Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

Class: Final Year (Computer Science and Engineering)

Year: 2022-23 **Semester:** 1

Course: High Performance Computing Lab

Practical No. 11

Exam Seat No : 2019BTECS00029

Name : Harshal Kodgire

Title of practical : Analysis of MPI Programs

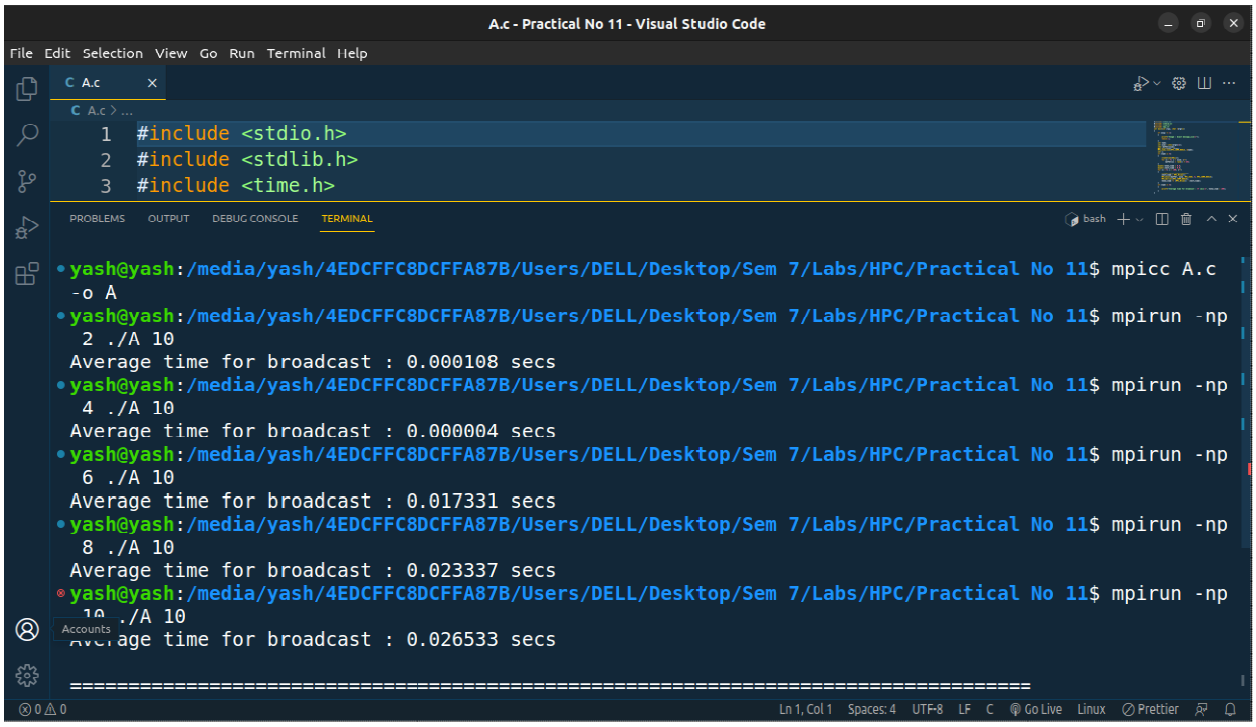
Problem Statement 1:
Execute the MPI program (Program A) with a fixed size broadcast. Plot the performance of the broadcast with varying numbers of processes (with constant messagesize). Explain the performance observed.

Screenshot 1:

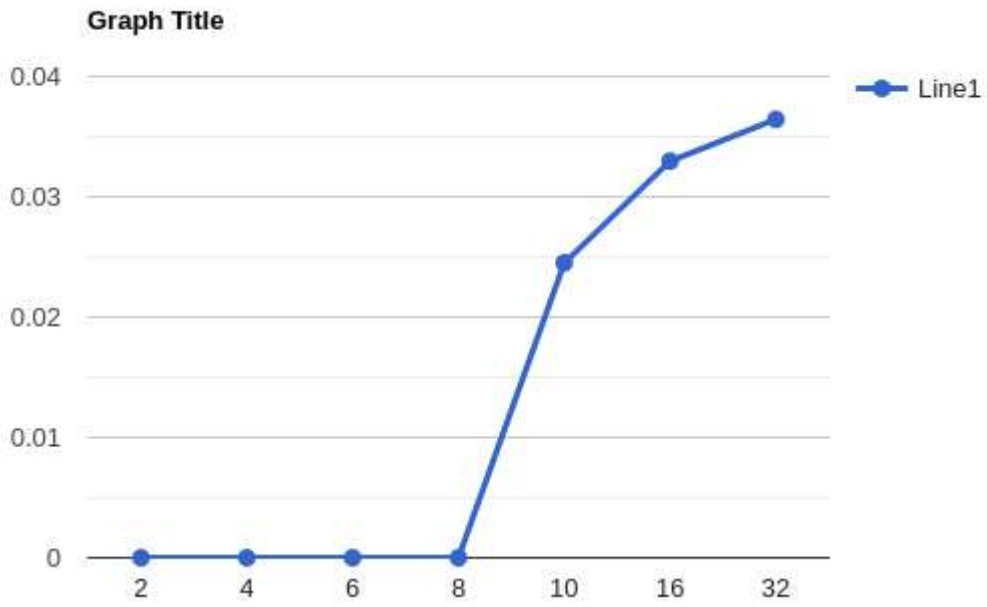
```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<mpi.h>
intmain(intargc,char*argv[])
{
    if(argc!=2)
    {
        printf("Usage : bcastmessage_size\n");
        return1;
    }
    int rank;
    int size =atoi(argv[1]);
    charbuffer[size];
```

```
    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    int i;
    if(rank ==0)
    {
        srand(time(NULL));
        for(i =0; i < size; i++)
            buffer[i]=rand()%256;
    }
    doubletotal_time=0.0;
    doublestart_time=0.0;
    for(i =0; i <100; i++)
    {
        MPI_Barrier(MPI_COMM_WORLD);
        start_time=MPI_Wtime();
        MPI_Bcast(buffer, size, MPI_CHAR,0, MPI_COMM_WORLD);
        MPI_Barrier(MPI_COMM_WORLD);
        total_time+=(MPI_Wtime()-start_time);
    }
    if(rank ==0)
    {
        printf("Average time for broadcast :
%fsecs\n",total_time/100);
    }
}
```

Screenshot 2:



Screenshot 3:



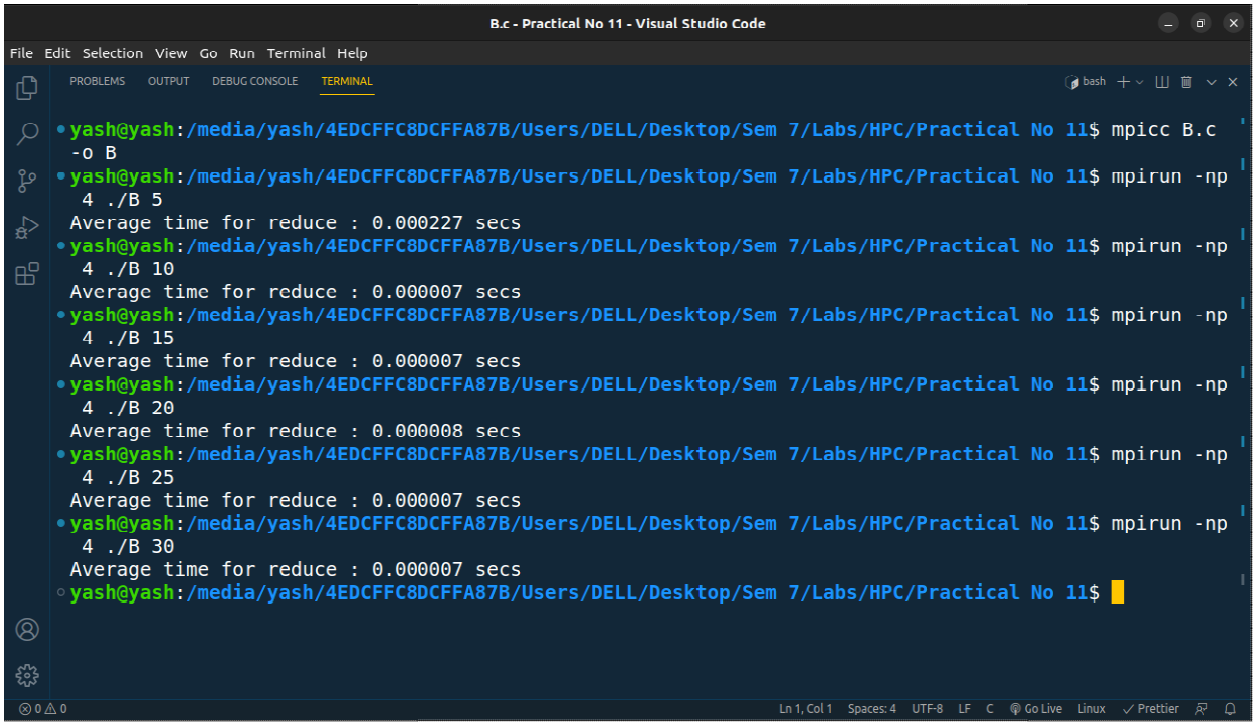
Problem Statement 2:
Repeat problem 2 above with varying message sizes for reduction (Program B).Explain the observed performance of the reduction operation.

Screenshot 4:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<mpi.h>
intmain(intargc,char*argv[])
{
    if(argc!=2)
    {
        printf("Usage : reduce message_size\n");
        return1;
    }
    int rank;
    int size =atoi(argv[1]);
    charinput_buffer[size];
    charoutput_buffer[size];
    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    int i;
    srand(time(NULL));
    for(i =0; i < size; i++)
        input_buffer[i]=rand()%256;
    doubletotal_time=0.0;
    doublestart_time=0.0;
```

```
    for(i =0; i <100; i++)
    {
        MPI_Barrier(MPI_COMM_WORLD);
        start_time=MPI_Wtime();
        MPI_Reduce(input_buffer,output_buffer, size, MPI_BYTE,
MPI_BOR,0, MPI_COMM_WORLD);
        MPI_Barrier(MPI_COMM_WORLD);
        total_time+=(MPI_Wtime()-start_time);
    }
    if(rank ==0)
    {
        printf("Average time for reduce : %fsecs\n",total_time/100);
    }
}
```

Screenshot 5:



Screenshot 6: