ELEN 4020 Lab 1

Uyanda Mphunga - 1168101 — Darren Blanckensee - 1147279 — Ashraf Omar - 710435 — Amprayil Joel Oommen - 843463

I. Introduction

The purpose of this lab was to expose the students to the concept of parallelism along with introducing the students to the openmp and pthread libraries so as to become familiar with when and how to use parallelism. The task was to create code that uses parallelism to transpose a matrix without using any additional space using first openmp and then pthreading. The code was to be run of several sized matrices and using different numbers of threads. The different matrix sizes are 128x128, 1024x1024 and 8192x8192. The number of threads to be used are 4, 8, 16, 64 and 128. These are to be timed and compared so as to understand the behaviour of parallelism when applied to computing a matrix transpose.

II. Open MP

A. Initial Approach and Problems

Initially the matrix had been declared as a one dimensional array with elements being assigned using row-major ordering. The transpose was done in a nested for loop with the outer loop going from 0 to the number of rows - 1 and the inner loop going from whatever the outer loop is + 1 to the number of columns. This is not ideal for open mp's parallel for loop decomposition. It would be ideal to fuse the nested for loop into a single for loop which open mp would then parallelise. This was not possible as the students could not find a way to define both nested for loop variables when there is just one loop. This is easy to do when both loop variables start from 0 and go up to a given number but becomes difficult when loop variables are defined in terms of each other which was the case here. Therefore it was decided that it would be left as a nested for loop and that the array would be a 2D array.

The collapse(2) command for collapsing nested for loops could not be used for the same reason that the loops could not be fused and therefore only one of the two loops in the nested for loop could be parallelised. It was determined that the best loop to parallelise was the outside loop (trial and error determined that this gave the best results). Dynamic scheduling was used with a chunk size of 1 such that each thread would work on the next available piece of data. The code was run using the various numbers of threads and on the different sized matrices and the times will be compared in the comparison section.

III. Pthreads

IV. Comparison

TABLE I: Time Comparison of OpenMP and Pthread

		Times		
		128	1024	8192
Pthread	No. of Threads			
	4			
	8			
	16			
	64			
	128			
OpenMP	4			
	8			
	16			
	64			
	128			