Parallel Computing in MPI Report.

Arya HajiTaheri

Vistas in Advanced Computing

6 August 2017

**PRIME:**

For assignment one, the C program tries to find prime numbers between 0 – NMAX using a naïve algorithm. I boosted the performance by measuring the time it takes to scan all prime numbers (excluding printing time).

For 1 serial process: it took 20.351 seconds.

For 2 nodes, it took 10.145 seconds.

For 4 nodes, it took 5.090 seconds.

For 8 nodes, it took 2.555 seconds.

For 16 nodes, however, it took 1.276 seconds.

**PI using integration:**

For assignment two, I used the trapezoidal rule to calculate pi. The Exact equation is:

Delta x for approximation is 50000. Estimate had error of <0.04%

For 2 nodes, it took 0.001033 seconds.

For 4 nodes, it took .000678 seconds.

For 8 nodes, it took 0.000594 seconds.

For 16 nodes, however, it took 0.000519 seconds.

**PI using Monte Carlo Method:**

Pi was also calculated in parallel using the Monte Carlo Method.

50000 iterations were used to calculate PI. Error is around 0.05% to 0.8%.

For 2 nodes, it took 0.002123 seconds.

For 4 nodes, it took .001278 seconds.

For 8 nodes, it took 0.000875 seconds.

For 16 nodes, however, it took 0.000704 seconds.

The Monte Carlo method seems to be less reliable in terms of accuracy, and slower to compute. However, the Monte Carlo method is more responsive to parallelism than trapezoidal rule.

**Rectangular Poisson using Jacobi**

A 24x24 table was used. This works with 8 nodes only, since it was optimized to divide the mesh into 8 processes.

Estimated time is ~0.00679 seconds to reach < 1.0e-3 error.