Zachary Humphries

COMP 605

24 March 2023

Assignment 3

1. Count Sort Question
   1. If one tries to parallelize the outer loop, which variables should be private and which shared?

If someone were to parallelize the outer loop, *int i*, *int j*,and *int count* would be private and *int a[]*, *int n*, and *int\* temp* would be shared. Each thread accessing and modifying both *int i* and *int j* would cause issues with the loops as when one thread changes *int i* and/or *int j*, the other threads might skip over steps. To parallelize the outer loop, *int i* would need to be split up. Each thread modifying *int count* would give inaccurate indexes for the sorted array.

The variables *int a[]* and *int n* are shared because all threads would need access to the information in these variables. Each thread would need to modify *int\* temp* to input their sorted portion of the array.

* 1. Are there any loop-carried dependencies in the previous parallelization? Explain your answer.

Assuming that *int i*, *int j*,and *int count* are private, there are no loop-carried dependencies. As the array *int a[]* is split up by changing the outer loop to something like *for (int i = my\_i; i < (my\_i + n/p); i++){}*, since the inner loop goes through the entire array *int a[]* and the resulting values are stored in a temporary array, there are no values that are being read or overwritten by other threads.

* 1. Is it possible to parallelize the call to *memcpy*? Explain your answer.

Parallelizing *memcpy* is difficult because if the threads are not mapped to different locations in memory, the threads would overwrite each other. Also due to memory bottleneck, it would need to be written close to the CPUs as writing to memory takes a while. This leads to even more issues with synchronization. Practically, our current knowledge of OpenMP does not my this possible.

* 1. Blah
  2. Blh
  3. Blah

1. Gaussian Elimination
   1. Determine whether the outer loop of the row-oriented algorithm can be parallelized.

The outer loop of the row-oriented algorithm cannot be parallelized because there is a data dependency where each thread would be modifying *x[row]* and calling *x[col]*.

* 1. Determine whether the inner loop of the row-oriented algorithm can be parallelized.

The inner loop of the row-oriented algorithm can be parallelized since each thread can call *x[col]* without *x[col]* being modified since the index, *col*, is outside the purview of the loop.

* 1. Determine whether the outer loop of the column-oriented algorithm can be parallelized.

The outer loop of the column-oriented algorithm cannot be parallelized because there is a data dependency where each thread would be modifying *x[row]* and calling *x[col]*.

* 1. Determine whether the inner loop of the column-oriented algorithm can be parallelized.

The inner loop of the column-oriented algorithm can be parallelized since each thread can call *x[col]* without *x[col]* being modified since the index, *col*, is outside the purview of the loop.

* 3. If your upper triangular system has 10, 000 variables, which schedule gives the best performance? Explain your answer.

The user-time of the serial schedule was faster as there is less overhead and the calculations are rather simple in nature with only needing to multiply and subtract.