POLITECNICO
MILANO 1863

Networked software for distributed systems

A Simple Model for Virus Spreading

Authors:
Stefano Carraro
Stefano Fossati
Andrea Restelli

Professors:
Prof. Luca Mottola
Prof. Alessandro Margara

2022-2023

Contents

1 Introduction

1.1 Request

1.1.1 Description

Scientists increasingly use computer simulations to study complex phenomena. In this project, you have 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. 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. An individual belongs to a country if it is in that country at the end of the day. A performance analysis of the proposed solution is appreciated (but not mandatory). In particular, we are interested in studies that evaluate (1) how the execution time changes when increasing the number of individuals and/or the number of countries in the simulation; (2) how the execution time decreases when adding more processing cores/hosts.

1.1.2 Assumption and Guidelines

- The program takes in input the following parameters
  - N = number of individuals
  - I = number of individuals that are initially infected
  - W, L = width and length of the rectangular area where individuals move (in meters)
  - w, l = width and length of each country (in meters)
  - v = moving speed for an individual
  - d = maximum spreading distance (in meters): a susceptible individual that remains closer than d to at least one infected individual becomes infected
  - t = time step (in seconds): the simulation recomputes the position and status (susceptible, infected, immune) of each individual with a temporal granularity of t (simulated) seconds
- You can make any assumptions on the behavior of individuals when they reach the boundaries of the area (for instance, they can change direction to guarantee that they remain in the area)

1.1.3 Technologies

Since the project involves a computationally heavy simulation that are suitable for parallel processing, we have chosen to use MPI (Message Passing Interface) as our parallel computing platform. MPI provides a standard for parallel computation and is well-suited for our simulation's computational requirements.

2 Project analysis and considerations

2.1 Considerations

Initially, we explored two different approaches and thoroughly evaluated the optimal design choice. The first approach involved evenly distributing the population among processes, while the second approach divided the entire area into subareas and assigned each subarea to a process.

After careful consideration, we analyzed the advantages and disadvantages of each design proposal:

- First approach:
  - *Advantages*: This design ensured an equal distribution of computation among processes, especially for updating the population's positions.
  - *Disadvantages*: Each process needed to be aware of the location of all infected individuals to perform infection checks. This approach required significant communication overhead and raised concerns about the computational complexity of infection control.
- Second approach:
  - *Advantages*: With this design, each process only managed a specific portion of the population in a particular subarea. Consequently, the communication requirement was reduced to sending information about individuals moving between areas.
  - *Disadvantages*: In the worst-case scenario, where all individuals were concentrated in a single subarea managed by a single process, potential bottlenecks could arise. Additionally, managing infections became more complex when non-infected and infected individuals were located in different process areas but in close proximity on the borders.

After careful consideration, we chose the first approach to minimize assumptions and ensure an evenly distributed workload among processes. However, it's important to note that the network could potentially become a bottleneck in this approach.

2.2 Additional Assumptions

- Individuals move in a random direction every time their position is updated.
- Individuals reappear in the other side of the area, so specularly to the center of the rectangle, if they overcome the boundaries of the whole area.
- Non-infected individuals become infected if and only if they are near at least one infected person for 10 consecutive minutes.
- Parameters W, L are multiple of w, l.
- Number of individuals $\gg$ number of countries.

3 Design and Implementation

In this section, we provide a brief overview of how our simulation algorithm works. Additionally, at the end of the section we present the detailed pseudocode of the algorithm.

3.1 World and countries organization

As the world of our simulation, we consider a rectangular area of dimensions $W \times L$, where each country has dimensions $w \times l$. The countries are organized in a grid structure, as illustrated in Figure 3.1. In the figure, the indexing criteria for the countries are also shown, indicating the row and column indices of each country.

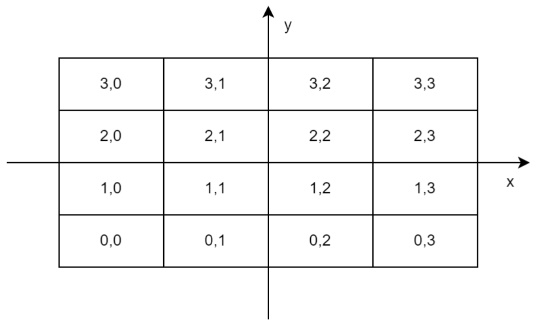


Figure 1: World and countries distribution

3.2 Algorithm implementation

The simulation algorithm is divided into three main phases.

1. **Input reading.** In this phase, the leader process reads the simulation parameters from a configuration file. After performing a validity check on the parameters, the leader distributes them to all other processes using a `Bcast` operation. Although a `Scatter` operation could have been used since the data to be sent are primitive integers, we chose broadcasting the configuration struct to minimize communication traffic.

2. **Simulation step.** This is the central phase of the algorithm, where each process performs the following steps:

   - Update the position of the individuals it manages.
   - Send the list of infected individuals to the leader.
   - Receive the list of all infected individuals from the leader.
   - Update the status of each individual it manages.

3. **Report building.** At the end of each day (i.e. when t is a multiple of 86400 seconds), each process sends an array to the leader. The array contains, for each country, the counts of NON_INFECTED, INFECTED and IMMUNE individuals among those it manages. The leader collects all the received reports, merges them, and creates a consolidated daily case report for the day. The report is then written to a CSV file.

Figure 3.2 illustrates the three phases described above in a visual diagram to facilitate comprehension.

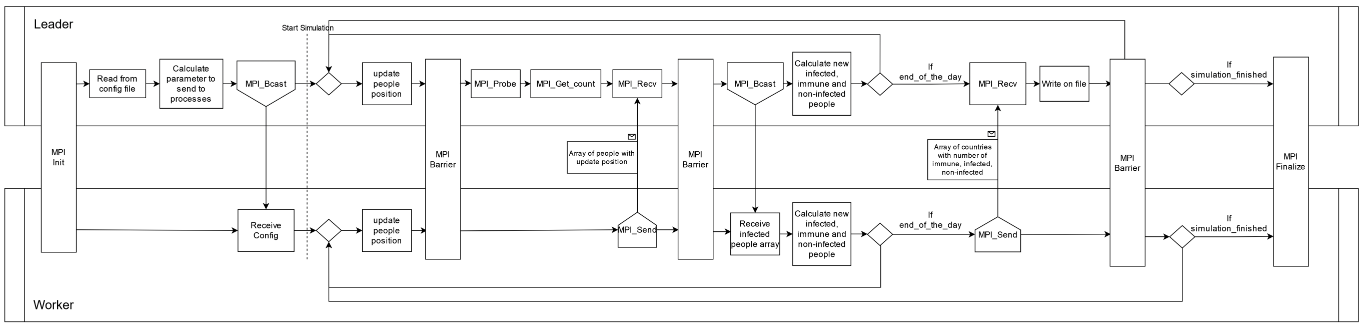


Figure 2: MPI processes call schema

Algorithm 1 Simulation Algorithm

```
if is Leader then
    Leader reads from file the simulation parameters
    Leader calculates the number of countries, how many non-infected and infected individuals for each process
end if
Broadcast of the simulation parameters (MPI_Broad)
while $time < totaltime$ do
    compute the new position
    if is not Leader then
        Send infected individuals to the leader (MPI_Send)
    else if is Leader then
        Receive infected individuals from non-leader processes (MPI_Recv)
    end if
    if is Leader then
        Broadcast infected_individuals
    end if
    for each individual do
        if IMMUNE || INFECTED then
            decrease timer
        else if NON-INFECTED then
            for each infected_individual do
                if NON-INFECTED is near INFECTED then
                    Decrease timer
                else if  then
                    Reset timer
                end if
            end for
        end if
    end for
    if is end of the day then
        Compute how many non-infected, infected and immune in each country
        if is not leader then
            Send information to the leader (MPI_Send)
        else if leader then
            Receive information from all non leader processes (MPI_Recv)
            Merge the data
            Write the data on file
        end if
    end if
end while
```

4 Performance Analysis

In this section we analyze how the *execution time* evolves by varying the number of individuals I and the number of countries C, as well as by adding more processing cores/hosts.

By analyzing the computational complexity of the designed algorithm we can make a theoretical forecast of the performance of the algorithm. We consider the following variables:

- N, the number of individuals.
- W, the number of worker processes.
- $S = \left\lfloor \frac{t_{\text{total}}}{t_{\text{step}}} \right\rfloor$, the number of steps of the simulation.

We can easily see that the complexity is upper bounded by the loop, performed by each worker, going through each NON_INFECTED individual and checking whether it is near at least one INFECTED individual.

As a result the computational complexity of the algorithm is

$$O\left(S\frac{N^2}{W}\right)$$

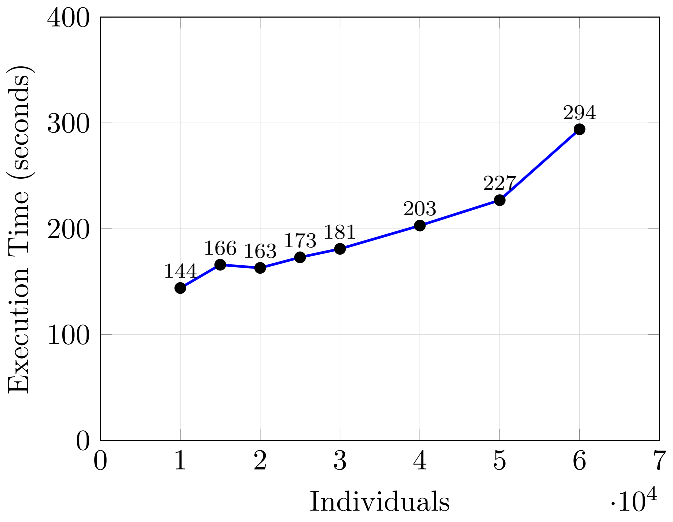
In the following subsections we explain the experiments we conducted to study the execution time variation when varying the variables affecting performances. For subsections 4.1 and 4.2 we used 2 processing hosts with 6 cores each. For all the experiments, t_{total} was equal to 30×10^6 seconds, more or less 347 days; t_{step} was equal to 5 minutes.

4.1 Varying the number of individuals

Our algorithm should be quite resilient to the growth in the number of individuals. This because, as previously highlighted, one of the advantages of our approach is to have an equal division among the working processes of the individuals to consider in the simulation.

Figure 3: Varying the number of individuals

N	**Execution Time**
10000	144s
15000	166s
20000	163s
25000	173s
30000	181s
40000	203s
50000	227s
60000	294s



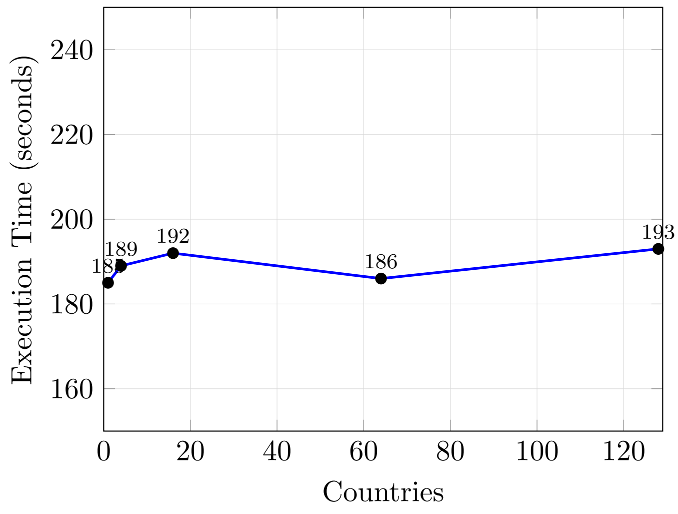
The main challenge in this case was to find an optimal equilibrium between parameters in order to avoid an early termination of the simulation due to lack of infectable individuals. However, the results obtained are more or less in line with what we expected: increasing the number of individuals doesn't make the curve of the execution time to explode exponentially.

4.2 Varying the number of countries

Our algorithm is designed to be unaffected by variations in the number of countries. This is because we only consider them at the end of the day when computing the daily report of cases. We merge the reports obtained by worker processes for the individuals they manage. We would also expect a slight decrease in performance as the number of countries increases since more reports need to be merged. As expected, the execution time of our algorithm remains stable and unaffected by the number of countries, as shown in Figure 4.

Figure 4: Varying the number of countries

C	Execution Time
1	185s
4	189s
16	192s
64	186s
128	193s



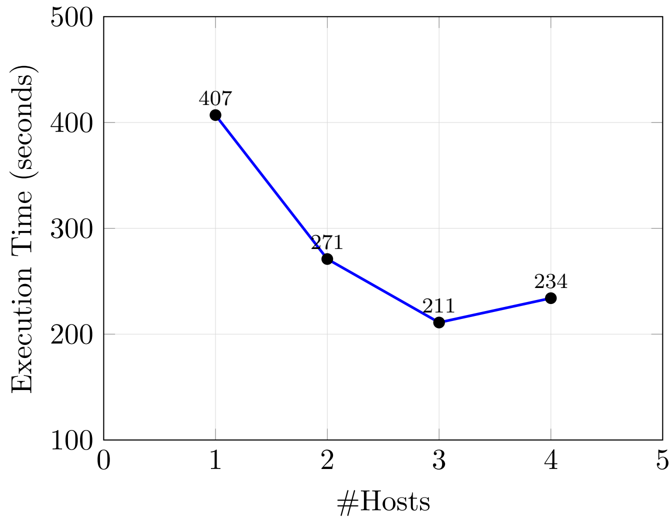
4.3 Adding more hosts

As highlighted in the computational complexity analysis of our algorithm, the performance is expected to improve with the addition of more hosts and processes. This expectation is confirmed by our experiment, where we considered 4 hosts, each with 2 cores (with the number of available cores limited to the minimum among them to ensure a fair comparison). We incrementally added hosts to the computation and observed the corresponding execution times.

From Figure 5, we can observe that, as anticipated, the execution time decreases as we add more hosts. However, an interesting observation is that when transitioning from 3 to 4 hosts, the performance actually decreases. We attribute this decrease to the fact that the advantage gained by adding 2 more cores to the computation is not sufficient to compensate for the decrease in performance caused by the latency introduced by the increased communication overhead among hosts.

Figure 5: Varying the number of hosts

I	Execution Time
1	407s
2	271s
3	211s
4	234s


Execution Time (seconds)
#Hosts