Assignment 2
High Performance Computing I
CE620/CSE547/MAE609/MTH667/PHY515
Fall 2015
Due 11:59 PM 10/18/2015

Instructions

Work all problems. You may consult with others, but the work submitted must be your own. Code may be written in `C`, `C++`, or `Fortran`. Submit all code through UBLearns, including steps to compile and run different experiments. Please submit written answers electronically through UBLearns. Clarity of code and plots will be a component of the credit you receive.

Problem 1

Consider the matrix addition problem from the first assignment.

1. Profile performance using gprof. Report on your findings for both variations of the loop order.
2. Now instrument your code using PAPI. Use PAPI to study the number of floating point operations and cache misses. Compare both loop-order variations.

Problem 2

Wikipedia defines the Mandelbrot set as " is the set of values of c in the complex plane for which the orbit of 0 under iteration of the complex quadratic polynomial $z_{i+1} = z_i^2 + c$ remains bounded."

We will iteratively estimate the area of the Mandelbrot set. In particular, we will use the above iteration and use the condition $|z| < 2$ as the check for boundedness.

1. Write a program to estimate the Mandelbrot set by partitioning the relevant part of the complex plane into a structured grid and iterate many times, say 10000, to check for convergence of the iteration for different c values (determined by the grid). Use an inital iterate of $z_0 = 0$.
2. Study the performance of your code.
3. Now, parallelize this code using OpenMP. Ensure that your results are correct, using your serial algorithm. Study both strong scaling and weak scaling. Be sure to experiment with different scheduling strategies.

4. Study paper included with the assignment on the Karp-Flatt metric. Use the Karp-Flatt metric to estimate the "serial fraction" of your code.

Problem 3

Wikipedia states that the Goldbach conjecture is "Every even integer greater than 2 can be expressed as the sum of two primes." We will write a code to test the Goldbach conjecture.

1. Write a code to both verify that the Goldbach conjecture is true up to an arbitrary number of even integers. A primitive primality test for an integer m is to check for any divisors for all integers $i \leq \sqrt{m}$.
2. Study the performance of your code.
3. Now, parallelize the outer loop of your code using OpenMP. Ensure that your results are correct, using your serial algorithm. Study both strong scaling and weak scaling. Be sure to experiment with different scheduling strategies.
4. Use the Karp-Flatt metric to estimate the "serial fraction" of your code.
5. Now, try parallelizing the *inner loop* instead of the outer loop. What do you observe? Be sure to study the scaling.

Problem 4

One of many ways to estimate π is to compute the following integral:

$$\frac{\pi}{4} = \int_0^1 \sqrt{1 - x^2}\, dx$$

A rudimentary numerical integration technique is the "midpoint rule" whereby we partition the domain into bins $i = 0, ..., N-1$, with spacing $\Delta x_i = x_{i+1} - x_i$, and, then, computing, and summing, local integral estimates of the function $f(x)$ by summing the estimate of the integral of the local bin using the area of the rectangle formed by $f(x_i)\Delta x_i$. In particular, assuming each bin is of equal length,

$$\int_a^b f(x)\, dx \approx \frac{b-a}{N}\left(\frac{f(a)}{2} + \sum_{k=1}^{N-1} f\left(a + k\frac{b-a}{N}\right) + \frac{f(b)}{2}\right)$$

1. Write a program to estimate π using the midpoint rule.
2. Study the performance of the code.
3. Now, parallelize your code using OpenMP. Ensure that your results are correct, using your serial algorithm. Study both strong scaling and weak scaling. Be sure to experiment with different scheduling strategies.
4. Use the Karp-Flatt metric to estimate the "serial fraction" of your code.