

Class: Final Year (Computer Science and Engineering)

Year: 2021-22 **Semester:** 1

Course: High Performance Computing Lab

Practical 6

Exam Seat No:2018BTECS00005

Name : Rahul Sanjay Naravadkar

Problem Statement 1 : MPI Scatter and Gather

MPI scatter:

MPI_Scatter is a collective routine that is very similar to MPI_Bcast. MPI_Scatter involves a designated root process sending data to all processes in a communicator. The primary difference between MPI_Bcast and MPI_Scatter is small but important. MPI_Bcast sends the same piece of data to all processes while MPI_Scatter sends chunks of an array to different processes.

Here is what the function prototype of MPI_Scatter looks like.

```
MPI_Scatter(  
    void* send_data,int  
    send_count,  
    MPI_Datatype send_datatype,  
    void* recv_data,  
    int  recv_count, MPI_Datatype  
    recv_datatype,int root,  
    MPI_Comm communicator  
)
```

MPI gather:

MPI_Gather is the inverse of MPI_Scatter. Instead of spreading elements from one process to many processes, MPI_Gather takes elements from many processes and gather them to one single process. This routine is highly useful to many parallel algorithms, such as parallel sorting and searching. Similar to MPI_Scatter, MPI_Gather takes elements from each process and gathers them to the root process. The elements are ordered by the rank of the process from which they were received. The function prototype for MPI_Gather is identical to that of MPI_Scatter.

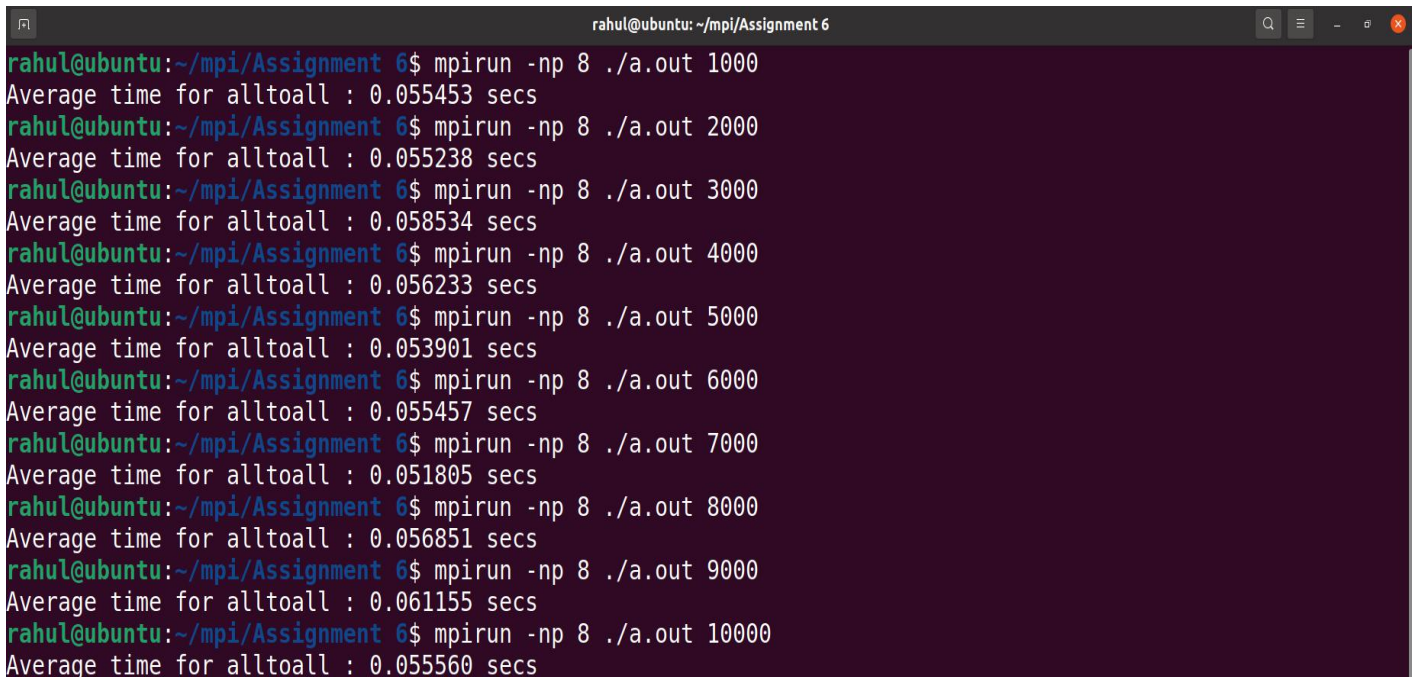
```
MPI_Gather(  
    void* send_data,int  
    send_count,  
    MPI_Datatype send_datatype,  
    void* recv_data,  
    int  recv_count,
```

```
MPI_Datatype recv_datatype,  
int root,  
MPI_Comm communicator  
)
```

Problem Statement 2 : 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. Explain the performance observed.

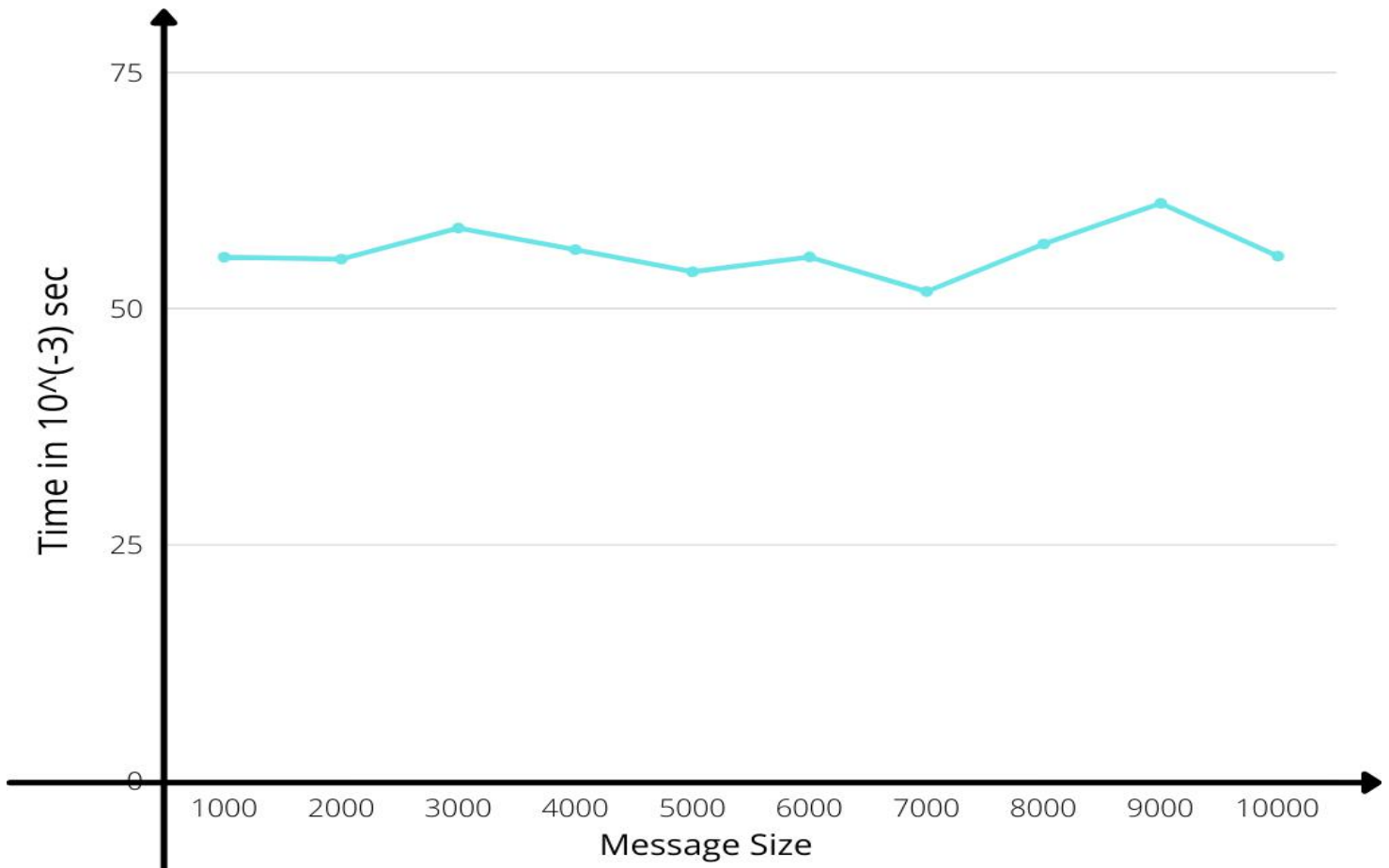
Screenshot :



The screenshot shows a terminal window titled 'rahul@ubuntu: ~/mpi/Assignment 6'. It displays a series of commands and their outputs for an all-to-all broadcast operation using 8 processes and varying message sizes from 1000 to 10000. The average time for alltoall is consistently around 0.055 seconds across all message sizes.

```
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 8 ./a.out 1000  
Average time for alltoall : 0.055453 secs  
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 8 ./a.out 2000  
Average time for alltoall : 0.055238 secs  
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 8 ./a.out 3000  
Average time for alltoall : 0.058534 secs  
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 8 ./a.out 4000  
Average time for alltoall : 0.056233 secs  
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 8 ./a.out 5000  
Average time for alltoall : 0.053901 secs  
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 8 ./a.out 6000  
Average time for alltoall : 0.055457 secs  
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 8 ./a.out 7000  
Average time for alltoall : 0.051805 secs  
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 8 ./a.out 8000  
Average time for alltoall : 0.056851 secs  
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 8 ./a.out 9000  
Average time for alltoall : 0.061155 secs  
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 8 ./a.out 10000  
Average time for alltoall : 0.055560 secs
```

Graph Analysis:



Information : Average time for broadcast increases as size of message increases.

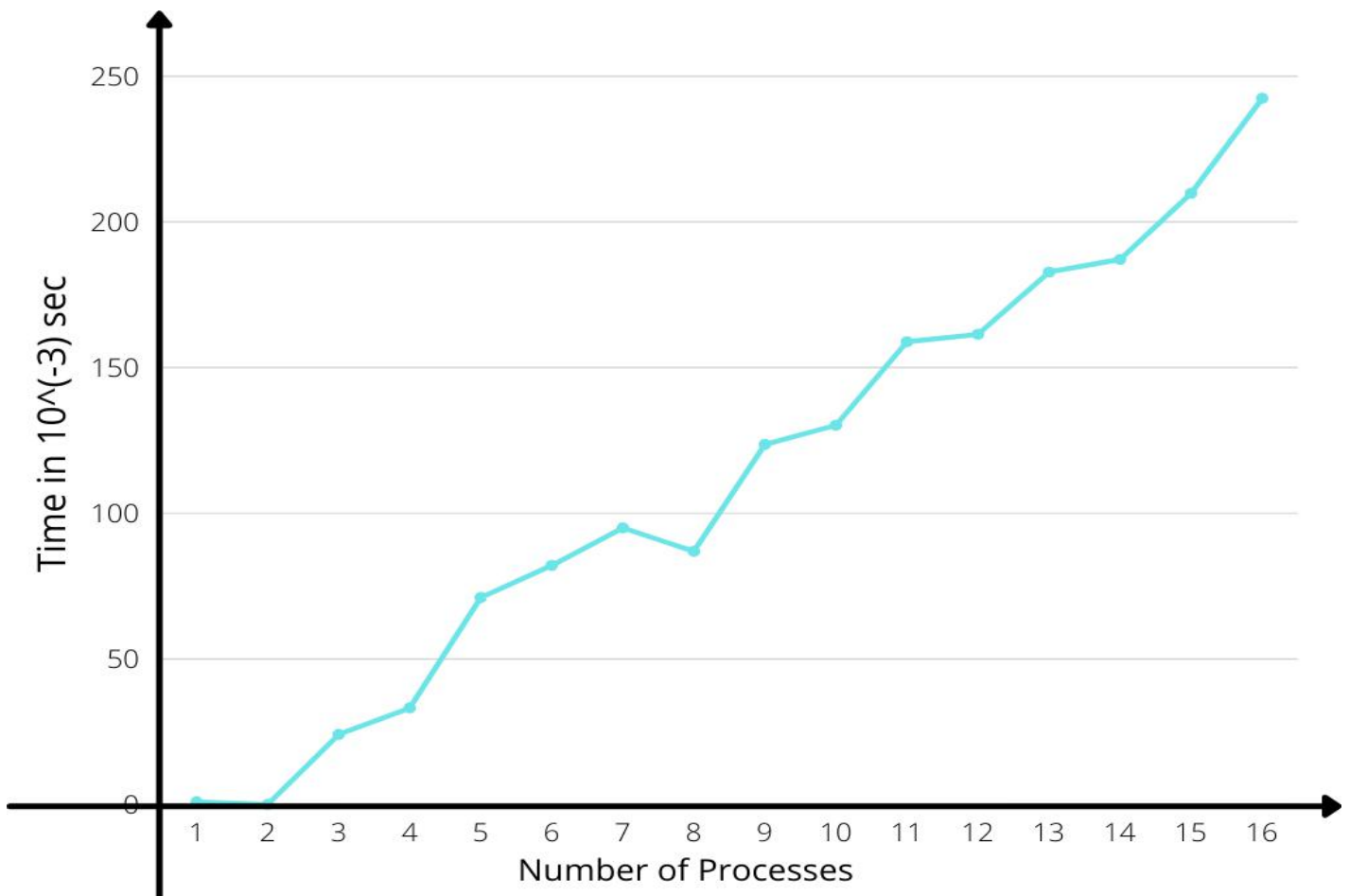
Problem Statement 3 : Execute the all-reduce operation (Program D) with varying number of processes (1 to 16) and fixed message size of 10K words. Plot the performance of the operation with varying number of processes (with constant message size). Explain the performance observed.

Screenshot :

```
rahul@ubuntu: ~/mpi/Assignment 6
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 1 ./Question2.o 10000
Average time for allreduce : 0.000944 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 2 ./Question2.o 10000
Average time for allreduce : 0.000036 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 3 ./Question2.o 10000
Average time for allreduce : 0.024046 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 4 ./Question2.o 10000
Average time for allreduce : 0.033142 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 5 ./Question2.o 10000
Average time for allreduce : 0.071064 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 6 ./Question2.o 10000
Average time for allreduce : 0.082083 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 7 ./Question2.o 10000
Average time for allreduce : 0.094967 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 8 ./Question2.o 10000
Average time for allreduce : 0.086902 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 9 ./Question2.o 10000
Average time for allreduce : 0.124630 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 10 ./Question2.o 10000
Average time for allreduce : 0.130191 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 11 ./Question2.o 10000
Average time for allreduce : 0.158845 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 12 ./Question2.o 10000
Average time for allreduce : 0.161418 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 13 ./Question2.o 10000
Average time for allreduce : 0.182764 secs
rahul@ubuntu:~/mpi/Assignment 6$
```

```
rahul@ubuntu:~/mpi/Assignment 6$ cd mpi/Assignment\ 6
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 14 ./Question2.o 10000
Average time for allreduce : 0.187115 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 15 ./Question2.o 10000
Average time for allreduce : 0.209823 secs
rahul@ubuntu:~/mpi/Assignment 6$ mpirun -np 16 ./Question2.o 10000
Average time for allreduce : 0.242382 secs
rahul@ubuntu:~/mpi/Assignment 6$
```

Graph Analysis:



Information : Average time for all-reduce increases as number of processes increases with fixed message size.

[Github Link](#)