# EECS-221-HPC Ramkumar Rajabaskaran 85241493 [rrajabas@uci.edu](mailto:rrajabas@uci.edu) #

\*\*\*\*\*\*\*\*\*\*\*\*\* **HOMEWORK 1** \*\*\*\*\*\*\*\*\*\*\*\*\*

1. To create a program for Parallel-Mergesort with Parallel Merge using OpenMP.

I had access to an Intel 8 core machine and an AMD 64 core machine.

The runtimes(s) on both the machines and the Speedup are: -

Average Speedup for the Intel-8 with 8 threads: - **3.514**

Average Speedup for the AMD-64 with 64 threads: - **6.971**

S1…S5 denotes serial time. P1…P5 denotes parallel time in seconds.



The output results as well as the parallel-mergesort.cc have been included in the tarball.

1. Matrix Transpose
   1. Consider a 4X4 Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| i1 j1 | i1j2 | i1j3 | i1j4 |
| i2j1 | i2j2 | i2j3 | i2j4 |
| i3j1 | i3j2 | i3j3 | i3j4 |
| i4j1 | i4j2 | i4j3 | i4j4 |

The Algorithm is correct as the cells that are read and written by the individual parallel blocks do not conflict or overlap.

In the algorithm, the \*Red\* cells remain the same and are not visited.

In the j=2 outer loop, i can only take the value one. So, the cells that are swapped are the \*Yellow\* cells, i1j2 and i2j1.

In the j=3 outer loop, i can take the values of 1,2. So the cells that are swapped are the \*Blue\* cells, i1j3 and i3j1, i2j3 and i3j2.

In the j=4 outer loop, i can take the values of 1,2,3. So the cells that are swapped are the \*Green\* cells, i1j4 and i4j1, i2j4 and i4j2, i3j4 and i4j3.

Each parallel thread accesses two unique cells and doesn’t conflict with the cells of the other threads.

Hence, in all the outer loops as well as the inner loops, the cells that are visited are different with respect to the other loops, hence there will be no conflict of read/write when they are executed parallelly. Thus this algorithm is correct.

* 1. The Work done by this algorithm is the same , as there are that many cells that need to be swapped.

The span of the algorithm is , since each of the parallel loop spans for a depth of and the total depth is

The parallelism is equal to the .

* 1. The work remains the same, as we do j times the inner loop which goes from 1 to j-1. Hence work is

The span is as in the worst cases, the inner loop needs to run n times.

The parallelism is .