Name: Manthan Dhole
Id : 201080020
Sub : Parallel Computing

Experiment 8

Aim: To Write a Hybrid program for MPI + OpenMP for calculating Pi, find the volume of cube, cuboid, tetrahedron and single side closed cylinder.

Theory:

OpenMP

OpenMP (Open Multi-Processing) is an API (Application Programming Interface) for parallel programming in shared-memory architectures. It allows the programmer to parallelize the execution of a program by specifying which parts of the code should be executed in parallel.

OpenMP is a set of compiler directives and library routines that enable the programmer to add parallelism to their code. The directives are typically added as pragmas in the C, C++, and Fortran programming languages, and specify how the compiler should parallelize the code. OpenMP supports many types of parallelism, including task parallelism, loop parallelism, and section parallelism.

OpenMP programs can be run on a variety of shared-memory architectures, including multi-core processors and symmetric multiprocessors (SMPs). OpenMP also provides a run-time library

that allows the programmer to specify how threads should be created, managed, and synchronized.

One of the advantages of OpenMP is that it allows for incremental parallelization of code, meaning that parts of the code can be parallelized one at a time, rather than requiring a complete rewrite of the code. This makes it easier to add parallelism to existing code.

Overall, OpenMP is a useful tool for programmers who want to take advantage of parallelism in shared-memory architectures.

MPI

MPI (Message Passing Interface) is a standardized communication protocol that allows multiple processes running on different computing nodes to exchange messages and data. It is commonly used in parallel and distributed computing to implement communication between processes, especially in high-performance computing (HPC) applications.

MPI provides a set of functions and data types that enable processes to send and receive messages with each other, synchronize their activities, and coordinate their computations. It supports various communication modes, such as point-to-point, collective, and one-sided communication, and can be used with different programming languages, including C, C++, Fortran, and Python.

MPI has several implementations available, including MPICH, Open MPI, and Intel MPI, and is widely used in scientific, engineering, and data analysis applications that require parallel processing on multiple processors or computing clusters.

Code:

```
%%writefile exp8.c
#include <stdio.h>
#include <math.h>
#include <mpi.h>
#include <omp.h>

double leibniz_pi(int n, int rank, int size) {
    double pi = 0.0;
    int sign = (rank % 2 == 0) ? 1 : -1;

    #pragma omp parallel for reduction(+:pi)
    for (int i = rank; i < n; i += size) {
```

```
        double term = 1.0 / (2*i + 1);
        pi += sign * term;
        sign *= -1;
    }

    double global_pi = 0.0;
    MPI_Allreduce(&pi, &global_pi, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
    return global_pi * 4;
}

double cube_volume(double a, double pi) {
    return pow(a, 3);
}

double cuboid_volume(double l, double w, double h, double pi) {
    return l * w * h;
}

double tetrahedron_volume(double b, double h, double pi) {
    return (1.0/3.0) * b * h;
}

double cylinder_volume(double r, double h, double pi) {
    return pi * pow(r, 2) * h;
}

int main(int argc, char** argv) {
    int n = 1000000;
    int rank, size;
    double pi;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    double start_time = MPI_Wtime();

    pi = leibniz_pi(n, rank, size);

    double end_time = MPI_Wtime();
    double elapsed_time = end_time - start_time;

    if (rank == 0) {
        printf("Pi = %.16f\n", pi);
        printf("Leibniz series terms = %d\n", n);
        printf("Elapsed time = %.4f seconds\n", elapsed_time);

        // Calculate volumes
        double a = 2.0;
        double l = 3.0;
```

```
      double w = 4.0;
      double h = 5.0;
      double r = 2.0;

      printf("Volume of Cube = %.2f\n", cube_volume(a, pi));
      printf("Volume of Cuboid = %.2f\n", cuboid_volume(l, w, h, pi));
      printf("Volume of Tetrahedron = %.2f\n", tetrahedron_volume(0.5*l*w, h, pi))
;
      printf("Volume of single side closed Cylinder = %.2f\n", cylinder_volume(r, h,
pi));
   }

   MPI_Finalize();
   return 0;
}
```

Code link:

https://colab.research.google.com/drive/18f3WynXoKN1R_zmImXV7Mk1DgbuL0-s_#scrollTo=WVEdOoq6GH0v

Output:

```
!mpirun -np 3 --allow-run-as-root --oversubscribe ./exp8

Pi = 3.0752815768839961
Leibniz series terms = 1000000
Elapsed time = 0.0646 seconds
Volume of Cube = 8.00
Volume of Cuboid = 60.00
Volume of Tetrahedron = 10.00
Volume of single side closed Cylinder = 61.51
```

Conclusion: We have Written a Hybrid program for MPI + OpenMP for calculating Pi, find the volume of cube, cuboid, tetrahedron and single side closed cylinder.