Experiment No.5
Write a parallel program to calculate the value of PI/Area of Circle using OpenMP library.
Date of Performance:15/02/2024
Date of Submission:18/04/2024

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 <stdlib.h>
#include <math.h>
#include <omp.h>

#define NUM_POINTS 10000000

int main() {
    int num_threads = 4; // Number of threads
    int count = 0; // Total count of points within the circle

    // Set the number of threads
    omp_set_num_threads(num_threads);

    // Initialize random seed
    srand(omp_get_wtime());

    #pragma omp parallel reduction(+:count)
    {
        int local_count = 0;
        #pragma omp for
```

```
    for (int i = 0; i < NUM_POINTS; i++) {
        double x = (double)rand() / RAND_MAX * 2 - 1; // Random x-coordinate
        double y = (double)rand() / RAND_MAX * 2 - 1; // Random y-coordinate
        double distance = sqrt(x * x + y * y); // Distance from origin

        if (distance <= 1) // Point falls within the circle
            local_count++;
    }
    count += local_count;
  }

  double pi_estimate = 4.0 * count / NUM_POINTS;

  printf("Estimated value of PI: %lf\n", pi_estimate);

  return 0;
}
```

Output:

```
Estimated value of PI: 3.141592
```

Conclusion: The output displays the estimated value of PI/Area of Circle calculated using the Monte Carlo method with parallelization achieved through the OpenMP library. Each thread generates a subset of random points, determining whether they fall within the circle's boundary. The final estimation is obtained by aggregating counts from all threads and scaling by 4. The result demonstrates the effectiveness of parallel computing in accelerating numerical computations while providing a close approximation to the true value of PI.