Numerical Integration (CS 447/547)

Ritik Ramaiya : B00960003

The goal of this project is to analyze the effect of multithreading on speedup and efficiency through a program that utilizes Monte Carlo Integration to compute a definite integral. The reasoning behind the choice of Monte Carlo is its simplicity in employing a non-deterministic approach. Monte Carlo Integration randomly chooses points to calculate an approximate value of the integral. My program makes a few other choices regarding global variables and race conditions. For the limits of integration and number of threads, I decided to use global variables because they are only involved in reads and are never written to. Therefore, there is no risk of race condition. On the other hand, I utilize an array of doubles and pointers to each index on the array to store the return value from each thread. This eliminates race conditions, keeping the results consistent and accurate.

I tested my program on up to 8 cores. The results are shown in *Speedup_Graph.png* and *Efficiency_Graph.png*. Speedup increased linearly with number of threads, up to a total of 8 threads. Reason being that as threads increase, my program divides the number of iterations equally amongst the threads. Furthermore, the absence of any locking mechanism and shared global variables allows the threads to run asynchronously on separate cores. Beyond 8, the speedup stays the same because the runtime is limited by the number of cores available on the remote machines. Running more than 8 threads on a machine with 8 cores adds overhead instead of improving runtime. Efficiency also varied between 100% at lesser number of threads and 95% at higher number of threads.