Adi David

CS 447

Professor Chiu

Project 3 Report

The method used to solve sin(x)/x integrals with bounds a and b in my implementation was the Monte Carlo method. By using the C++ pthread library, I was able to utilize threading to solve the integral in separate parts in order to improve the performance. However, once I exceed the number of threads threshold, the speedup plateaus and the efficiency diminishes. This is shown in the graph below which displays the data of 100 iterations of the executable from 1 thread to 100. Additionally, it was run using 20,000,000 samples with bounds 1 and 10 for the integral. As represented in the graph, the speedup continues to double up until around 8 threads where the speedup sits around 7 to 8 times as fast as the time with one thread. Therefore, the efficiency, which is measured by S_p / p, decreases as the number of threads increases to a certain extent. Since this was run using the remote machines provided by Binghamton, the thread threshold is most likely lower than a high end PC with a heavy duty processor that has more cores to handle more threads. Still, when running with 20,000,000 samples, 8 threads, and 1 to 10 bounds, it completes execution in 0.243s with an accurate estimate of the integral. Moreover, the number of samples used was crucial in outputting reliable data as the higher sample count produced consistent results. When testing with 100 or even 1,000 samples, both the estimate for the sin(x)/x and the speedup times varied between executions. From this implementation, the benefit of parallel processing becomes clear as the speedup of operations dramatically increases when using threading.