Name : Saurabh Nitnaware
Experiment No : 7
Implement a parallel program to demonstrate the cube of N numberwithin a set range.
Date of Performance : 28/03/24
Date of Submission : 19/04/24

Aim: Implement a parallel program to demonstrate the cube of N number within a set range

Objective: To Build the logic to parallelize the programming task.

Theory:

An algorithm is a sequence of instructions followed to solve a problem. While designing an algorithm, we should consider the architecture of computer on which the algorithm will be executed. As per the architecture, there are two types of computers −

- **Sequential Computer**
- **Parallel Computer**

Depending on the architecture of computers, we have two types of algorithms −

Sequential Algorithm − An algorithm in which some consecutive steps of instructions are executed in a chronological order to solve a problem.

Parallel Algorithm − The problem is divided into sub-problems and are executed in parallel to get individual outputs. Later on, these individual outputs are combined together to get the final desired output.

It is not easy to divide a large problem into sub-problems. Sub-problems may have data dependency among them. Therefore, the processors have to communicate with each other to solve the problem.

It has been found that the time needed by the processors in communicating with each other is more than the actual processing time. So, while designing a parallel algorithm, proper CPU utilization should be considered to get an efficient algorithm.

Note that to ensure that each processor has enough data to perform the multiplication, we may need to pad the matrices with extra rows or columns.

Code:

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#define N 10 // Define the upper limit of the range
int main(int argc, char** argv) {
    int rank, size;
    int start, end; // Range for each process
    int i, cube;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    start = rank * (N / size) + 1;
    end = (rank + 1) * (N / size);
    if (rank == size - 1) {
        end = N;
    }
    for (i = start; i <= end; i++) {
        cube = i * i * i;
        printf("Process %d: Cube of %d is %d\n", rank, i, cube);
    }
    MPI_Finalize();
```

```
    return 0;
}
```

Output:

D:\COMPUTER ENGINEERING\SEMESTER 8\HPC\MPI\x64\Debug>mpiexec -n 2 ./MPI

Process 0: Cube of 1 is 1

Process 0: Cube of 2 is 8

Process 0: Cube of 3 is 27

Process 0: Cube of 4 is 64

Process 0: Cube of 5 is 125

Process 1: Cube of 6 is 216

Process 1: Cube of 7 is 343

Process 1: Cube of 8 is 512

Process 1: Cube of 9 is 729

Process 1: Cube of 10 is 1000

Conclusion: The MPI-based parallel program efficiently computes the cube of numbers within a specified range, demonstrating the scalability and performance benefits of distributed computing. This approach underscores the importance of optimizing algorithms for parallel execution, equipping engineering students with essential skills for tackling real-world engineering challenges.