Fall Sem 2021-22

Assignment: 9

Date: 30/11/21

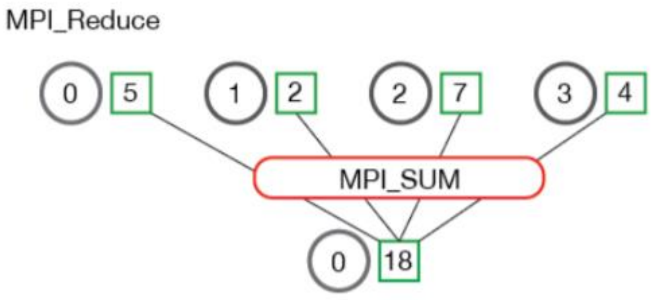
Name: Ishan Sagar Jogalekar

Reg no: 19BCE2250

Course: Parallel and distributed computing LAB - CSE4001

Slot: L55+L56

**Aim**: Write a C program to use MPI_Reduce that divides the processors into the group to find the addition independently.



**Ans –**

**SOURCE CODE:**

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

int main(int argc, char** argv) {
   //19BCE2250- Ishan Jogalekar

   int rank, size;
   //Intializing MPI
   MPI_Init(&argc, &argv);
   //Fidning rank and size of process
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);

   //Array , it's size and local sum , global sum declaration
   int glob_Sum, loc_sum = 0;
   int arr[] = { 5, 2, 7, 4 };
```

```
    int n = 4;
    // If intializing with 0th process rank then print working
    if (rank == 0)
    {
        printf("Addition of elements in array....\n");
    }
    //Addition of elements in array
    for (int i = rank; i < n;i += size)
    {
        printf("arr[%d]: %d\n",i, arr[i]);
        loc_sum += arr[i];
    }
    //MPI reduce to add locala sum in global sum
    MPI_Reduce(&loc_sum, &glob_Sum, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

    //If rank is 0 print Final sum
    if (rank == 0)
    {
        printf("Final Sum = %d\n", glob_Sum);
    }
    MPI_Finalize();
    return 0 ;
}
```

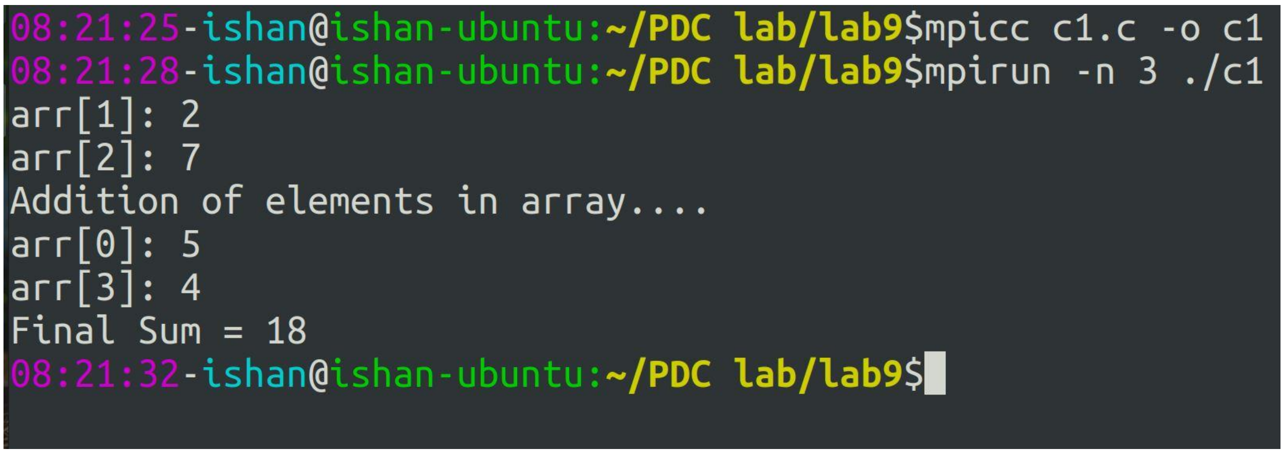
**EXECUTION:**

1. OpenMP is a library for parallel programming in the SMP (symmetric multi-processors or shared-memory processors) model.
2. Mpi.h is header file to use all mpi functions inside program.
3. MPI_COMM_RANK is used to determine process identifier, that processes id number.
4. MPI_COMM_SIZE is used to determine total size of process pool inside that particular MPI program. Also, deceleration of global sum and local sum to add sum of elements by sub-process and finally add it to global sum.
5. Using for loop to iterate from current rank of process till complete size of process pool, in order to add array elements in local sum.
6. MPI_Reduce function is to apply a reduction calculation with MPI_Sum function on MPI process. The values sent by the MPI processes will be combined using the reduction operation given and the result will be stored on the MPI process specified as root.

7. Using if statement at root process printing Global sum that is final sum of elements in array.
8. Using MPI_Finalize function ending the MPI interface within program.

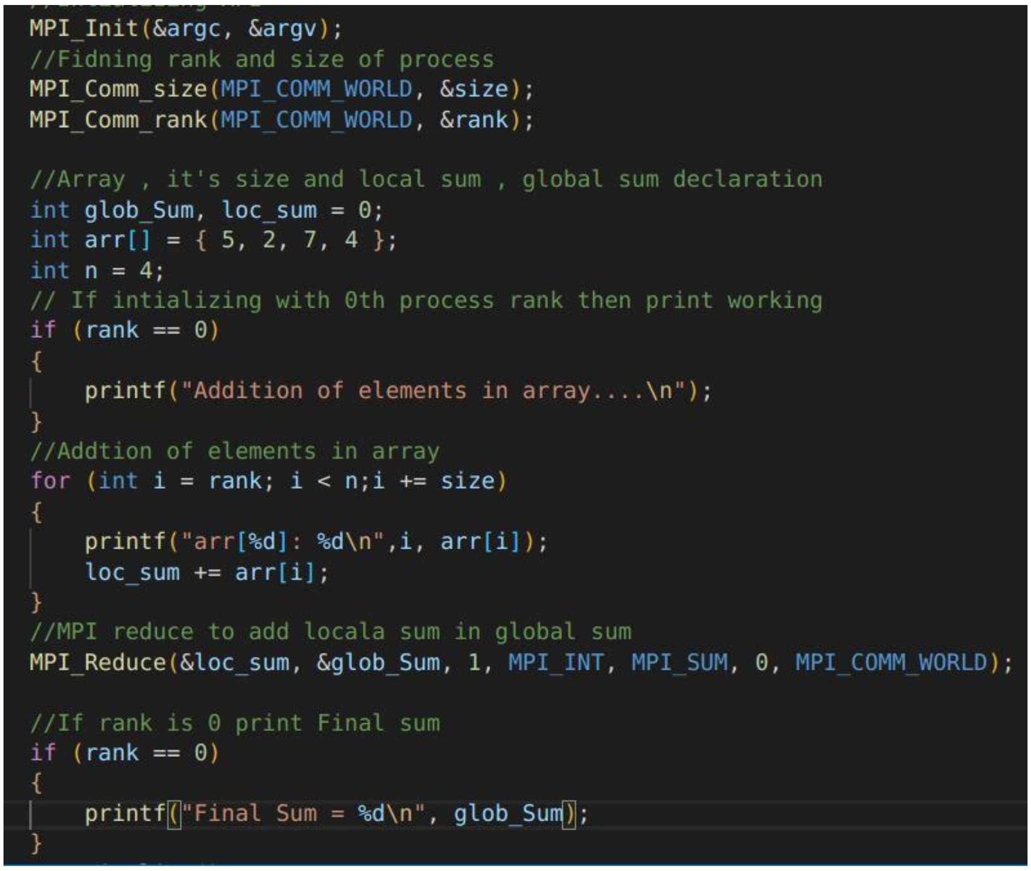
**RESULTS**:

Output:



```
08:21:25-ishan@ishan-ubuntu:~/PDC lab/lab9$mpicc c1.c -o c1
08:21:28-ishan@ishan-ubuntu:~/PDC lab/lab9$mpirun -n 3 ./c1
arr[1]: 2
arr[2]: 7
Addition of elements in array....
arr[0]: 5
arr[3]: 4
Final Sum = 18
08:21:32-ishan@ishan-ubuntu:~/PDC lab/lab9$
```

Code:



```c
MPI_Init(&argc, &argv);
//Fidning rank and size of process
MPI_Comm_size(MPI_COMM_WORLD, &size);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);

//Array , it's size and local sum , global sum declaration
int glob_Sum, loc_sum = 0;
int arr[] = { 5, 2, 7, 4 };
int n = 4;
// If intializing with 0th process rank then print working
if (rank == 0)
{
    printf("Addition of elements in array....\n");
}
//Addtion of elements in array
for (int i = rank; i < n;i += size)
{
    printf("arr[%d]: %d\n",i, arr[i]);
    loc_sum += arr[i];
}
//MPI reduce to add locala sum in global sum
MPI_Reduce(&loc_sum, &glob_Sum, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

//If rank is 0 print Final sum
if (rank == 0)
{
    printf("Final Sum = %d\n", glob_Sum);
}
```