



**CMSC 180**

**Introduction to Parallel Computing**

**Second Semester AY 2024-2025**

**Laboratory Research Problem 05**

**Distributed Computation of the Min-Max Transformation of  
Elements in a Column of a Matrix**

**Introduction**

The answer to question 5 of Laboratory Research Problem 04 should have been a one-to-many personalized broadcast (1MPB) of the matrix  $X$ . If you have implemented 1MPB in your `lab04`, it is expected that you will observe a significant improvement on average runtime as  $n$  and  $t$  increases. The 1MPB distributed parts of the matrix  $X$  to different slave processes which can be processed concurrently by different machines. But the respective Min-Max Transformations (MMTs) of the matrix  $X$ 's columns must be computed by the slave processes. The respective MMTs should be compiled in a matrix  $T$  (recall Equation 1 in Laboratory Research Problem 01), the elements of which would have been computed independently by the slave processes and which should be sent back to the master in a many-to-one personalized reduction (M1PR) manner to be rebuilt into one big matrix by the master process.

**Research Activity 1: How to do it?**

1. Extend your `lab04` into `lab05` which...
  - (1) should compute the Min-Max Transformations into a matrix  $T$  of the respective columns of the received parts of matrix  $X$ ;
  - (2) should communicate back to the master the respective parts of matrix  $T$  in a M1PR manner.
2. As in `lab04`, the master process must take note of the `time_before` before distributing the matrix  $X$  to the different slave processes and then take note of the `time_after` after the master has rebuilt the full matrix  $T$ . The master process must report the `time_elapsed`.
3. Each of the slave processes must take note of the `time_elapsed` ONLY in computing the Min-Max Transformations. Strictly, this means that a slave process must take note of the `time_before` after receiving the parts of matrix  $X$  and before computing for parts of the matrix  $T$  and then take note of the `time_after` after the parts of matrix  $T$  has been fully computed and before the part of matrix  $T$  is sent back to the master. The `time_elapsed` is the time spent by each slave process in actual computation. The slave must report the `time_elapsed`.
4. Conduct this activity using different machines, where the slave processes in each machine are running in a core-affine manner.

5. Fill in the **Table 1** below with your time readings from the master process only.

<b>Table 1.</b> Time elapsed as reported by the master process.					
$n$	$t$	Time Elapsed (seconds)			Average Runtime (seconds)
		Run 1	Run 2	Run 3	
20,000	2				
20,000	4				
20,000	8				
20,000	16				
25,000	2				
25,000	4				
25,000	8				
25,000	16				
30,000	2				
30,000	4				
30,000	8				
30,000	16				

6. Fill in **Table 2** below with your time readings from the slaves. Fill in only the maximum for each run.

<b>Table 2.</b> Time elapsed as reported by the slave processes.					
$n$	$t$	Time Elapsed (seconds)			Average Runtime (seconds)
		Max of Run 1	Max of Run 2	Max of Run 3	
20,000	2				
20,000	4				
20,000	8				
20,000	16				
25,000	2				
25,000	4				
25,000	8				
25,000	16				
30,000	2				
30,000	4				
30,000	8				
30,000	16				

## Research Activity 2: Communication and Computation

The average runtime that you obtained in **Table 1** above includes the computational cost (in terms of time) of both communication and computation of the whole distributed computation. Using any software (a spreadsheet will do), create a 3D plot of  $n$ ,  $t$  and average runtime, where the average runtime is in the vertical axis. Name the plot as **Figure 1** in your report.

Alternatively, if you can not create a 3D plot, create a 2D graph of  $t$  and average runtime, where  $t$  is plotted as the horizontal axis and the average runtime is the vertical axis. You will be plotting three plots on the same 2D graph where each plot corresponds to each of the  $n$ .

Discuss the pattern that you observed for increasing  $n$  and  $t$ .

## Research Activity 3: Computation Only

The average runtime that you compiled in **Table 2** above presents only the mean of the maximum computation cost spent by the slave processes. Why do we need to report the maximum of the `elapsed_time` for each run? Why not the average of each run?

As in Research Activity 2, plot  $n$ ,  $t$  and average runtime. Name the plot as **Figure 2** in your report.

Discuss the pattern that you observed for increasing  $n$  and  $t$ .

## Research Activity 4: Performance Metrics

For each problem size  $n$ , Compare the plots in Figures 1 and 2. You must be able to observe that the plots in Figure 1 has a higher runtime than the plots in Figure 2. Obviously because the runtime in Figure 1 includes both the communication and computation time while the runtime in Figure 2 only shows the computation time. But, how much did your `lab05` spent in communication over computation? Aside from communication and computation, do you think there are overhead that was spent by your `lab05`? If so, what are those (i.e., if there are idling, where are those in your code, if there are excess computation, why was the excess included)?

Using your result in Laboratory Research Problem 01 as serial runtime  $T_s$  (should be constant for each  $n$ ), the average runtime you obtained in Table 1 as the parallel runtime  $T_p$ , and the number of processors  $p$  as  $t$ , report as **Table 3** (see next page) the parallel overhead  $T_o$ , the parallel speedup  $S$ , the parallel efficiency  $E$ , and the parallel cost  $pT_p$  for increasing  $n$  and  $t$ .

Separately discuss the patterns of performance metrics that you obtained for each  $n$  and  $t$ . Did you observe superlinearity? Is your implementation cost-optimal?

### Reaction to possible problem in the activity:

Problem 1: *I was not able to implement in `lab04` and also here in `lab05` the IMPB and MIPR.*

Reaction 1: Include in your report that you were not able to but articulate in the report what would have been different had these broadcast schemes been implemented correctly.

**Table 3.** Performance metrics of the parallel, distributed Pearson Correlation Coefficient.

$n$	$t$	Serial $T_s$	Parallel			
			$T_o$	$S$	$E$	$pT_p$
20,000	2					
20,000	4					
20,000	8					
20,000	16					
25,000	2					
25,000	4					
25,000	8					
25,000	16					
30,000	2					
30,000	4					
30,000	8					
30,000	16					