Elevator-Simulation Project Report

CIS691 Parallele Prog & Multithreading

XinranXu   May, 6 2020

Chapter1: Introdunction

1.1 Interodunction to elevator simulation

Simulation is a way of developing application or design simulating real world, the way which help in clarifying the real idea. The reasons upon using simulations may be for literature, cost, time, or safety reasons.

The simulation runs according to the parameters defined by user. In a simulation, Elevators run individually, load/unload orders randomly generated on different floors. After a certain number of requests are completed, it gives several kinds of metric statistics based on overall performance.

Passengers arrive in the building at random times and. During the time a passenger is in the building may request elevator from the floor where he is currently located and destination time random generated. The request also specifies direction, up or down. Passengers on the lowest floor only request up service; those on the top floor only request down service; all others may request either service.

when a passenger entering an elevator, The elevator closes its door and moves to that destination floor, possibly stopping on intermediate floors to deliver other passengers who may have selected intermediate floors. When an elevator arrives at a destination floor it stops, opens its door, and discharges any passengers who have selected that floor. Having stopped, opened its door and unloaded its passengers, the elevator then admits any passengers who maybe waiting for service in the direction the elevator may be currently moving, subject to the restriction that the elevator may not exceed its capacity to carry passengers. Passengers who do not exceed in boarding the elevator because its full must continue waite in the line at the door.

For my implementation, assume that the elevator moves passengers from departure floor to destination floor in simulation. the basic objects of the system are: Elevator, Passenger and Scheduler which controls the elevators. Elevator class keeps track of current state of it, its capacity, service time required at each floor and time required to move to next floor, current number passengers on elevator, number of requests finished etc… Operation on Elevator object are synchronized to make the system thread safe.

Chapter2: Design

2.1 Thread

The simulation split to four kinds of thread:

1. Elevator thread:
   moving vertically and carries passengers to their destination floors. moving between each floor takes 200 milliseconds.
2. Passenger thread:
   Each passenger represents one request. Generating passengers every 1000 milliseconds. Departure floor and destination floor also randomly between 1 to 30.
3. Scheduler thread:
   Scheduling elevator to each passenger.
4. main thread:
   main thread organized all other threads and makes TCP connection between server side(c++) and client side(python).

2.2 Class

The simulation have two class:

1. Elevator
   Each elevator contains: status(up, down and idle), target up floor, target down floor, a list passenger on the elevator, current floor

```
class elevator {
   int id;
   int max_passenger;
   int finished_requests;
   int cur_floor;
   int cur_passengers;

   status s;
   int target_up_floor = 0;
   int target_down_floor = 31;
   list<passenger*> cur_queue; //elevator queue, passengers in elevator
   //list<passenger*> finished_request;


public:
   elevator(int id);
   ~elevator();
   void load();
   void unload();
   status get_status();
```

```
    int get_id();
    list<passenger*> get_curqueue();
    int get_cur_floor() { return cur_floor; }
    void going_up();
    void going_down();
    void is_idle();
    void set_target_down(int start_floor);
    void set_target_up(int start_floor);
    void print_elevator();
};
```

2. Passenger
   Each passenger contains: random start floor, random end floor, generated time, load time, unload time, direction(up or down)

```
class passenger
{
    int id;
    chrono::system_clock::time_point generate_time;
    int start_floor;
    int end_floor;
    chrono::system_clock::time_point load_time;
    chrono::system_clock::time_point unload_time;
    direction d;

public:
    passenger(chrono::system_clock::time_point generate_time, int id, int start_floor, int
end_floor);
    ~passenger();

    int get_start_floor();
    int get_end_floor();
    direction get_direction();
    chrono::system_clock::time_point get_generate_time();
    chrono::system_clock::time_point get_load_time();
    chrono::system_clock::time_point get_unload_time();
    void set_unload_time(chrono::system_clock::time_point time);
    void set_load_time(chrono::system_clock::time_point time);
    void print_passenger();
};
```

2.3 Logic

- There are 4 elevators (four elevator threads) available to take passengers up and down the floors of a 30 floor building

- Each elevator have three status: down, up and idle
- Each passenger decides which floor to go when generated
- It takes each elevator fixed time of moving to move between each floor
- the time asking for an elevator, their starting floor, and their destination floor random generated by passenger thread
- When an elevator arrives at a floor, it picks up all of the passengers waiting at the floor, up to the maximum capacity of 7, and then starts moving to the next target floor
- If stopped and there is no request in current direction, the elevator either begins moving up or moving down depending on the direction needed to pick up the passenger
- target floor holds a next destination floor the elevator will move, updated when elevator arrive the destination
- When stopped at a floor, if there are any passengers in the elevator with that end Floor as a destination, it discharges the passengers. Then, if there are any passengers with that floor as their start floor it picks up passengers up to the capacity of the elevator
- After a passenger arrives destination floor, finished request list add one.
- scheduler check the state of elevator every 6 seconds.

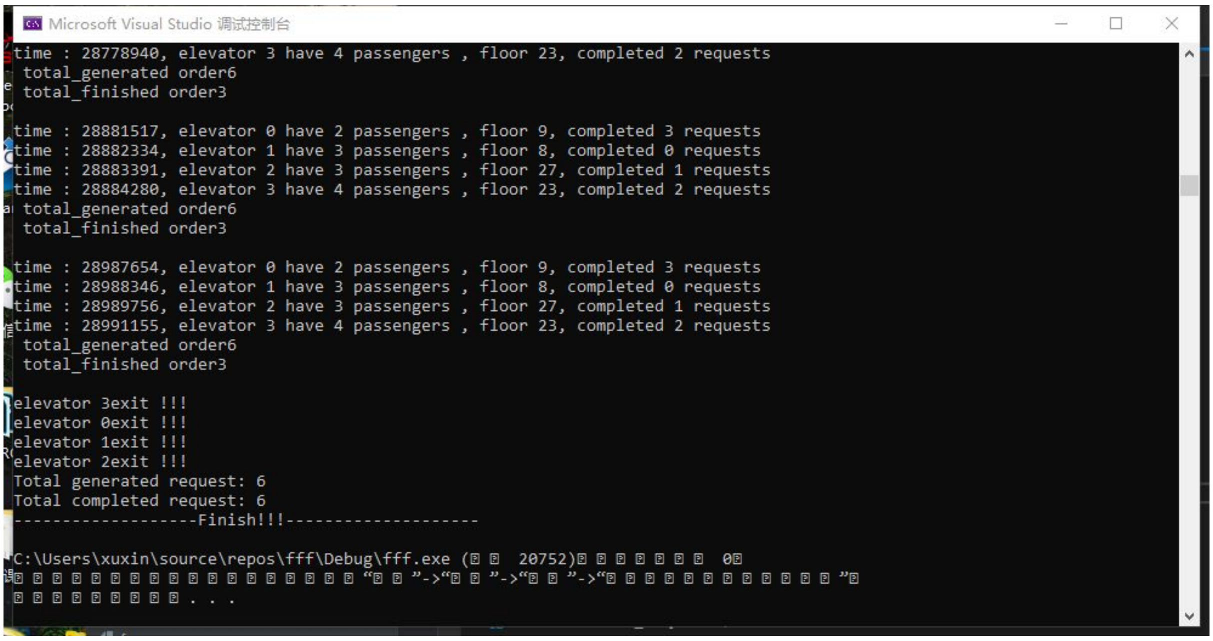
2.4 Shared Resource

Two important shared resource:

1. Each floor have a passenger lists which is s waiting list waiting for service.
2. Each floor have a mutex to protect passengers lists when these lists are updated.

Chapter3: Result and Analysis

3.1 Result & Screenshots



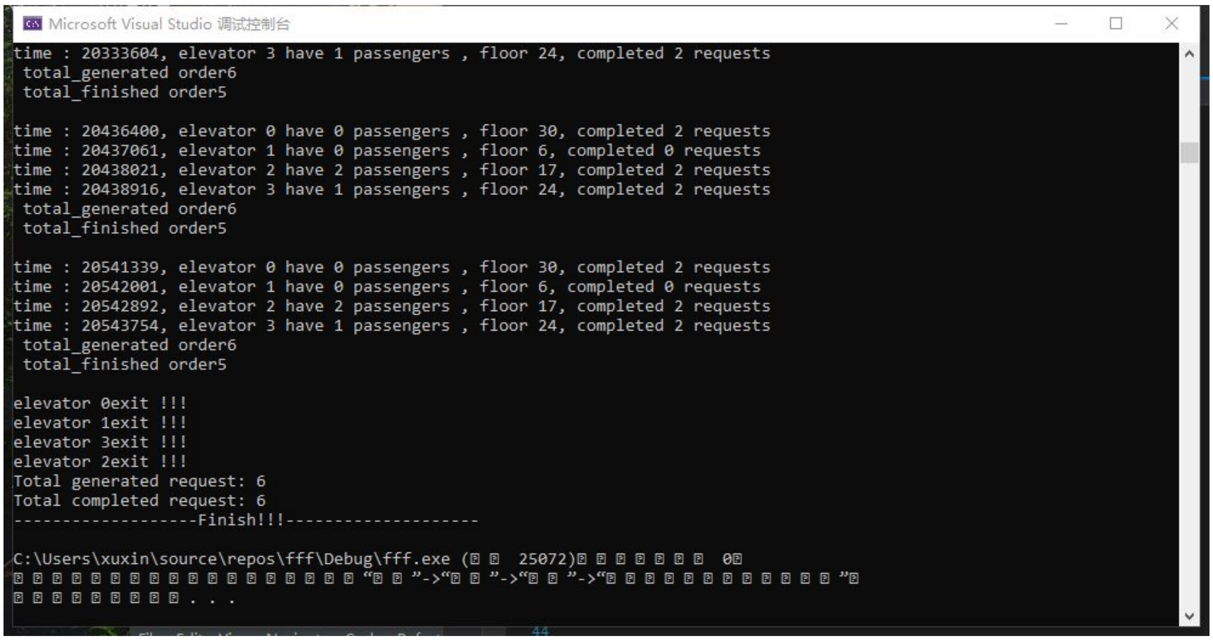
```
time : 28778940, elevator 3 have 4 passengers , floor 23, completed 2 requests
 total_generated order6
 total_finished order3

time : 28881517, elevator 0 have 2 passengers , floor 9, completed 3 requests
time : 28882334, elevator 1 have 3 passengers , floor 8, completed 0 requests
time : 28883391, elevator 2 have 3 passengers , floor 27, completed 1 requests
time : 28884280, elevator 3 have 4 passengers , floor 23, completed 2 requests
 total_generated order6
 total_finished order3

time : 28987654, elevator 0 have 2 passengers , floor 9, completed 3 requests
time : 28988346, elevator 1 have 3 passengers , floor 8, completed 0 requests
time : 28989756, elevator 2 have 3 passengers , floor 27, completed 1 requests
time : 28991155, elevator 3 have 4 passengers , floor 23, completed 2 requests
 total_generated order6
 total_finished order3

elevator 3exit !!!
elevator 0exit !!!
elevator 1exit !!!
elevator 2exit !!!
Total generated request: 6
Total completed request: 6
-------------------Finish!!!-------------------

C:\Users\xuxin\source\repos\fff\Debug\fff.exe (□ □  20752)□ □ □ □ □ □ □  0□
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ “□ □ ”->“□ □ ”->“□ □ ”->“□ □ □ □ □ □ □ □ □ □ □ □ □ □ ”□
□ □ □ □ □ □ □ □ □ □ . . .
```



```
time : 20333604, elevator 3 have 1 passengers , floor 24, completed 2 requests
 total_generated order6
 total_finished order5

time : 20436400, elevator 0 have 0 passengers , floor 30, completed 2 requests
time : 20437061, elevator 1 have 0 passengers , floor 6, completed 0 requests
time : 20438021, elevator 2 have 2 passengers , floor 17, completed 2 requests
time : 20438916, elevator 3 have 1 passengers , floor 24, completed 2 requests
 total_generated order6
 total_finished order5

time : 20541339, elevator 0 have 0 passengers , floor 30, completed 2 requests
time : 20542001, elevator 1 have 0 passengers , floor 6, completed 0 requests
time : 20542892, elevator 2 have 2 passengers , floor 17, completed 2 requests
time : 20543754, elevator 3 have 1 passengers , floor 24, completed 2 requests
 total_generated order6
 total_finished order5

elevator 0exit !!!
elevator 1exit !!!
elevator 3exit !!!
elevator 2exit !!!
Total generated request: 6
Total completed request: 6
-------------------Finish!!!-------------------

C:\Users\xuxin\source\repos\fff\Debug\fff.exe (□ □  25072)□ □ □ □ □ □ □  0□
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ “□ □ ”->“□ □ ”->“□ □ ”->“□ □ □ □ □ □ □ □ □ □ □ □ □ □ ”□
□ □ □ □ □ □ □ □ □ □ . . .
```

3.2 analysis

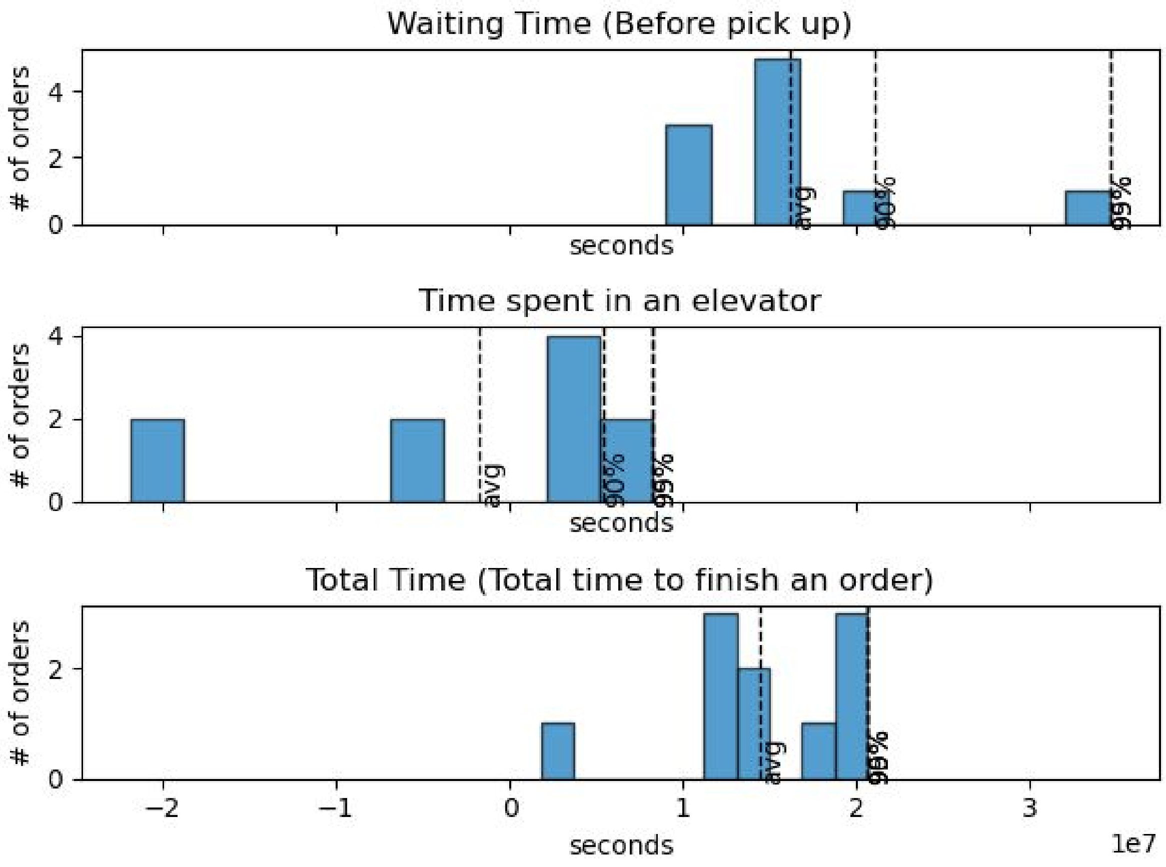
Comparing two cases using python

setting 1:

    max_floor : 30

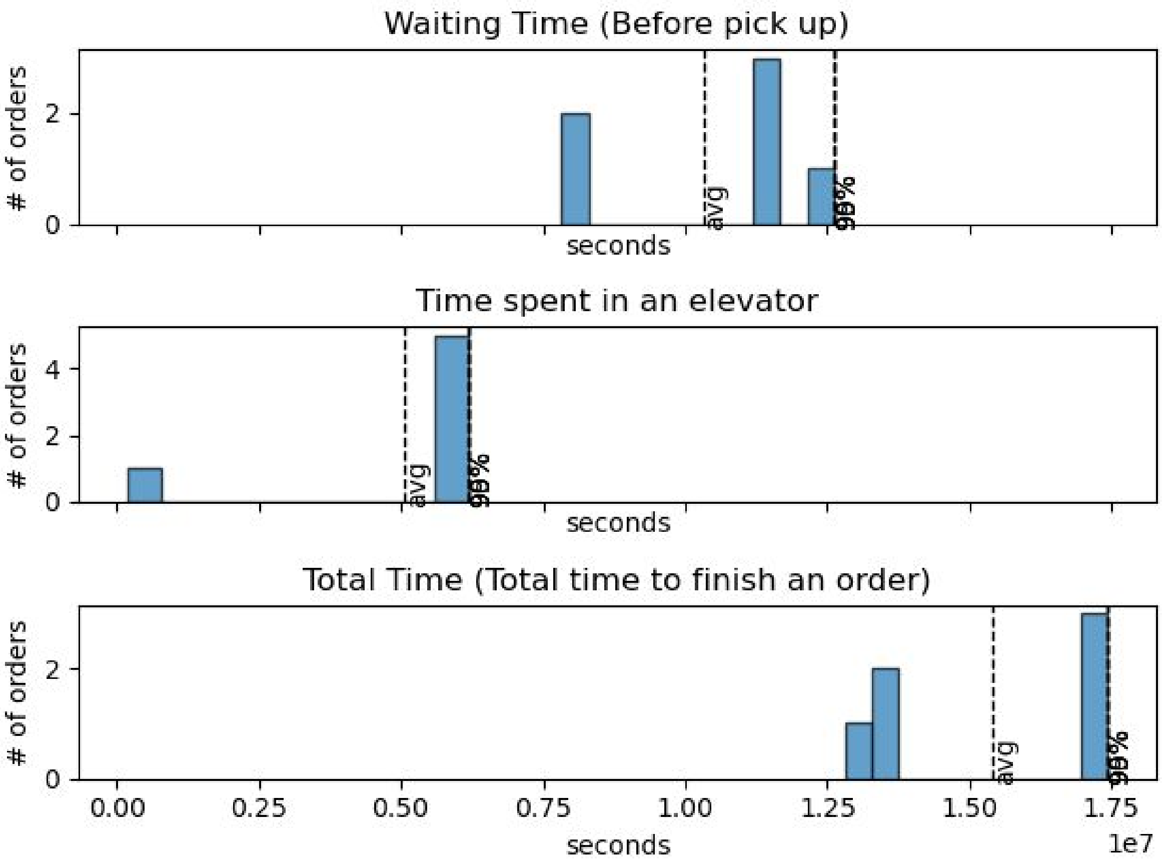
min_floor: 1
elevator_number: 4
passenger_interval: 2000
elevator_interval: 200
check_interval: 100
load_unload_interval: 1000
target number of requests: 10

data analysis result:



setting 2:
passenger_interval: 3000
elevator_interval: 400
check_interval: 100
load_unload_interval: 1000
target number of requests: 10

data analysis result:



Chapter4: Conclusion

Consider integrity of this elevator simulation, I still working on frontend by using javascript. TCP server side have set but user interface haven't finished. I will complete the the connection between frontend and backend in the future.