James Hahn
CS1645 High Performance Computing
Dr. Jaber Hasbestan
Homework #3

Algorithm presentation

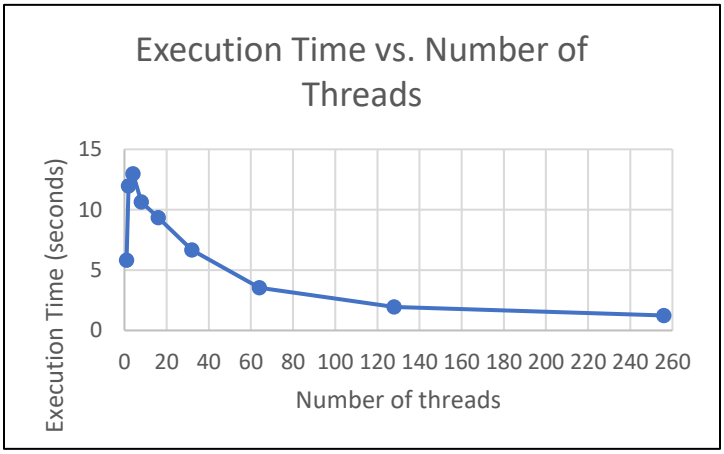
As an example, let's say the integral is being calculated on the interval from 0 to 1, with 8 threads, each thread computes the integral on some interval of length 1 / 8 = .125 . So, thread 0 would compute the integral on the interval [0, .125], thread 1 computes [.125, .250], and so on until thread 7 computes [.875, 1]; each interval is referred to as a chunk. Additionally, the user passes in a value **n** to determine the how granular each chunk will be. For example, if there are 8 chunks, but the user specifies **n = 32**, then each chunk will contain 32 / 8 = 4 parts. So, each thread will further break down their interval. For example, thread 0 will compute the integral over intervals [0, .03125], [.03125, .0625], [.0625, .09375], and [.09375, .125], further increasing the precision of our integral value.

Results

All results were computed for different values of n, resulting in higher accuracy in the computation of pi. A higher number of threads generally decreases the execution time of computation while a higher value of n increases the accuracy of the integral.

All times were gathered by using C's ctime library using the *timespec* structure. All times were averaged over five simulations.

N = 200,000,000 points	
N_t	**Time (seconds)**
1	5.8180
2	11.9517
4	12.9700
8	10.6460
16	9.3330
32	6.6676
64	3.5365
128	1.9335
256	1.2401



Oddly enough, initially, as the number of threads increases, the execution time increases as well. This is due to added overhead of maintaining the extra threads. However, once eight threads is used, execution time begins to decrease. Furthermore, with a shift from 32 to 64 threads, execution time halves, which is ideal and satisfactory. However, the times begin to level out as the number of threads exceeds 128. As shown in the graph, execution time increases, decreases as desired, and finally levels out around an execution time of a second.

Additionally, as n increases, the accuracy of the integral increases. This is shown in the below table. The values converge toward pi with a decimal precision of 9 digits.

n	**Integral value**
1	3.00000000
2	3.10000000
4	3.13117647
8	3.13898849
16	3.14094161
32	3.14142989
64	3.14155196
128	3.14158248
256	3.14159011

What happens if you over specify the cores?

The program results in a segmentation fault. In fact, on the University of Pittsburgh CRC's (Center for Research Computing) machines, the maximum number of threads able to be utilized is 32,749; 32,750 threads will produce a segmentation fault. This number was discovered through personal exploration.