COMP 6xx (Fall 2021): project 1
Name: Hossein Alishah
Student ID: 202048619
Section: Wed (7:00 pm – 9:45 pm)

Project 1. Design and implementation a parallel search algorithm using MPI

An array of given size is to be searched for occurrences of a specific target value. The size and type of array and the target value should be considered optional.

Answer: I used C programing language:
In this project I am going do the following steps:
Linear search using MPI:
1. initialize MPI environment
2. Get the number of processes
3. Get the ranks processes (Root process has Rank=0)

Root process task:
a) divide elements of input array into parts and send elements into each part to the remaining processes and assign one part of the array to itself.
b) call binary search function on the part of the array assign to itself.

Slaves process task:
a) Receive individual part of array from root process.
b) call binary search function the receive part of array.

I used 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. MPI_Scatter sends chunks of an array to different processes.

MPI_Scatter takes an array of elements and distributes the elements in the order of process rank. The first element goes to process zero, the second element goes to process one, and so on. Although the root process (process zero) contains the entire array of data, MPI_Scatter will copy the appropriate element into the receiving buffer of the process.

I put the array size (n= ...) in the code and there is an array generator which can generate array in the size of 0 to n-1.
In what follows, you can see the program codes:

```c
//Hossein Alishah
//Comp 620 - Fall 2021
//Project 1. Design and implementation a parallel search algorithm using MPI


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


int main(int argc, char **argv){
    int n=100;

    //impleneting a sample array with the n size
    int arr[n], b[n];
    for(int i=0;i<n;i++)
        {
            arr[i]=i+1;
        }


    //element to search
    int key=43;

    //initialize MPI environment
    MPI_Init(&argc, &argv);

    int rank,size;

    //Get the number of processes
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    // Get the ranks processes
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    MPI_Scatter(&arr,n/2,MPI_INT,&b,n/2,MPI_INT,0,MPI_COMM_WORLD);

```

```
    if(rank==0)
        {
            double start=MPI_Wtime();
            int index1 = 0;
            for(int i=0; i<n/2; i++){
                if(b[i]==key){
                    index1 = i;
                }
            }
            //printing elements of array
            if (index1 != 0){
                printf("\nthe key is %d\n",key);
                printf("\nthe size of array is %d\n\n",n);
                printf("\nNumber found on processor:  %d\t on the position:  %d\n\n",rank,index1);
                printf("Array is\n");
               for( int i =0; i<n; i++){
                   printf("%d ",arr[i]);
                   printf(",");}
            }
            double end=MPI_Wtime();
            printf("\nExecution time of Processor %d is %f\n\n", rank, (end-start)*1000);
            }
    else if(rank==1)
        {
            double start=MPI_Wtime();
            int index = 0;
            for(int i=0; i<n/2; i++){
                if(b[i]==key){
                    index = i+n/2;}
            }
            //printing elements of array
            if (index != 0){
                printf("\nthe key is %d\n",key);
                printf("\nthe size of array is %d\n\n",n);
                printf("\nNumber found on processor:  %d\t on the position:  %d\n\n",rank,index);
                printf("Array is\n");
               for( int i =0; i<n; i++){
                   printf("%d ",arr[i]);
                   printf(",");
               }
                printf("\n\n");
            }

            double end=MPI_Wtime();
            printf("\nExecution time of Processor %d is %f\n\n", rank, (end-start)*1000);
        }
    }
```

For the first try, I put the array size of 100, and the key= 43, and the output is going to be:

```
hosseinalishah@Hosseins-MacBook-Pro ~ % mpicc search2.c
hosseinalishah@Hosseins-MacBook-Pro ~ % mpirun -np 2 ./a.out

the key is 43

the size of array is 100


Number found on processor:  0    on the position:  42

Array is
1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,19 ,20 ,21 ,22 ,2
3 ,24 ,25 ,26 ,27 ,28 ,29 ,30 ,31 ,32 ,33 ,34 ,35 ,36 ,37 ,38 ,39 ,40 ,41 ,42 ,4
3 ,44 ,45 ,46 ,47 ,48 ,49 ,50 ,51 ,52 ,53 ,54 ,55 ,56 ,57 ,58 ,59 ,60 ,61 ,62 ,6
3 ,64 ,65 ,66 ,67 ,68 ,69 ,70 ,71 ,72 ,73 ,74 ,75 ,76 ,77 ,78 ,79 ,80 ,81 ,82 ,8
3 ,84 ,85 ,86 ,87 ,88 ,89 ,90 ,91 ,92 ,93 ,94 ,95 ,96 ,97 ,98 ,99 ,100 ,
Execution time of Processor 0 is 0.090000


Execution time of Processor 1 is 0.003000
```

Let's try for another n= 3450 and the key = 2931:

```
hosseinalishah@Hosseins-MacBook-Pro ~ % mpicc search2.c
hosseinalishah@Hosseins-MacBook-Pro ~ % mpirun -np 2 ./a.out

Execution time of Processor 0 is 0.012000


the key is 2931

the size of array is 3450


Number found on processor:  1    on the position:  2930

Array is
1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,19 ,20 ,21 ,22 ,2
3 ,24 ,25 ,26 ,27 ,28 ,29 ,30 ,31 ,32 ,33 ,34 ,35 ,36 ,37 ,38 ,39 ,40 ,41 ,42 ,4
3 ,44 ,45 ,46 ,47 ,48 ,49 ,50 ,51 ,52 ,53 ,54 ,55 ,56 ,57 ,58 ,59 ,60 ,61 ,62 ,6
3 ,64 ,65 ,66 ,67 ,68 ,69 ,70 ,71 ,72 ,73 ,74 ,75 ,76 ,77 ,78 ,79 ,80 ,81 ,82 ,8
3 ,84 ,85 ,86 ,87 ,88 ,89 ,90 ,91 ,92 ,93 ,94 ,95 ,96 ,97 ,98 ,99 ,100 ,101 ,102
 ,103 ,104 ,105 ,106 ,107 ,108 ,109 ,110 ,111 ,112 ,113 ,114 ,115 ,116 ,117 ,118
 ,119 ,120 ,121 ,122 ,123 ,124 ,125 ,126 ,127 ,128 ,129 ,130 ,131 ,132 ,133 ,134
```

Finally, I am going to run it for a big size of array n=123,456 to search for key=99,000, I avoid printing the array because of the big size of data:

```
[hosseinalishah@Hosseins-MacBook-Pro ~ % mpicc search2.c
[hosseinalishah@Hosseins-MacBook-Pro ~ % mpirun -np 2 ./a.out

Execution time of Processor 0 is 0.366000


the key is 99000

the size of array is 123456


Number found on processor:  1    on the position:  98999




Execution time of Processor 1 is 0.398000
```

I have attached the source code file to the zip file as well.