The report of Coputer-aided simulations lab

Lab11

QI AN s288400

Date: Dec. 04th 2022

Part 1 The introduction

1.1. The Introduction

This project adds some extensions to the queuing simulator (lab1) we have developed. The level of confidence interval is kept at 95%, and 8 groups of utilizations are set for simulation from 0.1 to 0.99.

This project is divided into 3 service time of scenarios:

EXP: exponential distribution with mean=1

DET: deterministic =1

HYP: hyper-exponential distribution with mean=1 and standard deviation=10

This project aims to:

(1). detect the end of transient in an automated way

(2). evaluate the accuracy of results

1.2. The principles

(1). Randomly generate service times using a hypergeometric distribution, according to mean=1 and standard deviation=10. (formular 1)

$$\begin{cases} E(h) = n\dfrac{M}{N} \\ Var(h) = \dfrac{nM(N-M)(N-n)}{N^2(N-1)} \end{cases} \quad (1)$$

(2). Customers enter, waiting in a queue for their number to be called out, get service from the company, and finally leave. This is a simple first in first out(FIFO) queueing system, and we encounter many queueing systems in our day to day lives, from grocery stores to amusement parks they're everywhere. And that's why we must try and make them as efficient as possible. There is a lot of randomness involved in these systems, which can cause huge delays, result in long queues, reduce efficiency, and even monetary loss. The randomness can be addressed by developing a discrete event simulation model, this can be extremely helpful in improving the operational efficiency, by analyzing key performance measures.

In this project, I am going to be simulating a queueing system for double servers.

1.3. Tool and platform

Program language: Python 3.8

Tool: the Google-Colaboratory for coding;

Text: Microsoft word

Part 2 The procedure

2.1. Core code Explanation

```
def arrival(time, FES, queue, service_time, utilisation):
        global users

        # cumulate statistics
        data.arr += 1
        data.ut += users*(time-data.oldT)
        data.oldT = time

        # sample the time until the next arrival
        conditional_time = 1.0/(service_time/utilisation)
        inter_arrival = random.expovariate(conditional_time)

        # schedule the next arrival
        FES.put((time + inter_arrival, "arrival"))

        # update the state variable, by increasing the no. of clients by 1
        users += 1

        # create a record for the client
        type = random.randint(0,2)
        customer = Customer(time, type)

        # insert the record in the queue
        queue.append(customer)

        # if the server is idle start the service
        if users==1:
                # sample the service time
                service_time = service_time
```

(figure2. Arrival function)

```
# ******
def departure(time, FES, queue, service_time):
        global users

        # get the first element from the queue
        customer = queue.pop(0)
        delayT = time-customer.arrival_time
        # cumulate statistics
        data.dep += 1
        data.ut += users*(time-data.oldT)
        data.oldT = time
        data.delay += (delayT)
        data.timelist.append(time)
        data.delayList.append(delayT)
        # update the state variable, by decreasing the no. of clients by 1
        users -= 1

        # check whether there are more clients to in the queue
        if users >0:
                # sample the service time
                service_time = service_time
                # schedule the departure of the client
                FES.put((time + service_time, "departure"))
```

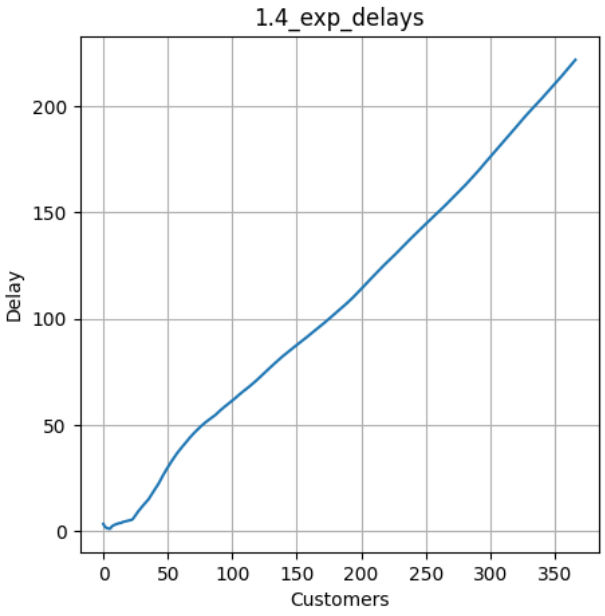
(figure3 Departure Function)

```
# we could add the simulation result in
def plot_TransientFigure(customerList, delayList, avgDelay, utilisation, indicator):
    scenario = ""
    if indicator ==1:
        scenario = "exponentially distributed"
    elif indicator ==2:
        scenario = "deterministic distributed"
    elif indicator ==3:
        scenario = "hyper-exponential distribution"
    #paint the time with number of users in group
    plt.plot(customerList,delayList,color ='blue')

    #paint the theorical average number of users
    plt.axhline(avgDelay, linestyle='--', color='C1', label='Avarage Simulation delay')

    plt.xlabel("time")
    plt.ylabel("delay time")
    plt.title(f"Transient Detectation with Cumulative Delay ({scenario}scenario) of {utilisation}utilisation")
    plt.legend()
    plt.grid()
    plt.show()
    plt.close()
```

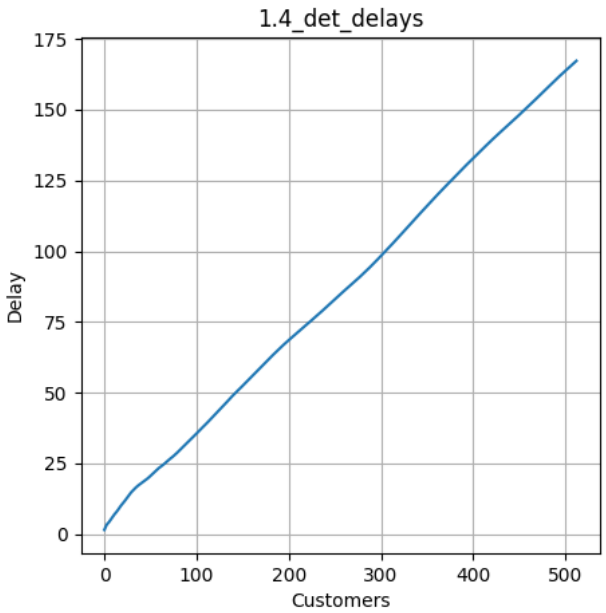
(figure4 Plot Function)

Part 3 Results

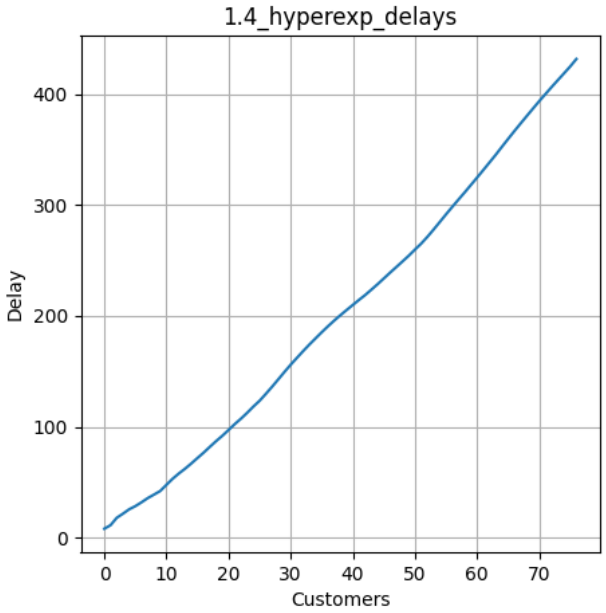
3.1. Curve



(figure5 result of exp curve)



(figure6 result of exp curve)



(figure7 result of exp curve)

3.2. Conclusion

These images show the number of customers versus delay under different conditions.

It can be seen that with the growth of customers, they all show a linear relationship with different rates of change.

But the performance of the index service time is due to the Det time.