixel in the new location. Using some High School geometry

we can work out the relationship between the source coordinates (x, y) and the destination coordinates (x', y')

$$\begin{aligned}x' &= x \cos \theta + y \sin \theta \\y' &= -x \sin \theta + y \cos \theta\end{aligned}\tag{1}$$

A complete pseudocode has been provide in Algorithm 1.

You will see some dots after this simple rotation algorithm, don't worry about that for this coursework.

0.2 Reading and Writing Image

You will need to read a PGM image, process the rotation, then write the rotated PGM image back to the file system. **Plain PGM** format is a simple grayscale graphic image format, each pixel is represented by its grey value number, with 0 being black and Maxval (defined in the PGM file) being white. The bellow image.pgm is given as an example, more detail pgm specifications can be found at <http://davis.lbl.gov/Manuals/NETPBM/doc/pgm.html>

image.ppm

```
P2
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 0 11 0 0 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 0 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

1 Parallel implementations (25 points for each implementation)

Each student should implement two difference parallel programs, using MPI and one other parallel technique (Pthread, OpenMP and CUDA). Note that you only need to speedup the core algorithm.

2 Performance analysis (30 points)

Analyse the performances of two parallel implementations, including speedup and efficiencies (15 points).

You should provide line plots to represent the results, including serial runtime, runtime of different processes, speedup and efficiencies. (5 points).

Compare and contrast the two implementations (Performance Metrics, Hardware limitation, Ease of Debugging and Testing, Communication and Synchronization, or any other aspects you can think of), choose your preferred parallel technique and justify your choice (10 points).

3 Reflection (10 points)

Did you face any challenges during the implementation? How did you solve them?

4 Submission

You should submit the following files in a zip:

- ***rotate_mpi/pthread/openmp.c*** Your parallel implementations.
- A ***Makefile*** that will compile your code, make sure the output executable names are correct.
- A ***pdf*** file contains all the source code and the report, you should put the ***Cover Page*** in the first page of the report.

Once you have all the files, please put them in a single directory (named A2) and compress it to a ***zip*** file. You must follow the following structure:

```
|— A2.zip
|   |— report.pdf
|   |— code
|   |   |— mpi
|   |   |   |— rotate_mpi.c
|   |   |   |— Makefile
|   |   |— openmp
|   |   |   |— rotate_openmp.c
|   |   |   |— Makefile
|   |   |— pthread
|   |   |   |— rotate_pthread.c
|   |   |   |— Makefile
|   |   |— cuda
|   |   |   |— rotate_cuda.cu
|   |   |   |— Makefile
```

The assignment must be submitted via Learning Mall to the correct drop box. Only electronic submission is accepted and no hard copy submission. All students must download their file and check that it is viewable after submission. Documents may become corrupted during the uploading process (e.g. due to slow internet connections). However, students themselves are responsible for submitting a functional and correct file for assessments.

Please note that quality of report and correctness of submission will also be marked (**10 points**, 5 points for the quality of report, 5 points for the correctness of submission).

Category	Exemplary (8-10)	Accomplished (6-7)	Capable (4-5)	Need Improvement (0-3)
Functionality (Correctness and output)	Code produces correct output. Demonstrate effective use of programming features as well as functions and libraries. Code is optimized for the maximum performance in parallel. The implementation follows best practices	Code produces partially correct output. Partially missing code or minor issues with excellent logic. Performance is not optimal.	Code doesn't compile. However, design idea is clear. Shows some understanding of the tasks and uses relevant programming features	Code doesn't compile. Design idea is not obvious. Doesn't know much understanding of the task.
Quality (Structure and best practice)	Excellent code quality with correct use of function and programming logic. Excellent design and the codes are commented, or design idea is clearly explained.	Code compiles. Result is faulty. Design and structure are not intuitive. Inappropriate use of function or algorithms.	Code doesn't compile. Incomplete code. However, design and structure are intuitive.	Code doesn't compile. Incomplete code. Design and structure are incorrect.
Justifications	Demonstrated correct and comprehensive justifications and analysis.	Correct justifications and analysis of algorithm with minor defects.	Limited justifications with critical errors.	Limited attempt or no justifications
Performance analysis	Provide a comprehensive analysis of performance, including detailed calculation of speedup and efficacies. Provide pleasing visualizations of results. Justifications of preferred parallel methods aligned with the actual implementations.	Provide a moderate performance analysis with mostly correct calculations. Sufficient justifications with minor mistakes.	Incomplete analysis of performance, lack of support justifications or contains major defects.	Limited attempt without any justifications.
Reflection	Reflection is thoughtful, carefully written, and demonstrates significant depth of self-reflection on the topic.	Reflection is carefully written and generally demonstrates considerable depth of self-reflection on the topic.	Reflection shows lapses in care and depth of self-reflection regarding the task at hand.	Limited attempt without any justifications.