Name : Manthan Dhole
ID : 201080020
Sub : Parallel Computing

Experiment 9

Aim : To write a Hybrid program for MPI + CUDA for Hello World.

Theory :

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.

CUDA

CUDA (or Compute Unified Device Architecture) is a parallel computing platform and application programming interface (API) that allows software to use certain types of graphics processing units (GPUs) for general purpose processing, an approach called general-purpose computing on GPUs (GPGPU). CUDA is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels.

Algorithm :

The **leibniz_pi** function uses OpenMP to parallelize the loop that computes the series terms, and MPI to combine the results from all processes using the **MPI_Allreduce** function.

The **cube_volume**, **cuboid_volume**, **tetrahedron_volume**, and **cylinder_volume** functions simply compute the volumes of the different shapes using the provided formulas.

The **main** function initializes MPI, calculates Pi using **leibniz_pi**, calculates the volumes of the different shapes, and prints out the results if the process rank is 0. Finally, MPI is finalized and the program returns 0.

Code :

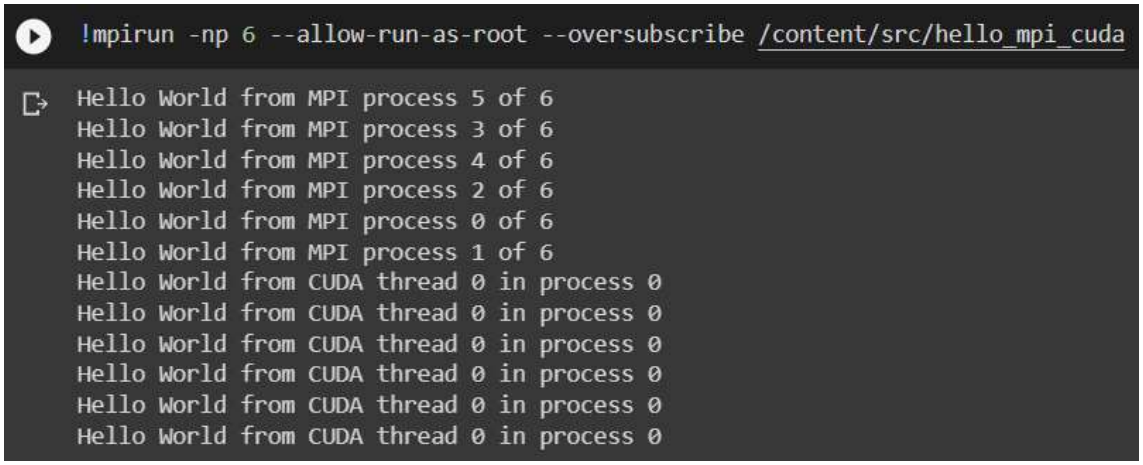
```
%%cuda --name hello_mpi_cuda.cu
#include <stdio.h>
#include <mpi.h>
#include <cuda_runtime.h>

__global__ void helloCUDA() {
    printf("Hello World from CUDA thread %d in process %d\n", threadIdx.x, blockIdx
.x);
}

int main(int argc, char** argv) {
    int rank, size;
```

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

    printf("Hello World from MPI process %d of %d\n", rank, size);

    // Call the CUDA kernel from each MPI process
    helloCUDA<<<1, 1>>>();
    cudaDeviceSynchronize();

    MPI_Finalize();
    return 0;
}
```

Code link:

https://colab.research.google.com/drive/11y0Haav5bEqjqqCe-ywzQI6P0Z1Dh6eO

Output:



Conclusion: We have Written a Hybrid program for MPI + CUDA for Hello World.