AMATH 583 – Problem Set #5 – 583 only questions
Written By: EJ Rainville, Spring 2021

Recursive Norm

1. **How much parallel speedup do you see for 1, 2, 4, and 8 threads?**

The performance values for the recursive two norm is shown in the following table.

N	Sequential	1 thread	2 threads	4 threads	8 threads	1 thread	2 threads	4 threads	8 threads
1048576	6.47966	5.37674	9.10655	11.108	11.8921	0	0	1.92318e-16	0
2097152	5.09159	4.92024	8.18728	10.1328	12.0401	5.43881e-16	9.51792e-16	5.43881e-16	5.43881e-16
4194304	4.76625	4.63972	8.32203	10.3819	10.8101	9.6159e-16	5.76954e-16	5.76954e-16	3.84636e-16
8388608	4.74139	4.41506	8.32457	10.4858	11.0153	1.3595e-16	0	4.07851e-16	2.719e-16
16777216	4.10631	4.13523	6.59778	6.74945	7.72635	3.84567e-16	1.92284e-16	0	1.92284e-16
33554432	3.72827	3.78078	4.34362	3.7596	3.15065	7.20652e-15	6.66264e-15	6.39069e-15	6.39069e-15

N	Sequential	1 thread	2 threads	4 threads	8 threads	1 thread	2 threads	4 threads	8 threads
1048576	6.23967	6.47966	6.31767	6.3574	6.47966	1.15258e-15	1.34467e-15	1.34467e-15	1.34467e-15
2097152	5.22148	5.04143	5.24826	5.38637	5.19498	2.71928e-15	2.03946e-15	2.03946e-15	2.17543e-15
4194304	4.80998	4.85452	4.78802	4.8771	4.85452	9.61294e-16	0	3.84518e-16	1.92259e-16
8388608	2.86979	4.64051	4.41506	4.62084	4.56284	1.49582e-15	5.43933e-16	4.0795e-16	1.35983e-16
16777216	2.51479	2.63911	3.06633	4.34965	4.28615	5.00099e-15	3.84691e-15	4.2316e-15	4.2316e-15
33554432	3.86795	3.7491	3.89037	3.92449	3.85683	3.67179e-15	5.57568e-15	4.35175e-15	4.62374e-15

Here we see that there is significant speed up using multiple threads for the smaller problems using the 'recursive_two_norm_a' algorithm while there is no noticeable speedup for any size problem using the 'recursive_two_norm_b' algorithm. The speedup values that we see for the smallest problem size for 'recursive_two_norm_a' for 1, 2, 4 and 8 threads are 0.831, 1.41, 1.71 and 1.83, respectively. We do not see parallel speedup since these threads are running concurrently rather than directly in parallel but we still see a speed up of almost 2 using 8 threads which is significant and the performance of the multi-threaded algorithm is much improved.

2. **What will happen if you use ``std:::launch::deferred`` instead of ``std:::launch::async`` when launching tasks? When will the computations happen? Will you see any speedup? For your convenience, the driver program will also call ``recursive_two_norm_b`` -- which you can implement as a copy of ``recursive_two_norm_a`` but with the launch policy changed.**

When using 'std::launch:deferred' the asynchronous tasks are not launched until the output from the task is prompted therefore all the tasks are launched at one time since the output is prompted for each of the tasks at once which then floods the CPU with all of the tasks at once similar to the sequential run which is why we see the performance is similar for the sequential run as it is for the 'deferred' launch method. The 'std::launch::async' method launches tasks immediately therefore each asynchronous task is launched as it is created and starts being computed concurrently as other task are being created which allows the tasks to more spread

out rather than flooding the CPU at the 'deferred' method or sequential computation does which leads to the improved performance that we see for the 'async' launch method.