Jerico Sabile

CMSC 180
Introduction to Parallel Computing
Laboratory Research Problem 02
Runtime-efficient Threaded Pearson Correlation Coefficient

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

> **The time complexity of solving the Pearson Correlation Coefficient of the columns in an n × n square matrix X and an n × 1 vector y when using n concurrent processors is O(n) because n jobs which have O(n) time complexity are split into n workers.**

Research Question 2: What do you think is the complexity of solving the Pearson Correlation Coefficient of the columns in an n × n square matrix X and an n × 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?

> **When using n/2 concurrent processors, the time complexity is still O(n) because it was simplified from O(2n). This also applies to n/4, and n/8 concurrent processors because O(4n) and O(8n) will be simplified to O(n). However, when n >> m, the real world time complexity will be worse than O(n) even though its theoretical time complexity is O(n). Also the processor assignment would not be obvious because n is much larger than m.**

n	t	Time Elapsed (seconds)			Average Runtime (seconds)
		Run 1	Run 2	Run 3	
25,000	1*	32.4982	32.6428	32.4009	32.5140
25,000	2	16.7015	16.6241	16.7421	16.6892
25,000	4	13.2852	13.4428	13.6376	13.4552
25,000	8	14.3579	14.3926	14.4362	14.3976
25,000	16	15.8943	15.9002	15.7846	15.8597
25,000	32	17.8518	17.7462	17.7822	17.7934
25,000	64	17.8313	18.0894	18.0490	17.9899

Jerico Sabile

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

> **t = 1 will be a bit higher than the average obtained from LRP01 because there is an overhead for using the creation and destruction of the thread.**

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

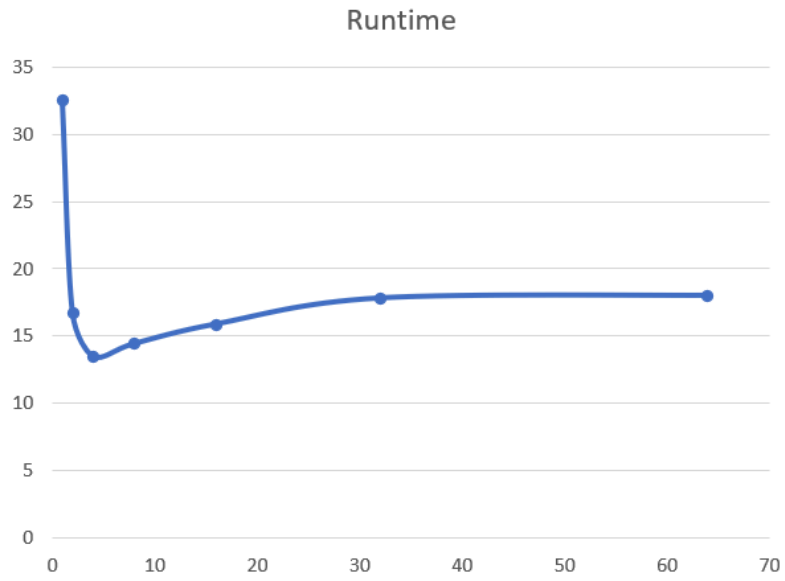
> **In this case, every thread that has n/t × n, calculates its summations in every column and returns these values to the main function. After all threads finish all of their computations, the main function adds all these summations for every column to compute the Pearson correlation coefficient.**

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?

> **It was able to achieve up to n = 50,000. However it was not able to run when n > 50,000, because 16GB RAM is not enough for the memory allocation for the matrix. Although the computation is faster than in the LRP01 since I used 4 threads, the memory is still not enough for the computation for n > 50,000**.

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?

> **I have observed that t = 4 has the fastest runtime. This is because the machine has 4 core cpu. Also I cannot go as far as t = n, because this would be inefficient and would have a lot of overhead especially when n is large. Also, t = n/2, t = n/4, and t=n/8 would be inefficient if n is large. The optimal number of threads to use would be less than or equal the number of cores in the processor.**



Jerico Sabile

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.

n	t	Time Elapsed (seconds)			Average Runtime (seconds)
		Run 1	Run 2	Run 3	
25,000	1*	17.6138	17.5709	17.6769	17.6205
25,000	2	9.0373	9.0347	9.1187	9.0635
25,000	4	7.3891	7.2962	7.2753	7.3202
25,000	8	7.1958	7.1925	7.1911	7.1931
25,000	16	6.0761	6.1081	6.1095	6.0979
25,000	32	5.4398	5.5353	5.4822	5.4858
25,000	64	5.2116	5.2653	5.2049	5.2273