WARSAW UNIVERSITY OF TECHNOLOGY

FACULTY OF MATHEMATICS AND

INFORMATION SCIENCE

FIRST SEMESTER 2021/2022

HIGH PERFORMANCE COMPUTING (HPC)

PROJECT 2

BY:

ANDRA UMORU (324334)

SUBMITTED TO:

OKULICKA-DLUZEWSKA FELICJA

TABLE OF CONTENT

PROJECT TWO

1.1 Project description

Create the virtual topology with at least 8 nodes and for each process find its neighbor's:

Student 12: Cartesian Size 4 x 4 non-periodic.

1.2 Overview of Virtual Topology

Virtual topology can allow MPI to optimize communications by creating scheme fitting the communication pattern. there may be no relation between the physical structure of the parallel machine and the process topology. The creating a topology produces a new communicator with new interior ranks. Topology types are graph topologies and Cartesian topologies.

1.3 MPI Cartesian Topology

Each process is 'connected' to its neighbors in a virtual grid. Boundaries can be cyclic , or not. Processes are identified by the Cartesian coordinates.

To create a cartesian topology, the following is the function that does that:

int MPI_Cart_create (MPI_Comm comm_old,
int ndims,
int *dims,
int *period,
int reorder,
MPI_Comm *comm_cart)

For my project two, ndims = 2

dims = [4, 4]

per [0, 0] – non – periodic

1.3.1 MPI Cartesian Rank

Mapping process grid coordinates to ranks, it has the following syntax:

int MPI_Cart_rank (MPI_Comm comm,
int *coords,
int *rank)

1.3.2 MPI Cartesian Shift

This cartesian functionality is used in computing ranks of neighboring processes. It has the following syntax in C programming language:

int MPI_Cart_shift (MPI_Comm comm,

int direction,

int disp,

int *rank_source,

int *rank_dest)

1.4 The Source Code

The source code was written in C programming language. Below is the source code for project two:

```
#include <stdio.h>     //including the C standard library
#include <mpi.h>      //Including the MPI library

// Cartesian topology 4 x 4
//  0 – 1 – 2  – 3
//  |   |   |    |
//  4 – 5 – 6  – 7
//  |   |   |    |
//  8 – 9 – 10 – 11
//  |   |   |    |
// 12 – 13 –14 – 15

// C Language main function
int main(int argc, char **argv)
{

  //Declaring the size and the rank variables
 int size,rank;

     //Initializing the MPI Environment
 MPI_Init(&argc,&argv);

     // Getting number of process
 MPI_Comm_size(MPI_COMM_WORLD,&size);

     // Getting rank of the process
 MPI_Comm_rank(MPI_COMM_WORLD,&rank);

 //Getting the number of dimensions
```

```
int dim [2]={4,4};

//Getting the period --> [0, 0] means non - periodic
int per [2]={0,0};
MPI_Comm com;

// Creating the cartesian topology
MPI_Cart_create(MPI_COMM_WORLD,2,dim,per,1,&com);

//Mapping the neighbours in the cartesian topology
int cord [2]={0,0};
int crank;
MPI_Cart_coords(com,rank,2,cord);

// Printing the cartesian topology together with the mappings
printf("I am %d and my coordinates are (%d,%d)\n", rank, cord[0],cord[1]);
printf(" my neighbours are:\n");
int c,d,neighbour;
for(c=0;c<2;c++)
{
for(d=-1;d<2;d++)
{ // shift left and right by 1 step (d=-1,d=1)
if (d!=0)
{
MPI_Cart_shift(com,c,d,&rank,&neighbour);
if (neighbour>=0)
printf(" neighbour [%d,%d] has rank %d\n",c,d,neighbour);
}
}
}

// Exiting or Terminating the cartesian MPI routine
MPI_Finalize();

return 0;
}
```

1.5 Result (Output)

After compiling and running the MPI cartesian program, the figures below displays is the output:

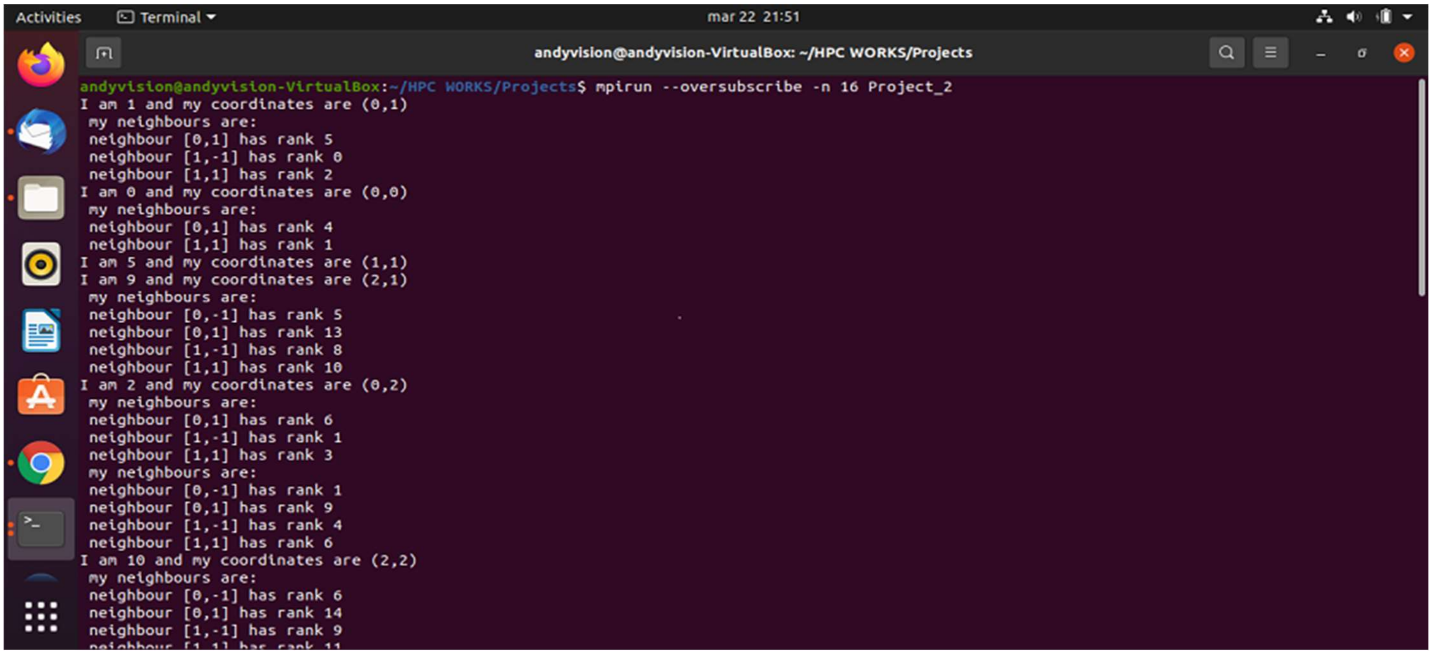

Figure 2a: The result

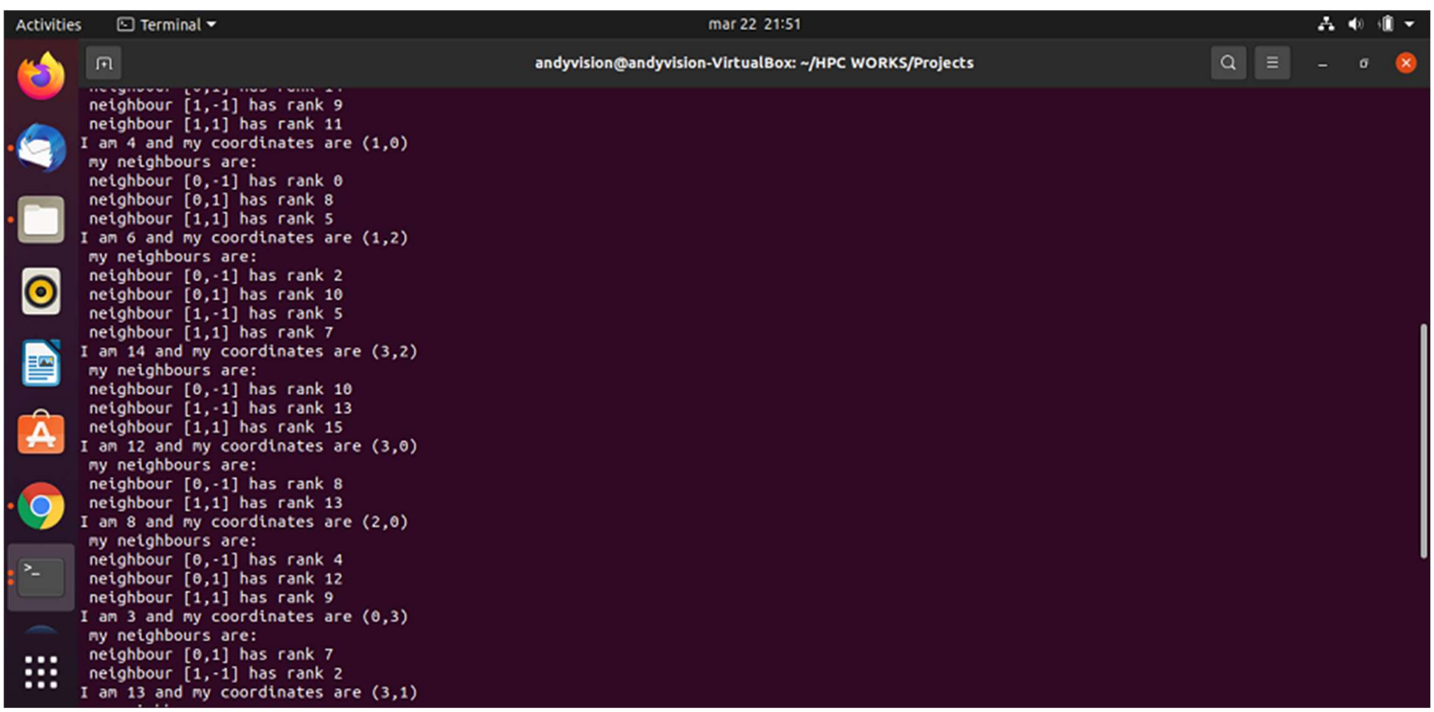

Figure 1b: The result

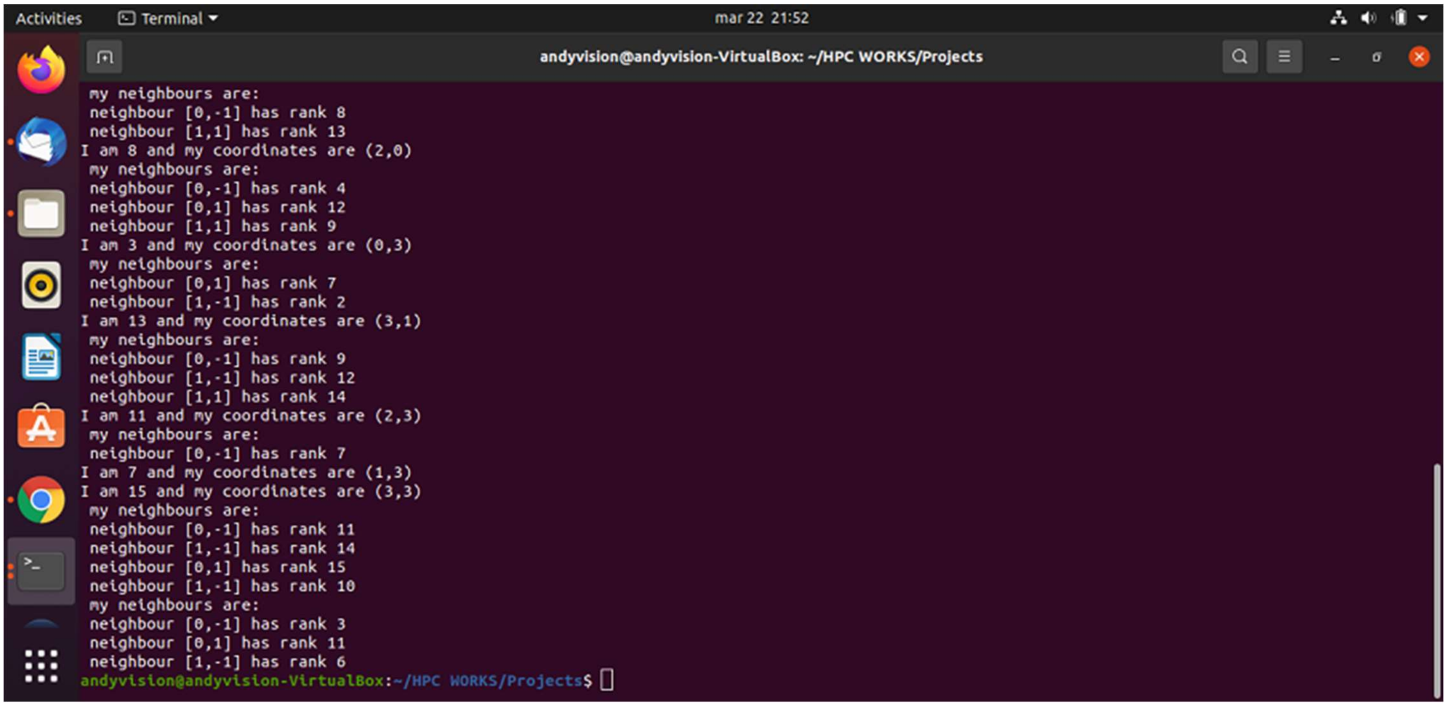

Figure 3c: The result

1.6 Reference

- HPC Lecture Notes