Networked Software for Distributed Systems – Project 3 Documentation
Group 7 – De Novellis Torralba Venkataraman

Problem Statement

The objective of this project is to implement a program that simulates how a virus spreads over time in a population of individuals. The program considers N individuals that move in a rectangular area with linear motion and velocity v (each individual following a different direction). Some individuals are initially infected. If an individual remains close to (at least one) infected individual for more than 10 minutes, it becomes infected. After 10 days, an infected individual recovers and becomes immune. Immune individuals do not become infected and do not infect others. An immune individual becomes susceptible again (i.e., it can be infected) after 3 months.

Implementation Logic

Based on the conditions provided in the project, the world was split into countries with individuals randomly assigned to each country. It is assumed that, when an individual reaches the border of the world, a "rebound" policy is adopted to make sure that the individual remains in it, for example if an individual reaches the right border the direction will change to left.

Country:

The overall area is split into smaller rectangular sub-areas representing countries. The program outputs, at the end of each simulated day, the overall number of susceptible, infected, and immune individuals in each country.

```
int country_id;
int x_start, x_end, y_start, y_end;
```

Individual:

An individual belongs to a country if it is in that country at the end of the day. Each individual starts with a position and direction of movement that is randomly assigned during initialisation. These values will be updated by the program to know the position of the individual during the run of the simulation.

```
int id;
int country;
int x_pos;
int y_pos;
int time_status;
char status;
char direction;
```

Program Flow

- **Initialising data:** The program first start by initializing the data to be used by each process. Each process accepts the input provided by the user and initialize the data by
    - Splitting the world into countries
    - Spreading it's part of individuals across the countries by setting their starting coordinates and setting a part of them as infected.

- **Updating position:** The program takes **t_step** and **t_simulated** as inputs which specify how often the position must be updated and until when. The new position will be calculated depending on the **velocity,** current **x_pos** and **y_pos** of the individual and the **direction** of the individual.

```
country_type country = countries[individual.country];
if(individual.direction=='R'){
int new_x = individual.x_pos + velocity*time_step;
if(new_x>world_w){
new_x=world_w;
individual.direction='L'; //BOUNCING POLICY
}
else if(new_x>country.x_end){
individual.country++;
}
individual.x_pos=new_x;
}
else if(individual.direction=='L'){
int new_x = individual.x_pos - velocity*time_step;
if(new_x<0){
new_x=0;
individual.direction='R';
}
else if(new_x<country.x_start){
individual.country--;
}
individual.x_pos=new_x;
}
else if(individual.direction=='U'){
int new_y = individual.y_pos + velocity*time_step;
if(new_y>world_l){
new_y=world_l;
individual.direction='D';
}
else if(new_y>country.y_end){
individual.country+=world_w/country_w;
}
individual.y_pos=new_y;
}
else{
int new_y = individual.y_pos - velocity*time_step;
if(new_y<0){
new_y=0;
individual.direction='U';
}
else if(new_y<country.y_start){
individual.country-=world_w/country_w;
}
individual.y_pos=new_y;
}
```

```
individual.time_status+=time_step;
```

- **Spreading data across processes:** The next step after simulating the movement of the individuals involves collecting the information about the status of all individuals and make the information available to all the processes, in order to check the exposure. For this purpose, **MPI_Allgatherv** is used.

- **Updating exposure status:** This function updates the **status** of the individual by calculating the distance between the neighbours of each individual and marking the status depending on whether they have been exposed for the adequate amount of time specified in the **time_status.** Every process is checking each of his assigned individuals against the full list of individuals, gathered in the previous step.

```
if(considered_individual.status=='S'){ //SUSCEPTIBLE
int has_been_exposed=0;
for(int j=0; j<size;j++){ //SEARCHING IF THE INDIVIDUAL HAS BEEN EXPOSED IN THIS TIME
STEP
if(individuals[j].id!=considered_individual.id && individuals[j].status=='I' &&
distance(considered_individual, individuals[j])<max_spreading_distance){
has_been_exposed=1;
break;
}
}
if(has_been_exposed){ //IF EXPOSED, CHANGE STATUS TO EXPOSED AND RESET COUNTER
considered_individual.status='E';
considered_individual.time_status=0;
}
}
else if(considered_individual.status=='E'){ //EXPOSED
int has_been_exposed=0;
for(int j=0; j<size;j++){//SEARCHING IF THE INDIVIDUAL HAS BEEN EXPOSED IN THIS TIME
STEP
if(individuals[j].id!=considered_individual.id && individuals[j].status=='I' &&
distance(considered_individual, individuals[j])<max_spreading_distance){
has_been_exposed=1;
break;
}
}
if(!has_been_exposed){ //IF NOT EXPOSED, CHANGE STATUS BACK TO SUSCEPTIBLE AND RESET
COUNTER
considered_individual.status='S';
considered_individual.time_status=0;
}
else if(considered_individual.time_status>10*MINUTE){ //IF EXPOSED AND COUNTER REACHED
10 MINUTES, THE INDIVIDUAL BECOMES INFECTED
considered_individual.status='I';
considered_individual.time_status=0;
```

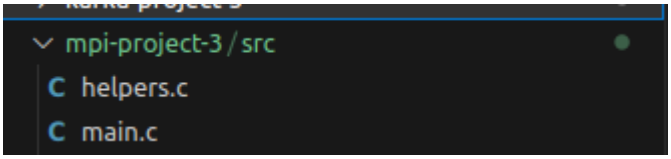
```
}
}
else if(considered_individual.status=='I'){ //INFECTED
if(considered_individual.time_status>=10*DAY){ //AN INFECTED INDIVIDUAL BECOMES IMMUNE
AFTER 10 DAYS
considered_individual.status='X';
considered_individual.time_status=0;
}
}
else{ //IMMUNE
if(considered_individual.time_status>=3*MONTH){ //AN IMMUNE INDIVIDUAL BECOMES
SUSCEPTIBLE AFTER 3 MONTHS
considered_individual.status='S';
considered_individual.time_status=0;
}
}
```

MPI Functions Utilised

1. **MPI_Allgatherv:** Used to gather data from each processes and deliver the combined data to all processes. The positional status of the assigned individuals at the end of each time step is calculated by each process and then sent to all the other processes. This ensures that all processes have information about all the individuals at the end of each time step and can perform the neighbour checks. The function is logically equivalent to a Gather followed by a Broadcast.
2. **MPI_Gather:** Used to gather together values from a group of processes to the main process (rank 0). This function is used to gather the summary of every country at the end of each day to the root process that will print them.
3. **MPI_Barrier:** Blocks until all processes in the communicator have reached this routine. This is used to ensure that the program waits adequately for all processes to have updated information.

Code Structure

The source code for the project contains only of 2 files:



1. **main.c:** This file is the main file that will be compiled and executed. The code for the following activities are included in this file:
    a. Configure input and initialise MPI
    b. Calculate the number of countries from user input and create countries
    c. Randomly initialise individuals across countries
    d. Execute the main loop to update status of the individuals
2. **helpers.c:** This file contains all the auxiliary functions that will be invoked from main.c. The definition of all the functions described below can be found in this file.

Execution

Pre-requisites:

To be able to run the program, the following packages must be installed with apt-get

```
sudo apt-get install gcc openmpi-bin
```

Compile and Execution:
The source can be compiled and executed with the following command:

```
mpicc mpi-project-3/src/main.c -o main.o -lm
mpirun mpi-project-3/src/main.o -np 4
```

Results
The expected result after executing the program is to see the number of susceptible, exposed, infected and immune individuals at the end of each day for each country. The following is a sample of the results when run for a simulated time of 2 days:

```
Number of individuals: 300
Number of individuals that are initially infected: 50
World Width: 1000
World Length: 1000
Countries Width: 500
Countries Length: 500
Velocity: 1.400000
Maximum Spreading Distance: 20
Time Step (seconds): 200
Time of Simulation (seconds): 172800
Number of countries: 4
Size of world: 1000000
Process 0, individuals: 75, infected: 14
Process 1, individuals: 75, infected: 12
Process 2, individuals: 75, infected: 12
Process 3, individuals: 75, infected: 12
Process: 0, Time: 200, Country: 0, Susceptibles: 41, Exposed: 21, Infected: 13, Immune: 0
Process: 0, Time: 200, Country: 1, Susceptibles: 73, Exposed: 22, Infected: 16, Immune: 0
Process: 0, Time: 200, Country: 2, Susceptibles: 46, Exposed: 17, Infected: 12, Immune: 0
Process: 0, Time: 200, Country: 3, Susceptibles: 28, Exposed: 2, Infected: 9, Immune: 0
Process: 0, Time: 86600, Country: 0, Susceptibles: 36, Exposed: 0, Infected: 39, Immune: 0
Process: 0, Time: 86600, Country: 1, Susceptibles: 54, Exposed: 3, Infected: 62, Immune: 0
Process: 0, Time: 86600, Country: 2, Susceptibles: 31, Exposed: 0, Infected: 44, Immune: 0
Process: 0, Time: 86600, Country: 3, Susceptibles: 7, Exposed: 0, Infected: 24, Immune: 0
Process: 0, Time: 173000, Country: 0, Susceptibles: 36, Exposed: 0, Infected: 39, Immune: 0
Process: 0, Time: 173000, Country: 1, Susceptibles: 54, Exposed: 3, Infected: 62, Immune: 0
Process: 0, Time: 173000, Country: 2, Susceptibles: 31, Exposed: 0, Infected: 44, Immune: 0
Process: 0, Time: 173000, Country: 3, Susceptibles: 7, Exposed: 0, Infected: 24, Immune: 0
```

Tests and Evaluation
The program was testing both on a single machine and also across distributed machines by setting up password-less public key authentication on a virtual machine and executing the program with `mpirun mpi-project-3/src/main.o -host <vm_ip> -np 4`

Performance Evaluation:
An analysis of the time taken to execute the function was performed to evaluate how modifying the input parameters affects the execution of the program. The baseline execution time was calculated based on the following inputs:

Number of individuals: 100
Number of individuals that are initially infected: 50
World Width: 1000
World Length: 1000
Countries Width: 500

Countries Length: 500
Velocity: 1.400000
Maximum Spreading Distance: 20
Time Step (seconds): 200
Time of Simulation (seconds): 172800
Number of countries: 4
Size of world: 1000000

The results when timing the run of the program yielded:

real 0m0.500s
user 0m0.560s
sys 0m0.128s

1. **Modifying number of individuals:** The tests and results after modifying the individuals are as follows:

Input Parameters	Results
Number of individuals: 50 Number of individuals that are initially infected: 50 World Width: 1000 World Length: 1000 Countries Width: 500 Countries Length: 500 Velocity: 1.400000 Maximum Spreading Distance: 20 Time Step (seconds): 200 Time of Simulation (seconds): 172800 Number of countries: 4 Size of world: 1000000	real 0m0.478s user 0m0.232s sys 0m0.243s
Number of individuals: 150 Number of individuals that are initially infected: 50 World Width: 1000 World Length: 1000 Countries Width: 500 Countries Length: 500 Velocity: 1.400000 Maximum Spreading Distance: 20 Time Step (seconds): 200 Time of Simulation (seconds): 172800 Number of countries: 4 Size of world: 1000000	real 0m0.670s user 0m0.996s sys 0m0.218s
Number of individuals: 1000 Number of individuals that are initially infected: 50 World Width: 1000 World Length: 1000 Countries Width: 500 Countries Length: 500 Velocity: 1.400000 Maximum Spreading Distance: 20 Time Step (seconds): 200	real 0m5.683s user 0m21.088s sys 0m0.229s

Time of Simulation (seconds): 172800 Number of countries: 4 Size of world: 1000000	

Increasing the number of individuals seems to increase the execution time and decreasing the number of individuals decreases execution time. This is expected as the more number of individuals implies more possibility of status change leading to more computations.

2. **Modifying number of processes:** The tests and results after modifying the number of processes running in a system with 8 CPUs are as follows:

Input Parameters	Results
Number of individuals: 100 Number of individuals that are initially infected: 50 World Width: 1000 World Length: 1000 Countries Width: 500 Countries Length: 500 Velocity: 1.400000 Maximum Spreading Distance: 20 Time Step (seconds): 200 Time of Simulation (seconds): 172800 Number of countries: 4 Size of world: 1000000 Number of processes: 2	real 0m0.568s user 0m0.582s sys 0m0.228s
Number of individuals: 100 Number of individuals that are initially infected: 50 World Width: 1000 World Length: 1000 Countries Width: 500 Countries Length: 500 Velocity: 1.400000 Maximum Spreading Distance: 20 Time Step (seconds): 200 Time of Simulation (seconds): 172800 Number of countries: 4 Size of world: 1000000 Number of processes: 6	real 0m0.534s user 0m0.552s sys 0m0.182s
Number of individuals: 100 Number of individuals that are initially infected: 50 World Width: 1000 World Length: 1000 Countries Width: 500 Countries Length: 500 Velocity: 1.400000 Maximum Spreading Distance: 20 Time Step (seconds): 200	real 0m0.584s user 0m0.628s sys 0m0.203s

Time of Simulation (seconds): 172800 Number of countries: 4 Size of world: 1000000 Number of processes: 8	

There does not seem to be any clear correlation between the number of processes and the execution time. Initially increasing the number of processes decreases execution time, however, beyond a certain limit, increasing the processes does not seem to have any significant impact on the performance.