CHED National Center of Excellence in Information Technology Education
INSTITUTE OF COMPUTER SCIENCE
College of Arts and Sciences
University of the Philippines Los Baños
College 4031, Laguna, PHILIPPINES
☎ Phone (63-49) 536-2313 ✆ Fax (63-49) 536-2302 ✆ Campus VoIP 6123
✉ E-mail jppabico@up.edu.ph 💻 Web www.ics.uplb.edu.ph/jppabico

CMSC 180
Introduction to Parallel Computing
Second Semester AY 2023-2024

Laboratory Research Problem 02
Runtime-efficient Threaded Pearson Correlation Coefficient

Introduction

Given an $m \times n$ matrix **X** with m rows and n columns and an $m \times 1$ vector $\boldsymbol{y}$, a $1 \times n$ vector $\boldsymbol{r}$ holds the Pearson Correlation Coefficient of the columns in **X** and **y** shown in Equation 1 of Laboratory Research Problem 01 (LRP01).

Research Question 1: What do you think is the complexity of solving the Pearson Correlation Coefficient of the columns in an $n \times n$ square matrix **X** and an $n \times 1$ vector **y** when using n concurrent processors? The obvious processor assignment is one column of **M** and the vector **y** for each processor.

Research Question 2: What do you think is the complexity of solving the Pearson Correlation Coefficient of the columns in an $n \times n$ square matrix **X** and an $n \times 1$ vector **y** when using $n/2$ concurrent processors (what is the obvious processor assignment here)? What about with $n/4$ concurrent processors (i.e., processor assignment)? What about with $n/8$ concurrent processors? What about with n/m concurrent processors, where $n >> m$? Is the processor assignment still obvious at n/m concurrent processors?

Research Activity 1: Extend the serially efficient computer program that you wrote in LRP01 to use threads to compute the Pearson Correlation Coefficient of the columns in an $n \times n$ square matrix **X** and an $n \times 1$ vector **y**. In other words, transform your efficient serial computer program into a (hopefully more efficient and faster) threaded computer program.

<u>How to do it?</u>

1. Write the main program `lab02` that includes the following:
   (1) Read n and t as user inputs (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 non-zero $n \times n$ square matrix **X** whose elements are assigned with random non-zero integers;
   (3) Create a $1 \times n$ vector **y** whose elements are also non-zeroes;
   (4) Divide your **X** into t submatrices of size $n/t \times n$ each, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t$;
   (5) Take note of the system time `time_before`;
   (6) Create t threads, where for the ith thread call `pearson_cor`($\boldsymbol{x}_i$, n, n/t);← very important
   (7) Recreate the correct $\boldsymbol{r}$ from the output of each threaded `pearson_cor()`;← very important
   (8) Take note of the system time `time_after`;
   (9) Obtain the elapsed time `time_elapsed:=time_after - time_before`;
   (10) Output `time_elapsed`;

2. Fill in the following table with your time readings:

n	*t*	**Time Elapsed (seconds)**			**Average Runtime (seconds)**
		Run 1	**Run 2**	**Run 3**	
25,000	1*				
25,000	2				
25,000	4				
25,000	8				
25,000	16				
25,000	32				
25,000	64				

*This should be closed but a little bit higher to the average that you obtained in LRP01.

Research Question 3: Why do you think that $t = 1$ will be a little bit higher than the average that was obtained in LRP01?

Research Question 4: In step (4) in number 1 above, explain what will happen if we divide **X** into $n \times n/t$ instead? How are we going to do it so that the same answer can be arrived at?

Research Activity 2: With `lab02`, repeat the activities in LRP01 for $n = 30{,}000$ and $n = 40{,}000$. Do you think you can now achieve $n = 50{,}000$ and even $n = 100{,}000$? Try it 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 think you still can not?

3. 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 Activity 3: Repeat research activity 1 but for the division of **X** as described in Research Question 4. What sort of thing do you need to do so that this division can be implemented? Graph the average as in number 3 above, if obtained.

<u>Solutions to Possible Problems that may Come up</u>

Problem 1: *I do not know how to write threaded programs*.
Answer 1: *Is that really a problem?*

Problem 2: *The programming language I used for LRP01 does not have a robust library for writing threaded codes.*
Answer 2: *See Answer 1.*

Problem 3: *My* [*boy*|*girl*]*friend broke up with me today and I can not really concentrate on this exercise.*
Answer 3: *See Answer 1.*

Problem 4: *I have not eaten any meal since last Friday.*
Answer 4: *Pretend you are a freshman and attend any org's orientation.*