Name : Saurabh Nitnaware
Experiment No : 5
Write a parallel program to calculatethe value of PI/Area of Circle using OpenMP library.
Date of Performance : 22/02/24
Date of Submission : 19/04/24

Aim: Write a parallel program to calculate the value of PI/Area of Circle using OpenMP library.

Objective: To Analyze and understand the OpenMp library.

Theory:

To calculate the value of PI/Area of Circle in parallel using OpenMP, we can use the Monte Carlo method. This method involves generating random points inside a square and counting the number of points that fall within a circle inscribed in that square. The ratio of the number of points inside the circle to the total number of points is proportional to the ratio of the area of the circle to the area of the square. We can use this ratio to estimate the value of PI.

Here is a step-by-step algorithm for a parallel program to calculate the value of PI/Area of Circle using OpenMP library:

1. *Initialize the number of threads and the number of points to generate.*
2. *Set up a parallel region using the OpenMP library.*
3. *Within the parallel region, each thread should generate a subset of the total number of points to calculate.*
4. *Each thread generates a random (x, y) coordinate pair within the range [-1, 1].*
5. *Calculate the distance of the point from the origin using the distance formula: distance = sqrt(x^2 + y^2).*
6. *If the distance is less than or equal to 1 (i.e., the point falls within the circle), increment a thread-local count.*
7. *Use an OpenMP reduction to combine the counts from each thread.*
8. *Calculate the ratio of the number of points that fell within the circle to the total number of points generated.*
9. *Multiply the ratio by 4 to estimate the value of PI.*
10. *Output the estimated value of PI*

Code:

```
#include <stdio.h>
#include <omp.h>
#define NUM_STEPS 1000000
int main() {
    double pi = 0.0;
    double step = 1.0 / NUM_STEPS;
    #pragma omp parallel
    {
        double x;
        double sum = 0.0;
        #pragma omp for
        for (int i = 0; i < NUM_STEPS; i++) {
            x = (i + 0.5) * step;
            sum += 4.0 / (1.0 + x * x);
        }
        #pragma omp critical
        {
            pi += sum * step;
        }
    }
    double area = pi * pi;
    printf("Value of PI/Area of Circle: %f\n", area);
    return 0; }
```

Conclusion: The OpenMP program for computing π/area of a circle showcases parallelization of a demanding task. By assigning work to multiple threads, system processing power is efficiently utilized. Each thread calculates a partial sum of the series for π approximation, safely combining partial sums in a critical section. Parallel implementation accelerates computation, ideal for repetitive math tasks. Overall, the program underscores parallel computing's efficacy in enhancing numerical computation speed.