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

High Performance Computing Lab

Class: Final Year (Computer Science and Engineering) **Year:** 2022-23

PRN: 2019BTECS00089 – Piyush Pramod Mhaske **Batch**: B3

Practical 11

Github link: https://github.com/Piyush4620/2019BTECS00089HPCLab

Hosted Link : https://better-sidecar-c10.notion.site/HPC-038e2693a633408c8604841fc50f74e2

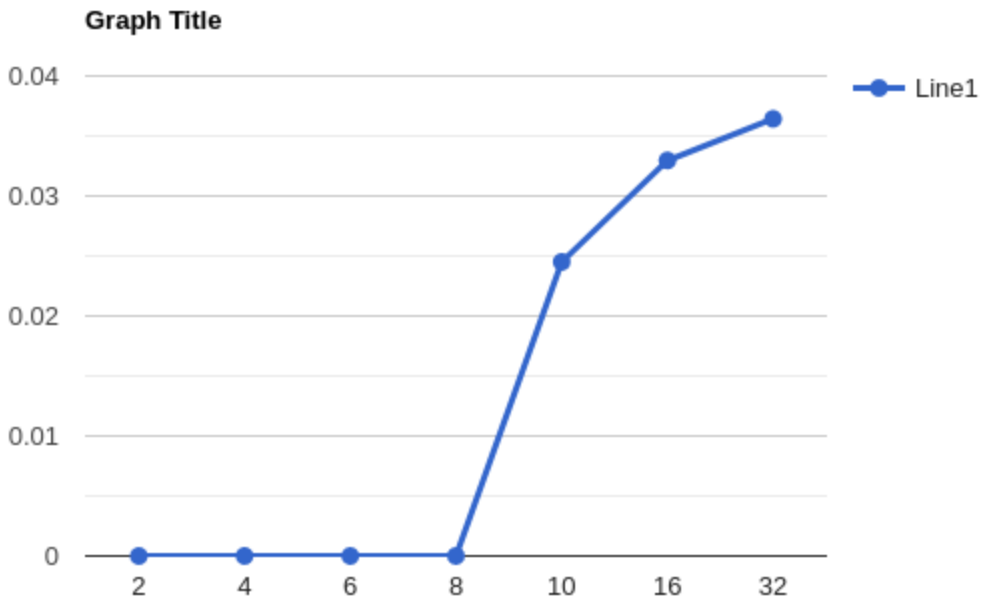
Problem statement 1

Q1. 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.

```
#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();
}
```

Performance:



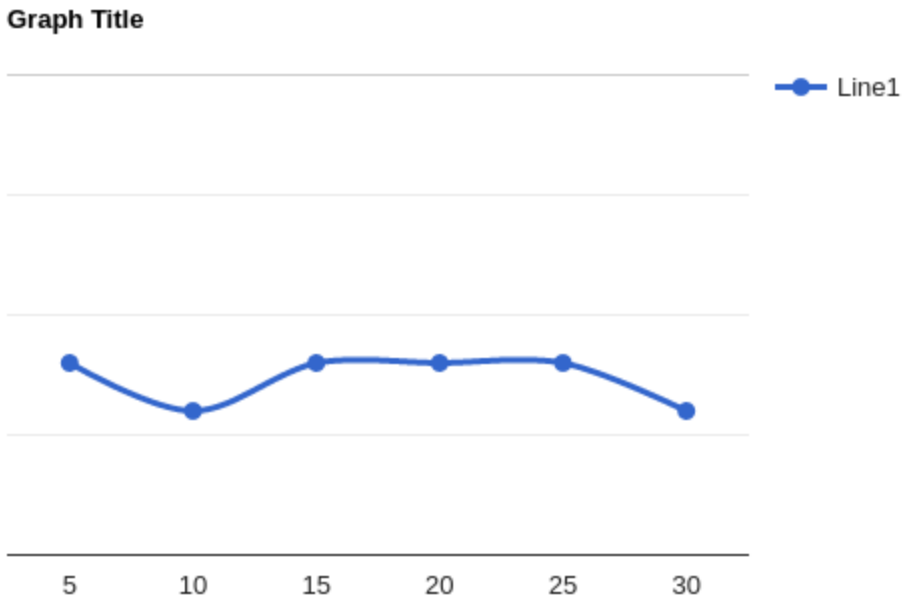
Problem statement 2:

Q2. 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.

```
#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();
}
```

Performance:


Graph Title
Line1
5
10
15
20
25
30