**Report**

**Cloud Computing**

**Tarek Fouda**

This project simply requires us to simulate a datacenter that monitors vehicles in a parking lot of a plant. The simulation will be working for 24 hours (One day). The parking lots number is 2,560 parking lots and the number of employees (cars) in this plant is 7,680 cars.

The Input for my program is simply a text file that is generated randomly and is consisted of 7,680 lines which corresponds to the arrival time of each and every car of the 7,680 we have in the plant. We do not need another input file (Departure time of cars) as the professor stated that all cars will leave after the 8 hours shift. So basically cars stay 8 hours and the departure time will be equal to arrival time + 8.

I have used minutes in my program, so for example, the simulation lasts for 1440 minutes which is equal to 24 hours x 60 minutes.

My code consists of four different classes, Jobs.java, cars.java, parkingSpot.java and Simulation.java

**Cars.java:**

This class is an object for a one car and has instances variables such as this car ID, the spot number that this car is occupying if valid, the arrival time for this car, the departure time for this car, the time spent in the plant which is a constant (8 hours), a Boolean variable named busy which states if the car is busy or not by a job. It has a constructor method that initialized the instance variables and a ToString method that displays information.

**Jobs.java:**

This class is an object for one job and has instances variables such as the job ID and the car ID that is handling thisjob, also a Boolean value which states if the job is completed or not, a Boolean value which states if the job is migrated successfully or not, a job duration variable. A double variable that saves the data the job will produce which I did research and figured that the best value would be (Math.*random*()%(3000-1000)+1000) \* 2, the VM size which we assumed that the job are between 1 GB and 4 GB and the value for it is Math.*random*()%(3000-1000)+1000; .

It has a constructor and a method to calculate the time it takes to replicate the data in same cluster and in a different cluster as well.

Data storage time is a constant which is equal to the data this job will produce /3250. Note that 3250 is the disk bandwidth in Mb/minute as the professor stated that the disk bandwidth is 54 Mb/s, so I did the transformation to get this 3250.\

**parkingSpot.java:**

This is an object that resembles a parking spt having instances variables like the parking lot spot id, and the corresponding cluster number,region number and group center number. Also it shows if it is occupied or not and if it is, then it saves the car ID occupying this spot. Also a ToString method to display all the information is in this class as well.

**Simulation.java:**

It is the heart of the code, as this simulation class contains the main method and important method that runs the simulation itself and initializes all the instance variables and read the input file which is called run().

Also it has four methods that initializes the regions of the parking lots, as we know we have 2,560 parking space. So if we divide them into 4 regions we will get 640 each which are the DC children. This 640 is divided into 4 other parts so we have 160 spots which are the RC children. Then the GC has four children, each is in charge of 40 parking spots. This has been implemented according to the understanding of the project statement.

In other words, if the parking lot has an ID that is below 640 then it is definitely in region 1.

Terminology:

parkingSpot ID < 640 then it exists in region 1.

Parking spot id < 160 then it exists in region number 1

Parking spot id < 40 then it exists in cluster number 1

parkingSpots(**int** spot\_id, **int** cluster\_number, **int** region\_number, **int** group\_center\_number, **int** occupied)

**run():**

The algorithm of run() method is so simple, it simply initialized arrays of cars Object which has 7,680 cars as states in the project description. Initializes jobs array of Jobs object and the size of this array is 7,680 jobs as well, and last but not least I initialize the array of parking spots object and the number of it is 2,560.

I keep in mind that I read the information from an input file, this input file has all the arrival times of the 7,680 cars, so I read line by line and assign the arrival times of every car.

So for example the first car, car[0]. Arrivaltime will be equal to the first line in the input file Car[1].arrivaltime will be equal to the second line of the input file and so on. Also I initialize the car ID’s for all cars, I basically loop all over the array from car 1 to car 7680, and I assign an ID to each car, so for example car 1 id is 0, car 2 id is 1, car 3 id is 2 and so on.. I do the same and initialize the id numbers for the parking spots and the jobs as well.

Then comes initializing the departure times for all of the cars, which is simply the arrival time + 8 hours. And we assume that car[0] is parked in parkinglot[0] and is doing job[0], car[1] is parked in parkinglot[1] and is doing job[1] and so on.

Then we initialize the spot numbers regions, basically I have four methods, each one corresponds to a certain region, and according to the spot number ID, I pass a specific value for the cluster number, region number to the constructor, depending on the terminology that I mentioned above which I understood from the project.

I start with printing all the information and displaying it for the cars and the jobs, I also print the cars that I will be able to keep track off along the simulation, and those are the cars that have departuretime less than the end time of simulation.

After all of the initialization steps, comes the actual simulation, as I said earlier we will deal with minutes, and we will simulate for 24 hours i.e: 1440 minutes. So we have a counter that starts from 1 to 1440 and this will be the iteration for every minute.

For every car in the car array we check if the minute we are in is greater than the departure time of this car, then that means this car left now and we will need to change the variables for it, like we need to make this car not busy, not assigned to a spot number anymore, the parking spot is not assigned to this car anymore, the corresponding job is no longer assigned to this car, and so on.

Also we need to check if the minute we are in right now is equal to an arrival time for any car outside the parking lot, and that this car is not assigned to any parking spot, then in this case we will have to search among the 2560 parking lots for a suitable spot for this car, mainly unoccupied parking spot, if we found one, then the car will reserve this, if not then the car will turn away.

**Migration:**

So when are we supposed to migrate a job from a car to another?

I assumed that we will have to migrate a job from this car to another in case that the job duration is greater than the time spent in the plant for a specific car. In this case we will have to migrate the job from this car that is about to leave to another car, 1 minute before the car actually has to leave.

What is the best car to handle the job?

To find an alternative car, we needed to search in the cars array for a specific car that has time spent in the plant GREATER than the job\_duration remained for this job. Also we have to make sure that this car is not busy with another job, and the departure time of this car is greater than the departure time of the leaving car + job remained. If we found one that meets all this requirements, we migrate the job and make the new car responsible for this job. Otherwise we just say that we failed to migrate this specific job.

The task is to implement data replication and I have googled and made a lot of research to come up with a technique to allow me to do that. Also the task was to implement job migration before the car leaves the parking lot successfully so we could assure availability and that will reduce the number of jobs that failed.

It was so challenging and I had to gather so much information along with the assumptions documented in the code to come up with a strategy that makes us reach our target and goal.

I have replicated the data into three other cars one is randomly chosen in the same cluster, and the other is randomly chosen in the same region while the last car is any car in the whole parking lot which contains 2560 cars.

I displayed all the initial information and the simulation as soon as we start it from minute 1 to minute 1440 in a word document. It displays everything and all the information and states which jobs failed to migrate and which were successful.

As I said earlier, my output is in a text file that prints out for the user the information initially of all the jobs, cars and parking spots, and then it displays these information after the simulation explaining which jobs have been migrated successfully and which ones failed to be migrated. Also it shows every job has been replicated to which three cars, because as the project statement requires, a job should be replicated to three cars, one in the same cluster, one in the same region and the last one in the parking lot. That has been handled in the run() method.