COLLEGE OF ARTS AND SCIENCES
UNIVERSITY OF THE PHILIPPINES LOS BAÑOS
1/F Wing C Physical Sciences Bldg., Harold Cuzner Royal Palm Ave.
College 4031, Laguna, Philippines
Phone +63 49 536 2313
ics.uplb@up.edu.ph www.ics.uplb.edu.ph
INSTITUTE OF COMPUTER SCIENCE

CMSC 180: Introduction to Parallel Computing
Second Semester 2022-2023

Laboratory Exercise 03
Core-affine Threaded Computation of Interpolating Elevation

Introduction

Rewrite the program that you wrote in Exercise 02 so that the threads that you created must be running specifically unto one specific core only. If your computing machine has 4 or 8 threads, assign your threads to three or seven cores respectively.

Exercise Specifications

Research Activity 1:

A. Write the main program lab03 that includes the following:

1. Read *n* and *t* as user input (maybe from a command line or as a data stream), where *n* is the size of the square matrix, *t* is the number of threads to create, and *n > t;*
2. Create a zero *n x n* square matrix **M.** Assigned a randomize non-zero value to grid points divisible by 10 such (0,0), (0,10), (10,0), (20,0),(10,10) …… You can use a function for this but the running time of this will not be considered in the *time_elapsed*
3. Divide your M into *t* submatrices, m_1,m_2,...., m_t; *You can add an additional filter if the matrix size **n** is not divisible **t** such as if n=10 while t =3, then the input values cannot be processed because there is excess column or row.*
4. Take note of the system *time_before* ;
5. Create t threads, to interpolate the values for each submatrix. **(IMPORTANT)**
6. Assign the thread to a core. **(VERY IMPORTANT, Additional step in this exercise)**
7. Take note of the system time *time_after*;
8. Obtain the elapsed $time\_elapsed = time\_after - time\_before$;
9. Output the *time_elapsed*
10. (Optional) You can output the resulting matrix.

B. Fill in the following table with your time reading:

n	t	Time Elapsed (seconds)			Average Runtime (seconds)
		Run 1	Run 2	Run 3	
8,000	1				
8,000	2				
8,000	4				
8,000	8				

8,000	16				
8,000	32				
8,000	64				

Research Question 1: What is the difference of the average time that you obtained for t = 1 compared to the average that was obtained in Exercises 01 and 02?

Research Activity 2: Repeat research activity 1 for *n = 16,000* and *n = 20,000*. Do you think you can achieve *n=50,000* and even *n = 100,000*? **Try to see if you can**. If you were able to do so, why do you think you can now do it? If not yet, why do you still can not?

C. Using a graphing software for each *n*, graph *t* versus **Average** obtained from the Table above. Describe in detail what you have observed. Do you think you can go as far as *t = n*? If not, what about *t = n/2* ? Or, *t= n/4* ? Or, *t = n/8* ?

Research Question 2: Did your average time improve compared to Exercise 01 and Exercise 02 for every t? Which *t* is the average time better? Statistically the same? Slower? Why?

Lab Report Guidelines

Submit a report on your answers to the research questions posted in this exercise. All laboratory reports and term projects must be written in a technical way. That means each must have the following sections:

1. Introduction,
2. Objectives,
3. Methodology,
4. Results and Discussion,
5. Conclusion,
6. List of Collaborators, *(Yes, you can collaborate with other students but make sure that you can explain your work)*
7. References, and
8. Appendices.

You will include in the appendices the respective fully commented source codes of your programs. Submit your report through the Google Classroom Laboratory Exercise 03 portal.

Notes:

- You may set the core affinity of a POSIX thread(pthreads in C/C++) using the function ***pthread_setaffinity_np*** .
- You may use the ***multiprocessing library*** in Python to automatically assign/allocate CPUs to your tasks.