CONCURRENT AND DISTRIBUTED SYSTEMS ASSIGNMENT
Vaccine Distribution Plan

Draghici Bogdan
The Faculty of ACE, Craiova
Third Year, Calculatoare romana
CR 3.2.A

January 15, 2021

Contents

1 Problem statement

1.1 Context

In the actual context of the Covid-19 pandemics, the good news just came. Some pharmaceutical companies just announced that they found an efficient vaccine to fight against the SARS-Cov-2 virus. But they know they have to deliver the vaccine to all countries around the world and they need our help to serve all the requests. The pharmaceutical company owns multiple production facilities. Each facility has production robots creating vaccine doses and packing robots that pack the doses. They need our help to create a proper plan by using Java Concurrency to distribute all the ordered doses in time to end the Covid-19 crisis.

1.2 Pharmaceutical Company Instructions

1.2.1 Production facilities

There are a random number of factories (between 2 and 5) that create vaccine doses (it does not know how many are needed until the production plan is created and runs). The production facilities are designed as matrices with dimension N x N, where 100 <= N <= 500. Each production facility has a list of production robots. Every few seconds the production facility will request the position of the production robot in the factory. There can be no more than N / 2 production robots in each production facility.

1.2.2 Production robots

The production robots are spawning randomly in each production facility at a random place in the factory at a random time t where 500<=t<=1000 (milliseconds). Two production robots cannot be in the same place. The production robots create doses by moving in one direction (left, right, up, down). When they reach a wall of the production facility, they move in any other direction than the wall's.With each move, they create a vaccine dose. If a production robot is surrounded by other production robots and cannot move it stops for a random time t where 10<=t<=50 (milliseconds). Each time a dose is created, the production robot informs the production facility to know the created dose serial number. After each dose created, the production robot needs to recharge for a short while (X * 30 milliseconds). The production robots will work until all the orders are done.

1.2.3 Packing robots

The packing robots await to receive doses from the production facilities. They are available in the pharmaceutical company's headquarter. They can always read information from the production facility regarding the number of doses, but they won't be allowed access when the production robots notify the production facilities about newly created doses or when the production facility itself asks the production robots about new vaccine doses. A maximum of 10 packing robots can read from the production facility at a time. A random time must pass between two consecutive production facility readings. All packing robots will deliver the vaccine doses to the company headquarter. The packing robot number is known from the beginning (there are more than 8 because the company needs backup).

1.2.4 The Pharmaceutical Company (headquarter)

The headquarter contains all the production facilities. It creates production facilities and the production robots and assigns them to random production facility (remember that each production robot needs to register by itself to the production facility).

1.2.5 Things that are necessary

Please note that the (concurrent) control flow in this example is rather subtle. The moving of production robots and reporting to the production facilities is initiated independently by each of the production robots, as each of them is running a separate thread of control. Because of that, different production robots may execute the move action and report concurrently. But calling report also triggers calling the individual report method of all the production robots registered with the production facility. This again must be properly taken care of, because both report and move methods are synchronized, which means that they will not be called concurrently for a given object.

2 Choice of implementation, arriving to a solution

For the implementation of the program, I used synchronized methods and data structures, such as locks and semaphores. Also used a buffer class for moving the production robots and a TCP server for transporting the vaccines from the production facilities to the headquarters.

In the following I will explain each class with their fields and methods used in implementation of the program.

2.1 Pharmaceutical company

The pharmaceutical company represents the headquarters of the company. When instantiated, it creates the number of vaccine doses that must be produced, the production facilities, the packing robots and the vaccine receiver. All of them, beside the production robots, are available in the company headquarters as fields.
This class is also a thread, that when it is started, it also activates the whole process of production by starting the threads of the production facilities, packing robots and the vaccine receiver.
While the thread is running, and the production facilities have not yet completed the creation of the vaccine doses, its only task is to create and assign a new production robot to a random production facility every [0.5, 1] seconds. In order to create and add a new production robot, the company must first check the existence of at least one production facility which can add another production robot to its workforce. If the precondition checks, a new production robot is created and added randomly to a non-full production facility.
After the production facilities stopped producing new doses of vaccine, the company will shut down the packing robots and the vaccine receiver.

2.2 Vaccine receiver

The vaccine receiver is a component of the pharmaceutical company, and it's role is to serve as a TCP server. It also stores the number of vaccine doses that must be created by the production facilities and transported by the packing robots.
This class is also a thread and must run for the whole period of the production time, due to the fact that it will frequently receive vaccine doses transported by the packing robots from the production facilities to the headquarters.
Whenever such vaccine is received, the vaccine receiver will add it to the list of vaccine doses received by the headquarter, list which is synchronized in order to avoid the case the vaccine receiver tries to add multiple vaccine doses in the same time. After this operation, it will be checked if the number of vaccine doses received has reached the headquarters' desired number of vaccine doses. If it was reached, the vaccine receiver informs the production facilities to stop the working process.

2.3 Packing robot

The packing robot has the tasks of packing the vaccine doses received from the production facilities and transport them to the headquarters via TCP, to the vaccine receiver.
The packing robot extends the Thread class. Each packing robot works individually, and will stop when it is signalled that the desired number of vaccine doses has been achieved.
When working, the packing robot will choose randomly one of the production facilities from which to read the number of vaccine doses created that are waiting to be transported. If there are any vaccine doses waiting to be transported, the packing robot will try to "pack" a vaccine for transportation (get the id of a vaccine), thus removing it from the production facility. After the vaccine was packed successfully, the packing robot will transport the vaccine to the headquarters via TCP.
Due to the fact the headquarters can process only one vaccine dose at a time, it is important that the transportation method used by the packing robots to be declared as static and synchronized, to avoid any potential conflicts. After the vaccine dose was transported, the packing robot will rest for a period of time before proceeding read the vaccines from another randomly chosen production facility.

2.4 Production facility

The production facility is in charge of storing the vaccine doses created by the production robots, and delivering the vaccine doses to the packing robots. In this class happens most of the synchronization in the application, and it was necessary the use of a semaphore and two locks.
The production facility is also a thread, and while it is running, every few seconds ([3,5] seconds) will request the production robots' locations, time in which the request lock will be used to prevent the production robots from moving and the packing robots from asking for number of vaccines available or to receive vaccines. After every location request, the production facility will output its id, the vaccines available, the number of production robots in the facility and their locations.
When given the signal to stop working, the production facility will also notify the production robots in this manner. After the production facility finished working, it will output the total number of vaccines created and the number of vaccines that remained in it.
Whenever the pharmaceutical company asks the facility if it can receive another production robot, the method called will compare the current number

of production robots in the facility with the maximum number of robots allowed in it, and based on this comparison, it will confirm or infirm the possibility of receiving another robot.
The production robots notify the production facility about the creation of a new vaccine dose by calling a synchronized method that has as a parameter the id of the vaccine created. In this function the notification lock will be used in order to prevent packing robots from asking for number of vaccines available or to receive vaccines.
When a packing robot asks for the number of vaccine doses available or to receive a vaccine dose, the semaphore will be used to limit the number of packing robots to a maximum of 10, and the production facility will execute the request only when both locks are not in use.

2.5 Robots monitor

The robots monitor is used to and store the production robots, retrieve their locations in the facility, move one robot at a time, add a new production robot. It is also a component of the production facility, thus each production point having an instance of this class.
The monitor shares the request lock of the production facility, in this class being called movement lock. It is used to avoid the situation when a production robot tries to move to a new location in the same time the production facility is asking for the robots' locations.
When a production robot tries to move to a new location, it is verified to not be a wall in that location (the margin of the facility), then the movement lock is used to see the locations of all robots in order to verify the existence of any robot on that specific location. In the case both checks passed, the robot is moved to it's new location. After this, the movement lock is unlocked. The movement method will return if the movement operation was a success or not.
When the production facility is receiving a new production robot, its starting location will be generated randomly till an empty location is generated and the robot is assigned to it. After that, the robot is added to the list of production robots of the monitor, has its id set the id and then starts to work.
When the production facility asks for the locations of the robots, the robots monitor will return of list with the locations of all robots.

2.6 Production robots

The production robot is the one that creates the vaccine doses by moving in one of the 4 cardinal directions. This class is also a thread that runs

individually to create the requested vaccine doses.
While the production robot thread is running, it will try to move to one adjacent location, chosen randomly. If it is occupied by another robot, it will try other adjacent location, again chosen randomly. If all 4 adjacent locations are occupied by other production robots, the robot will wait for a period of time before trying to move again. In case it was able to move to an adjacent position, it creates a new vaccine and notifies the production facility about the vaccine id, and then wait another period of time before trying to move again.

3 Generating test data

For generating the data for the pharmaceutical company I created another class that contains static methods for generating the number of production facilities and their sizes, number of packing robots, number of vaccines to create and a number in a given interval. This class also contains the minimum and maximum values for the numbers to be generated mentioned earlier.

3.1 The generator

The main method used for generating random values is *createNumber(minVal, maxVal)*, that created and returns an integer number in a given interval using the *random()* method from the Math class.
For generating the production facilities is used the *createNumber(minVal, maxVal)* for tow purposes: first to create the number of facilities (the company can have [2,5] facilities), and second to create the size of each facility (the facilities have the size in the interval [100,500]). After the production facilities are created with an ID and a size, a list that contains all facilities generated will be returned by the function.
When generating the packing robots, the production facilities must be given as a parameter, in order to assign them to each packing robot created. The *createNumber(minVal, maxVal)* function is used here in order to determine the number of packing robots that must be created. When instantiating a new packing robot, it will receive the production facilities as a parameter. After all the packing robots have been created, the method will return a list with the packing robots created.
To determine how many doses of vaccine should be created, it is enough to return the integer created by the *createNumber(minVal, maxVal)* function, where minVal and maxVal are the minimum and maximum values for the number of vaccines to be created by the pharmaceutical company.

3.2 Data interpretation and graphics

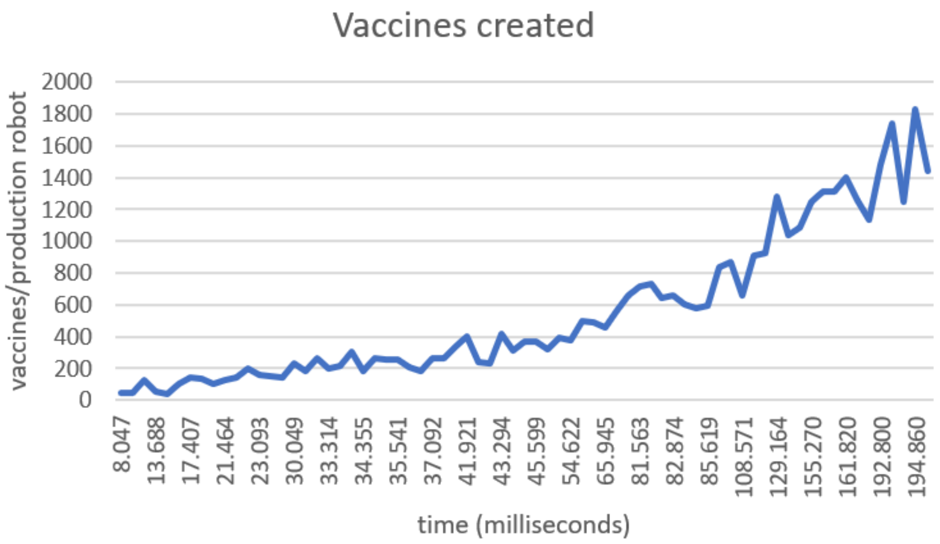

Figure 1: Vaccines created per production robot

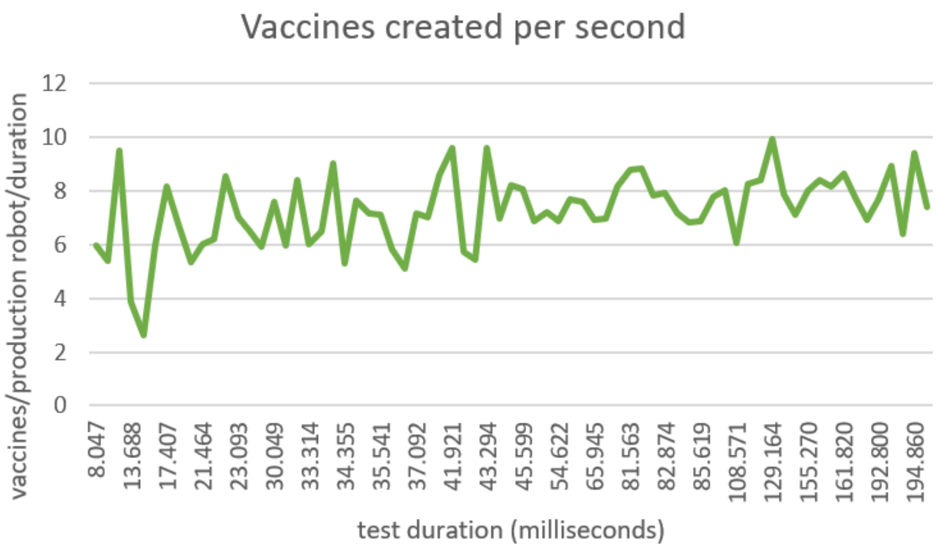

Figure 2: Vaccines created per production robot each second

It is observed that the total number of vaccines created by a production robot increases linearly with the test duration, and the number of vaccines created by a production robots each second fluctuates around 7 vaccines per second.

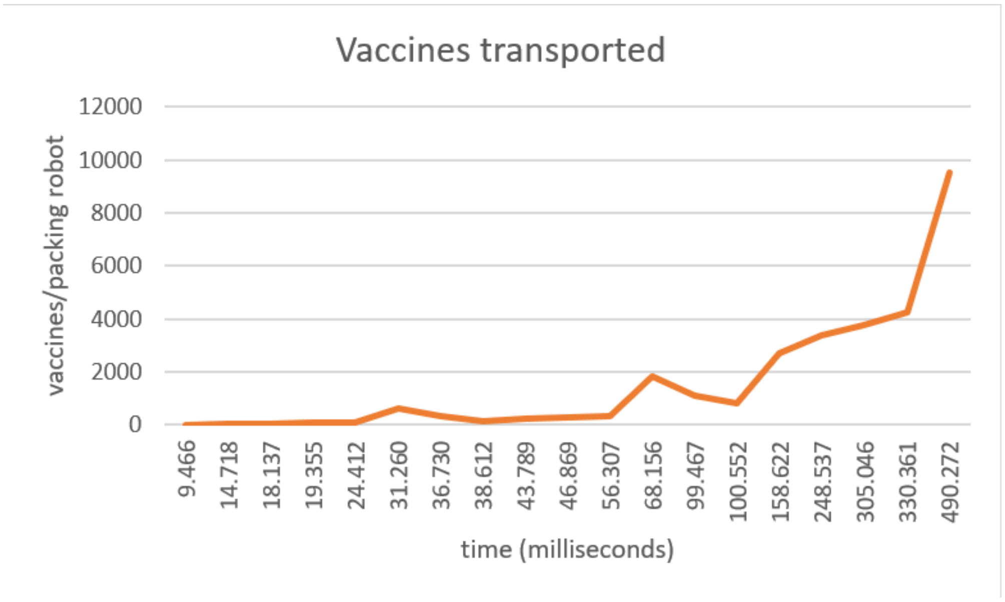

Figure 3: Vaccines transported per packing robot

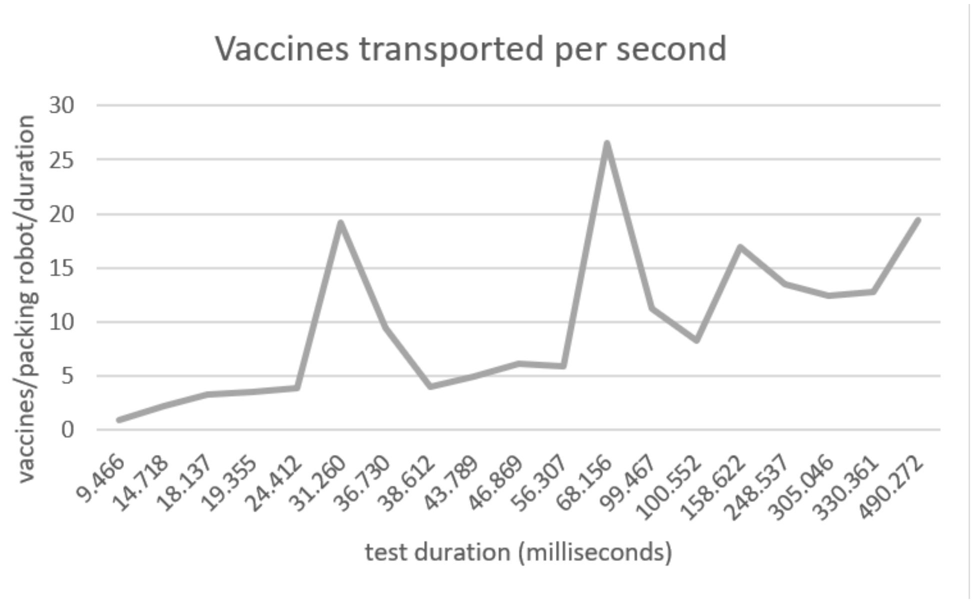

Figure 4: Vaccines transported per packing robot each second

The total number of vaccines transported by a packing robot does not increase that much for small tests, but when having bigger tests it starts increasing more reported to the test duration. The number of vaccines transported by a packing robot each second slightly increases as the test duration increases.

4 Observations and interpretations

4.1 The specification of the input

The input is auto-generated at the instantiation of the pharmaceutical company. The maximum number of packing robots I chose to be 50, due to the fact that it is the maximum number of robots that can read simultaneous from a maximum of 5 production facilities. The number of vaccines to be created I set to be in generated in the interval [20000,50000], in order for the test to finish neither in less than a minute, nor in more than 4 minutes.

4.2 The specification of the output

In the console are logged the following:

1. the information about the generated data;
2. after each request of the facility for the robots' locations: the number of vaccines waiting, number of robots and their locations in a facility;
3. every time the headquarters receives another 1000 vaccines;
4. when the requested number of vaccines has been received;
5. how many vaccines each facility produced and and for how long;
6. the total number of vaccines available in the company headquarters and how much it took to receive all of them.

5 Results and Conclusion

5.1 Summary of achievements

By making this problem, I learned how to use and interconnect locks, semaphores, monitors, and synchronization on lists and methods, in a somehow complex program with that utilizes a large number of threads to execute concurrently different calculations. If it was to be executed by a single thread, not only would have taken a huge amount of time, but also be unrealistic compared to the working process within a company.

5.2 Behavior and observed potential improvements

After around 10000 vaccines received by the headquarters, the total number of production robots is large enough to produce more vaccines than the packing robots can handle, thus the number of vaccines waiting to be transported growing till the headquarter receives the desired number of vaccines and stops the whole process of production.
When lowering the waiting time between two readings of the packing robot below 5 milliseconds, I observed no real improvement in the lengths of the lists of vaccine doses waiting to be transported. I believe this is due to the limitation of a maximum of 10 packing robots reading from the same production point.
To keep a balance between the production and transport, so the number of vaccines that wait for transportation will not start to grow without stopping after a threshold, I think we could increase the maximum number of packing robots that can read from a facility using a ratio of the number of production robots that work in a facility, or slightly increase the waiting time after the creation of a dose whenever a new production robot is added to the facility.