CSC3210 Assignment 3 Task 3B – Tony Ngo

2.2 – parallelLoopEqualChunks.c

A screenshot of a cell phone

Description automatically generated

This is the code for parallelLoopEqualChunks.c shown in the nano text editor.

A screenshot of text

Description automatically generated

This is my creating and creating the executable of the parallelLoopEqualChunks.c program. When I originally compiled it, I forgot to remove the numbers beforehand from the copy and paste which did not allow me to compile it, creating those errors. Once I removed it I was able to create the executable using “gcc parallelLoopEqualChunks.c -o pLoop -fopenmp”



This is the execution of pLoop with multiple threads. The threads themselves did not run in order, but they were executed in the correct iteration. This is because the threads will perform in the order that the CPU believes is most efficient.

2.2 – parallelLoopChunksOf1.c

A screenshot of a cell phone

Description automatically generated

This is the code for parallelLoopChunksOf1.c shown in the nano text editor.

A screenshot of a cell phone

Description automatically generated

This is the creation of the parallelLoopChunksOf1.c file. I created the “pLoop2” executable using “gcc parallelLoopChunksOf1.c -o pLoop2 -fopenmp”.

A picture containing text

Description automatically generated

This is the execution of pLoop2 with different values. The way that splitting it into chunks works is that the CPU will use each thread as needed for each operation, which is seen in this code. Because for each iteration, a different thread is being used in order (i.e. with four cores being used, Thread 0 performed iteration 0 -> Thread 1 performed iteration 1 -> Thread 2 performed iteration 2 -> Thread 3 performed iteration 3 …).

A picture containing bottle

Description automatically generated

This is the pLoop2 output with the uncommented variation. The output is the same for both.

4.2 - reduction

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Here is the code for reduction.c shown in the nano text editor

A screenshot of a cell phone

Description automatically generated

This is the creation of the parallelLoopChunksOf1.c file. I created the “reduction” executable using “gcc reduction.c -o reduction -fopenmp”.

4.3

A screenshot of a cell phone

Description automatically generated

This is the code ran with the first // comment removed at line 39. The parallel sum is NOT the same as sequential sum. I noticed that the number would be bigger with the less amount of CPU cores that were used in the calculation.

A picture containing text

Description automatically generated

This is the output ran with the second // comment removed also at line 39. The parallel sum is the same as the sequential sum in this case. I believe that this is because the initialization of the pragma openMP needs to be utilized or multithreading would not be enabled, which was seen in the first execution with just the first // comment removed.