High Performance Computing Lab

Practical No. 7

Title of practical:

Installation of MPI & Implementation of basic functions of MPI

Problem Statement 1:

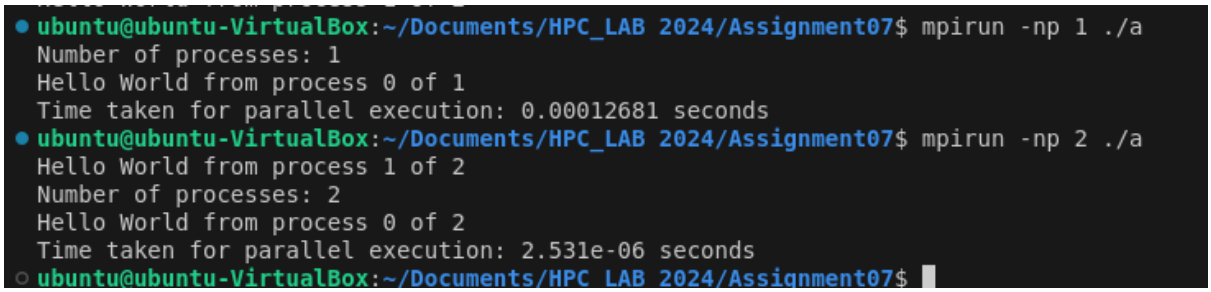
Implement a simple hello world program by setting number of processes equal to 10

Code:

```
#include <mpi.h>

#include <iostream>
#include <chrono>

int main(int argc, char *argv[]) {
int rank, size;
// Initialize MPI environment
MPI_Init(&argc, &argv);
// Get the rank of the process
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
// Get the number of processes
MPI_Comm_size(MPI_COMM_WORLD, &size);

if (rank == 0) {
std::cout << "Number of processes: " << size << std::endl;
}

// Start timer for parallel execution
auto start = std::chrono::high_resolution_clock::now();

// Print "Hello World" from each process
std::cout << "Hello World from process " << rank << " of " << size <<
std::endl;
// End timer for parallel execution
auto end = std::chrono::high_resolution_clock::now();
std::chrono::duration<double> elapsed_time = end - start;

if (rank == 0) {
std::cout << "Time taken for parallel execution: " <<
elapsed_time.count() << " seconds" << std::endl;
}
```

```
// Finalize the MPI environment
MPI_Finalize();

return 0;
}
```

Screenshots:



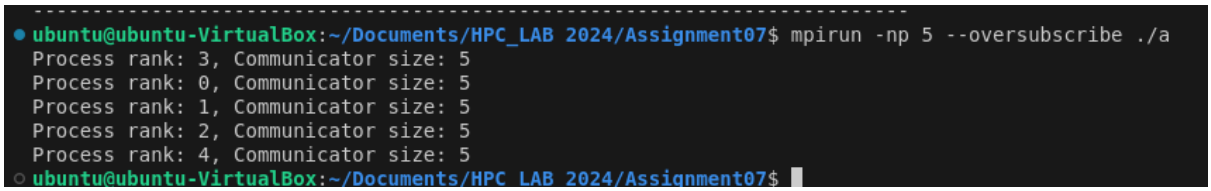
Information :

With varying number of processors --> Sequence of execution changes as well as time required also changes

Problem Statement 2:

Implement a program to display rank and communicator group of five processes

Screenshots:



Information:

With varying number of processors --> Sequence of execution changes