UNIVERSITI SAINS MALAYSIA

CST435: Parallel and Cloud Computing

Assignment 2

Group Members:

1. Name: _____ Matric No. : _____

2. Name: _____ Matric No. : _____

3. Name: _____ Matric No. : _____

Assignment Objectives

Upon completion of this assignment, students will be able to:

1. Design and implement parallel programs using multiple parallel computing paradigms
2. Deploy and execute parallel applications on Google Cloud Platform (GCP)
3. Analyze performance characteristics of parallel algorithms through speedup and efficiency metrics

Assignment Overview

In this assignment, you will implement a parallel image processing system that applies various filters to a collection of images from the Food-101 dataset. You will implement the same image processing pipeline using different parallel programming paradigms and deploy your implementations on Google Cloud Platform (GCP). This assignment will help you understand the trade-offs between different parallelization approaches and gain hands-on experience with cloud computing.

You will use the Food-101 dataset for this assignment:
https://www.kaggle.com/datasets/dansbecker/food-101
You do not need to use all images. Create manageable subsets for testing.

Image Processing Operations

Your system must implement the following image filters:

1. Grayscale Conversion - Convert RGB images to grayscale using luminance formula
2. Gaussian Blur - Apply 3×3 Gaussian kernel for smoothing
3. Edge Detection - Sobel filter to detect edges
4. Image Sharpening - Enhance edges and details
5. Brightness Adjustment - Increase or decrease image brightness

All filters operate on individual pixels or small neighborhoods, making them suitable for parallel processing.

Implementation Tasks

You must implement the image processing pipeline using two different parallel paradigms. You may choose to implement in either C++ or Python:

Option 1: C++ Implementation

Implement using both of the following:

- C++ threads (std::thread)
- OpenMP

Option 2: Python Implementation

Implement using both of the following:

- multiprocessing module
- concurrent.futures

Deliverables

You must submit a technical report containing the following:

1. Implementation
   - Code structure and organization explanation
2. Performance Analysis
   - Speedup and efficiency metrics for different thread/process counts
   - Comparison tables showing execution times for both paradigms
   - Analysis of scalability and bottlenecks
3. GitHub/GitLab Repository Link
   - Link to your source code repository
   - Your repository must include:
     - All source code files with proper comments
     - Comprehensive README file with project description
4. YouTube Video Link
   - Link to your demonstration video (public or unlisted)
   - The video must demonstrate:
     - Live demonstration of both parallel implementations running on GCP
     - Performance comparison and analysis
     - Discussion of results and insights gained

Submission Instructions

- This is a group submission – submit ONE technical report per group (four in a group group)
- Upload the technical report to elearn@USM
- Marks will be deducted if:
  - Repository link and/or YouTube video link are not included in the report
  - Links are broken, incorrect, or inaccessible
- Verify that your links are correct and accessible before submission
- Email submissions of links will **NOT** be accepted – all links must be included in the PDF report

Assignment Due Date: 11th January 2026, 11.59pm

All assignments MUST be submitted before/on the given date. Late submissions will NOT be entertained.

Plagiarism/pirating and copying are serious academic offences. Students that are found to plagiarize or copy will get an F for the assignment/report or for the whole coursework grade.

Thank you

Grading Rubrics:

Criteria	Weight	Excellent (70-100%)	Sufficient (52-69%)	Fair (36-51%)	Poor (0-35%)
Parallel Implementation Design	40%	Both parallel paradigms implemented with excellent efficiency and proper synchronization mechanisms. All five image filters correctly implemented with appropriate parallelization strategy. Well-structured, clean code demonstrating deep understanding of chosen paradigm. Excellent load balancing and resource utilization.	Both paradigms functionally implemented with adequate performance. All filters working correctly. Clear code structure with proper synchronization. Shows good understanding of parallel programming fundamentals.	Basic parallel implementation with limited optimization. Most filters working but may have minor issues. Simple code structure with some synchronization problems. Demonstrates basic understanding of parallelization.	Incomplete or non-functional parallel implementation. Filters not working properly or missing. Poor code organization. Missing or incorrect synchronization. Shows minimal understanding of parallel programming.
Performance Analysis and Comparison	30%	Comprehensive speedup and efficiency metrics tested across multiple thread counts (2, 4, etc). Professional comparison tables and graphs (bar charts, line graphs) clearly showing performance differences. Insightful analysis of scalability, bottlenecks, and trade-offs between both paradigms.	Adequate performance metrics with testing on multiple thread counts. Clear comparison tables and basic graphs showing execution times. Good analysis identifying main performance characteristics and basic bottlenecks for both paradigms.	Limited performance testing with few thread counts. Basic tables or simple graphs. Minimal analysis with limited discussion of performance differences or scalability issues.	Incomplete or missing performance metrics. No proper comparison between paradigms. Poor or absent visualization. No meaningful analysis of results.

		Discussion includes Amdahl's Law and practical implications.			
Code Documentation and Repository	20%	Excellent code organization with comprehensive comments explaining parallelization decisions. Complete README with detailed project description. Clean repository structure. All links working and accessible.	Good code organization with adequate comments. Clear README with project description. Organized repository structure. Links included and functional.	Basic code organization with minimal comments. Simple README with limited information. Basic repository structure. Some documentation may be unclear or incomplete.	Poor code organization with few or no comments. Missing or incomplete README. Disorganized repository. Missing, broken, or inaccessible links.
Video Presentation	10%	Clear, professional demonstration showing both implementations running on GCP with live performance comparison. Comprehensive explanation of results with insightful discussion of findings, trade-offs, and lessons learned. Well-paced presentation within appropriate time.	Clear demonstration of both implementations on GCP with adequate performance comparison. Good explanation of results and basic insights. Reasonable pacing within time limit.	Basic demonstration showing implementations with simple comparison. Limited explanation of results. May lack clarity or exceed time limit.	Unclear or incomplete demonstration. Missing performance comparison or explanation. Poor presentation quality. Video link not provided or inaccessible.