Parallel Programming Lab 6: MPI_Reduce B

Name: Raghav Maheshwari Batch: A4
Roll no: PA53 Panel: A

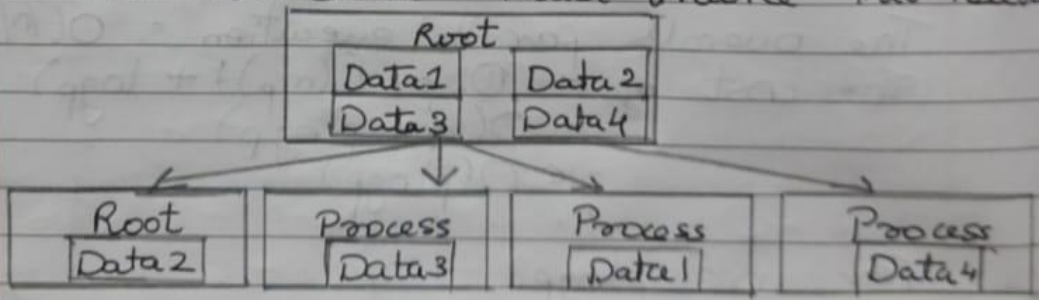
Aim: To understand the behavior of scaling the system.

Objective: To understand the behavior of MPI_REDUCE.

Theory:

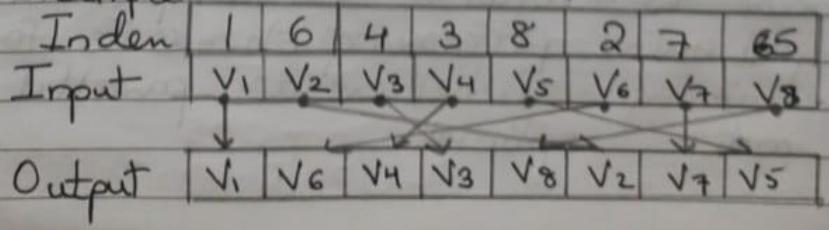
Theory.

The scatter operation is mapping data from the root to all the connected nodes. The scatter operation sends/maps ~~all if~~ a unique collection of data to processes. This operation can be performed by the programmer using MPI_Scatter. The MPI_Scatter ~~button~~ dispatches data from a process across all processes in the same communicator. As a blocking operation the buffer passed can be safety reused as soon as the routine returns. MPI_Scatter is a collective operation, all the processes in the communicator must invoke this routine.



In essence the scatter operation is a series of random read and sequential write operation.

Example 2



Code:

```c
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    // Initialize the MPI environment
    MPI_Init(&argc, &argv);
    int arr[]={1,2,3,4};
    // Get the number of processes
    int myrank,sum,min,max,x,i=0;
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);


    MPI_Scatter(arr,1,MPI_INT,&x,1,MPI_INT, 0,MPI_COMM_WORLD);

    for(i=0;i<4;i++){

        if(myrank==i){
            printf("Value on Scattered =%d core is %d\n",myrank,x);
        }


    }
         MPI_Reduce(&x,&min,1,MPI_INT,MPI_MIN,1,MPI_COMM_WORLD);
    if(myrank==1){
            printf("Min=%d on The core is %d\n",min,myrank);
        }
         MPI_Reduce(&x,&max,1,MPI_INT,MPI_MAX,2,MPI_COMM_WORLD);
        if(myrank==2){
            printf("Max=%d on The core is %d\n",max,myrank);
        }

    // Finalize the MPI environment.
    MPI_Finalize();
    return 0;
}
```

Result:

```
keyuroak@keyuroak-VirtualBox:~/Desktop/PPMPI$ mpicc -pg -o A test.c
keyuroak@keyuroak-VirtualBox:~/Desktop/PPMPI$ mpirun A
Value on Scattered =0 core is 1
Value on Scattered =1 core is 2
Value on Scattered =2 core is 3
Value on Scattered =3 core is 4
Min=1 on The core is 1
Max=4 on The core is 2
keyuroak@keyuroak-VirtualBox:~/Desktop/PPMPI$ gprof A gmon.out > profile.txt
keyuroak@keyuroak-VirtualBox:~/Desktop/PPMPI$ gedit profile.txt
```

```
1 Flat profile:
2
3 Each sample counts as 0.01 seconds.
4  no time accumulated
5
6   %   cumulative   self              self     total
7  time   seconds   seconds    calls  Ts/call  Ts/call  name
8
```

Conclusion: The minimum and maximum number is calculated on different processes simultaneously

FAQ's:

1. Explain the working of MPI_SCATTER and MPI_Broadcast()

Ans: MPI_Bcast() sends the same piece of data to everyone, while MPI_Scatter() sends each process a part of the input array. MPI_Bcast() is the opposite of MPI_Reduce() and MPI_Scatter() is the opposite of MPI_Gather(). A little scheme like this one is self-explanatory. And
both MPI_Scatter() and MPI_Bcast() have an argument named int root to specify the root process.

2. Explain the difference between MPI and Openmp
Ans: OpenMP is a way to program on shared memory devices. This means that the parallelism occurs where every parallel thread has access to all of your data. It as parallelism can happen during execution of a specific for loop by splitting up the loop among the different threads.

MPI is a way to program on distributed memory devices. This means that the parallelism occurs where every parallel process is working in its own memory space in isolation from the others.