UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

DISTRIBUTE AND CLOUD SYSTEMS PROGRAMMING (5CS022)

WEEK 2 WORKSHOP

Student Id : 2065697
Student Name : Dhiraj Kumar Sah Kanu
Group : L5CG12

Submitted on : March 14, 2022

5CS022 Distribute and Cloud Systems Programming Week 2 Workshop

Tasks

1. The following C program sums up all the values in array "data" and displays the sum total.:

```
#include <stdio.h>
#define
NUMDATA 10000
int
data[NUMDATA];

void LoadData(int data[])
{
  for(int i = 0; i <
    NUMDATA; i++){ data[i] =
    1;
  }
}

int AddUp(int data[], int count)
{
  int sum = 0;
  for(int i = 0; i <
    count; i++){ sum +=
    data[i];
  }
  return sum;
}

int
  main(vo
  id) {
  int
  sum;

  LoadData(data);
  sum = AddUp(data, NUMDATA);
  printf("The total sum of data is
  %d\n", sum); return 0;
}
```

Convert it to MPI to run with any number of nodes including just one.

File Edit Selection View Go Run Terminal Help qno1.c - Week 2 - Visual Studio Code

Get Started | qno1.c 2

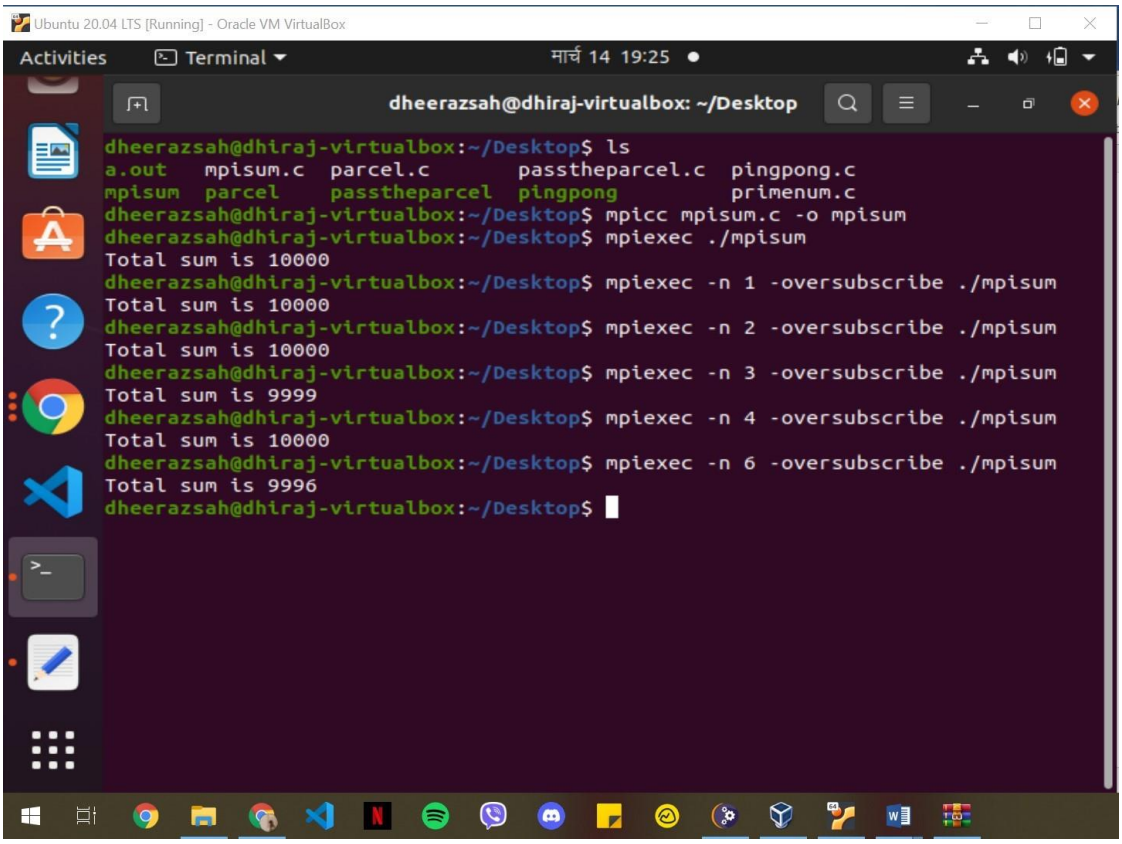
qno1.c > main(void)

```c
#include <stdio.h>
#include <mpi.h>
#define NUMDATA 10000
int data[NUMDATA];
void LoadData(int data[])
{
  for(int i = 0; i < NUMDATA; i++){
    data[i] = 1;}
}
int AddUp(int data[], int count)
{
  int sum = 0;
  for(int i = 0; i < count; i++){
    sum += data[i];}
  return sum;
}
int main(void) {
  int sum;
  int size;
  int rank;
  int tag =0;
  int chunksize;
  int start;
  int result;

  MPI_Init(NULL, NULL);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  chunksize = NUMDATA / size;
  if (rank == 0){
    LoadData(data);
    for(int i =1; i < size; i++){
      start = i * chunksize;

      MPI_Send(&(data[start]), chunksize, MPI_INT, i, tag, MPI_COMM_WORLD);
    }
    sum = AddUp(data, chunksize);
    for(int i = 1; i < size; i++) {
      MPI_Recv(&result, 1, MPI_INT, MPI_ANY_SOURCE, tag, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
      sum += result;
    }
    printf("Total sum is %d\n", sum);
  }

  else{
    MPI_Recv(data, chunksize, MPI_INT, 0, tag, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    sum = AddUp(data, chunksize);
    MPI_Send(&sum, 1, MPI_INT, 0, tag, MPI_COMM_WORLD);
  }
  MPI_Finalize();
  return 0;
}
```



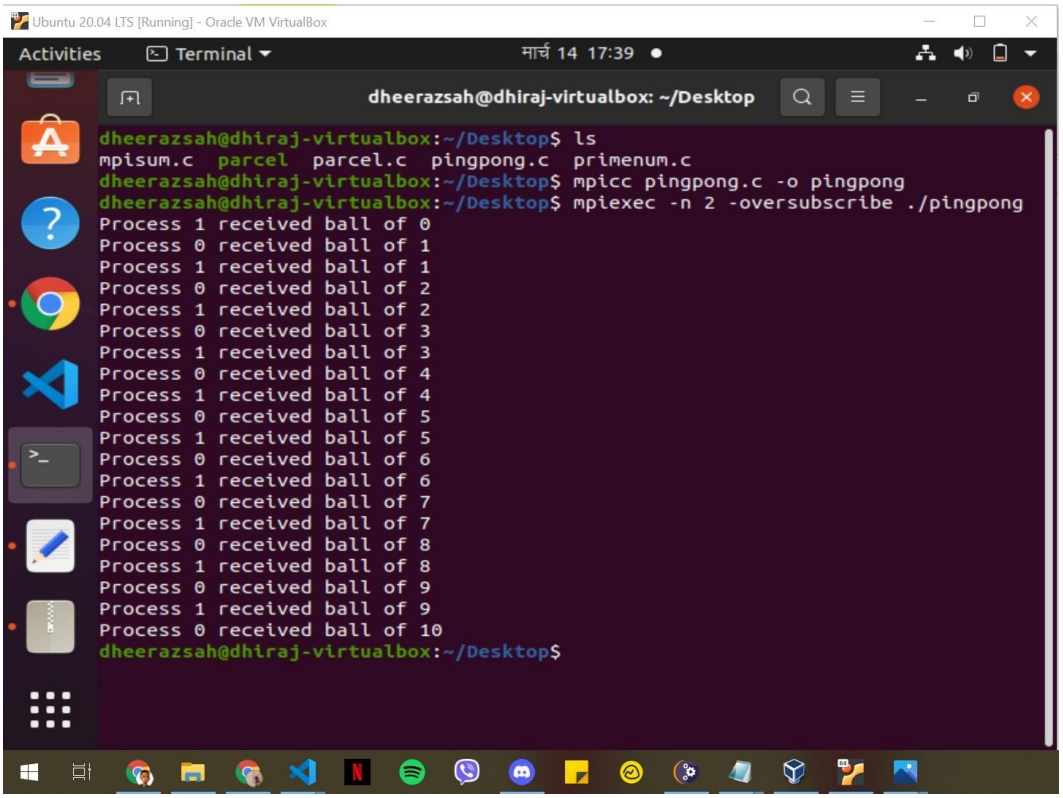
In the above screenshot, the program is executed using different processes. At first, saved mpisum.c file is searched and then selected, after that mpisum.c is compiled. While executing using mpiexec ./mpisum, the program is compiled and gives the result. The program is also executed which gives total sum is 1000 when two or four processes are used. But when 3 or 6 processes are used the total sum are not 1000, it gives different value.

2. Write an MPI program called pingpong.c to run with exactly 2 process. Process rank 0 is to send an integer variable call "ball" initialised with the value zero to Process rank 1. Process rank 1 will add 1 to the ball and send it back. This will repeat until the ball has a value of 10 in Process rank 0.

```c
#include <stdio.h>
#include <mpi.h>
int main(void){
    int sum;
    int size;
    int rank;
    int tag =0;
    int ball = 0;
    MPI_Init(NULL, NULL);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (size != 2){
        printf("Needs two process!\n");
        MPI_Finalize();
        return 0;
    }
    while(1){
        if(rank ==0){
            MPI_Send(&ball, 1, MPI_INT, 1, tag, MPI_COMM_WORLD);
            MPI_Recv(&ball, 1, MPI_INT, 1, tag, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            printf("Process 0 received ball of %d\n", ball);
            if(ball == 10){
                break;
            }
        }
        else{
            MPI_Recv(&ball, 1,MPI_INT, 0, tag, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            printf("Process 1 received ball of %d\n", ball);
            ball++;
            MPI_Send(&ball, 1, MPI_INT, 0, tag, MPI_COMM_WORLD);
            if(ball == 10){
                break;
            }
        }
    MPI_Finalize();
    return 0;
}
```

```
dheerazsah@dhiraj-virtualbox:~/Desktop$ ls
mpisum.c  parcel  parcel.c  pingpong  pingpong.c  primenum.c
dheerazsah@dhiraj-virtualbox:~/Desktop$ mpicc pingpong.c -o pingpong
dheerazsah@dhiraj-virtualbox:~/Desktop$ mpiexec ./pingpong
Needs two process!
dheerazsah@dhiraj-virtualbox:~/Desktop$
```

In the above screenshot, the program is executed with the available processes. The program shows it needs two processes to compile the program.



In the above screenshot, the program is executed using 2 processes. At first, saved pingpong.c file is searched and then selected, then is compiled. This shows how many times the ball was initialized and received by different processes, it works until the ball has a value of 10.

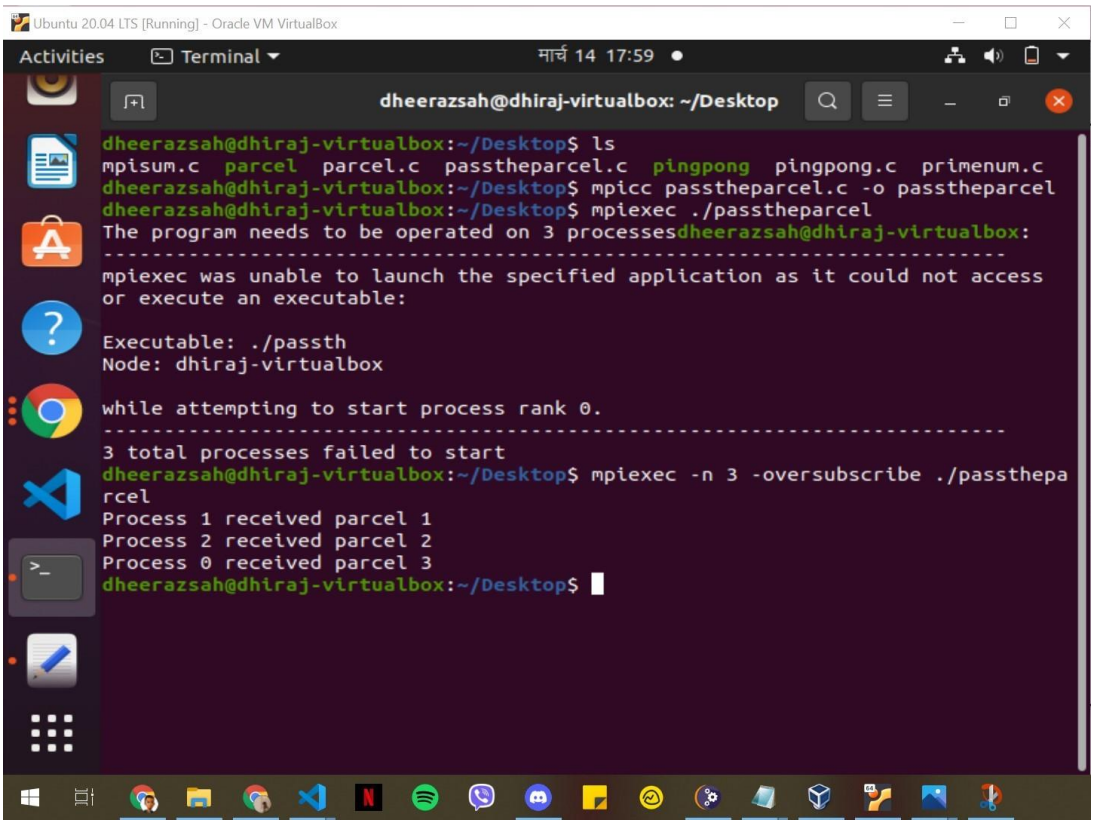
3. Write a "Pass-the-parcel" MPI program that will run with 3 or more nodes, such that Process rank 0 will send an integer variable call "parcel" initialised with 1, to Process rank 1 which will add 1 to the parcel and then send it to Process rank 2, and so on until the highest rank process will send it back to Process rank 0, at which point the parcel variable should contain the value of the number of nodes there are.

qno3final.c - Week 2 - Visual Studio Code

qno1.c | qno2 | qno3 2 | qno3final.c 2

qno3final.c > ...

```c
#include <stdio.h>
#include <mpi.h>
int main(int argc, char **argv)
{ int i, c;
  int size,rank;
  int ball = 0;
  MPI_Init(NULL,NULL);
  MPI_Comm_size(MPI_COMM_WORLD,&size);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  int sender , receiver;
  int parcel = 1;
  if(size != 3){
  printf("The program needs to be operated on 3 processes"); }
  else{
if(rank ==0){
sender = size-1;}
  else{
  sender = rank-1;}
  if (rank == size -1){
  receiver = 0;}
  else{
  receiver = rank +1;}
  if (rank == 0){
  MPI_Send(&parcel ,1,MPI_INT,(rank+1),0,MPI_COMM_WORLD);
  MPI_Recv(&parcel,1,MPI_INT,(rank-1),0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
  printf("Process 0 received parcel %d \n",parcel);
  }
  else{
  MPI_Recv(&parcel,1,MPI_INT,(rank-1),0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
  printf("Process %d received parcel %d\n",rank,parcel);
  parcel++;
  MPI_Send(&parcel,1,MPI_INT,receiver,0,MPI_COMM_WORLD);
  } }
  MPI_Finalize();
  return 0;
  }
```



In the above screenshot, the program is executed using available processes due to which it shows error saying the program can only operate on 3 processes. Then the program is executed using 3 processes, which shows the result where parcel are received.