Github Link for Code - https://github.com/Aashita06/HPC_Practicals

Q.1) Execute the all-to-all broadcast operation (Program C) with varying message sizes. Plot the performance of the operation with varying message sizes from 1K to 10K (with constant number of processes, 8). Explain the performance observed.
→

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        printf("Usage : alltoall message_size\n");
        return 1;
    }
    int rank;
    int num_procs;
    int size = atoi(argv[1]);
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    int i;
    char input_buffer[size * num_procs];
    char recv_buffer[size * num_procs];
    srand(time(NULL));
    for (i = 0; i < size; i++)
        input_buffer[i] = rand() % 256;
    int j;
    for (j = 1; j < num_procs; j++)
    {
        int k = 0;
        for (i = j * size; i < j * size + size; i++)
        {
            input_buffer[i] = input_buffer[k];
```

```
            k++;
        }
    }
    double total_time = 0.0;
    double start_time = 0.0;
    for (i = 0; i < 100; i++)
    {
        MPI_Barrier(MPI_COMM_WORLD);
        start_time = MPI_Wtime();

        MPI_Alltoall(input_buffer, size, MPI_CHAR, recv_buffer, size,
MPI_CHAR, MPI_COMM_WORLD);
        MPI_Barrier(MPI_COMM_WORLD);
        total_time += (MPI_Wtime() - start_time);
    }
    if (rank == 0)
    {
        printf("Average time for alltoall : %f secs\n", total_time / 100);
    }
    MPI_Finalize();
}
```
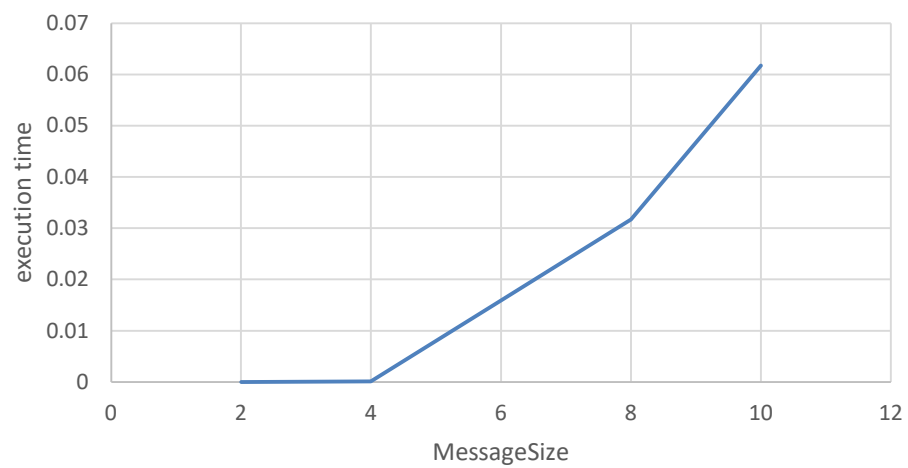
Output:

PROBLEMS 7   OUTPUT   TERMINAL   GITLENS   DEBUG CONSOLE

```
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 4 .\alltoallbroadcast.exe 1000
Average time for alltoall : 0.000039 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 4 .\alltoallbroadcast.exe 2000
Average time for alltoall : 0.000100 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 4 .\alltoallbroadcast.exe 3000
Average time for alltoall : 0.000047 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 4 .\alltoallbroadcast.exe 4000
Average time for alltoall : 0.000113 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 4 .\alltoallbroadcast.exe 5000
Average time for alltoall : 0.000096 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 4 .\alltoallbroadcast.exe 6000
Average time for alltoall : 0.000096 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 4 .\alltoallbroadcast.exe 7000
Average time for alltoall : 0.000079 secs
```

```
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 4 .\alltoallbroadcast.exe 8000
Average time for alltoall : 0.000074 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 4 .\alltoallbroadcast.exe 9000
Average time for alltoall : 0.000066 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 4 .\alltoallbroadcast.exe 10000
Average time for alltoall : 0.000152 secs
```

Analysis

execution time

MessageSize

0   2   4   6   8   10   12

0.07
0.06
0.05
0.04
0.03
0.02
0.01
0

Q.2) Execute the all-reduce operation (Program D) with varying number of processes (1 to 16) and fixed message size of 10K words. Plot the performance of the operation with varying number of processes (with constant message size). Explain the performance observed.
→
Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <mpi.h>
int main(int argc, char *argv[]) {
 if (argc != 2) {
 printf("Usage : allreduce message_size\n");
 return 1;
 }
 int rank;
 int size = atoi(argv[1]);
 char input_buffer[size];
 char recv_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;
 double total_time = 0.0;
 double start_time = 0.0;
 for (i = 0; i < 100; i++)
{
 MPI_Barrier(MPI_COMM_WORLD);
 start_time = MPI_Wtime();

 MPI_Allreduce(input_buffer,recv_buffer,size,MPI_BYTE,MPI_BOR,MPI_COMM_WORLD);
 MPI_Barrier(MPI_COMM_WORLD);
 total_time += (MPI_Wtime() - start_time);
 }
 if (rank == 0) {
 printf("Average time for allreduce : %f secs\n", total_time/100);
 }
 MPI_Finalize();
}
```

Output:

```
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 1 .\allreduce.exe 10000
Average time for allreduce : 0.000001 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 2 .\allreduce.exe 10000
Average time for allreduce : 0.000029 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 3 .\allreduce.exe 10000
Average time for allreduce : 0.000074 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 4 .\allreduce.exe 10000
Average time for allreduce : 0.000095 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 5 .\allreduce.exe 10000
Average time for allreduce : 0.007221 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 6 .\allreduce.exe 10000
Average time for allreduce : 0.015912 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 7 .\allreduce.exe 10000
Average time for allreduce : 0.021371 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 8 .\allreduce.exe 10000
Average time for allreduce : 0.031690 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 9 .\allreduce.exe 10000
Average time for allreduce : 0.043294 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 10 .\allreduce.exe 10000
Average time for allreduce : 0.061745 secs
```

```
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 11 .\allreduce.exe 10000
Average time for allreduce : 0.056595 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 12 .\allreduce.exe 10000
Average time for allreduce : 0.069643 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 13 .\allreduce.exe 10000
Average time for allreduce : 0.093954 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 14 .\allreduce.exe 10000
Average time for allreduce : 0.084082 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 15 .\allreduce.exe 10000
Average time for allreduce : 0.090087 secs
PS C:\Users\Ashitra\OneDrive\Desktop\Mpip> mpiexec -n 16 .\allreduce.exe 10000
Average time for allreduce : 0.131134 secs
```

Analysis

time required