Data Structures

Bonus Project

Dr/Ossama Ismail

Dr/Mohamed Alhabrouk

Server Queue

Simulation(2 computers)

Classes:

Data class ~ an object that has all the information of each user

Node class ~ a node object that represents each user, Contains a Data object and Node pointer

Queue ~ a linked list implementation of the queue simulating the server queue system

MainClass ~ the main function of the program

Important Methods

.enqueue() ~ adds a user the server

.timedEnqueue() ~ runs the server for a given time(in seconds)

.print() ~ prints the entire server log excluding the initial server node

.delaySum() ~ returns the total waiting time for all users of the server

.getRandom() ~ returns a random number using exponential distribution having a given mean value (inversion method)

Overview:

in our implementation each user is represented as a node.

each node has an object of type Data and a Node pointer pointing to the next user.

the Data object in each node contains all the information about the user

(time is measured in seconds)

Interarrival time ~ represents the time taken to arrive in queue after the last user

arrival time ~ time of arrival of the user relative to server start time (server starts at 0 seconds)

service ~ represents the time needed by the user to finish a certain task

delay ~ the time the user had to wait in queue to use one of the 2 computers

worktime A/B ~ represents the time needed for each computer to be available for use

serv A/B ~ a boolean variable that shows if a certain computer is available to the user without needing to wait in queue

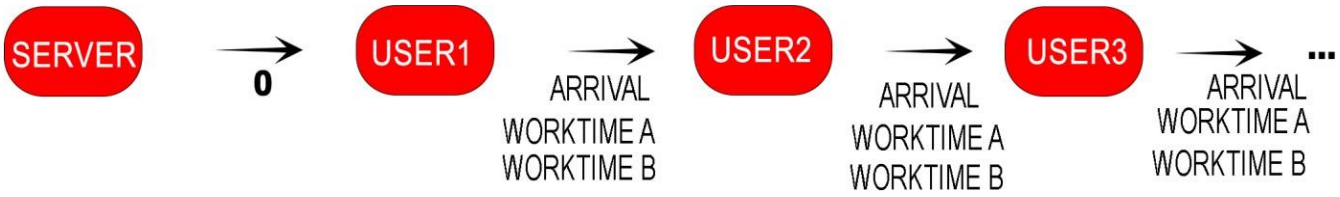
(at least 1 true = no need to wait, if the 2 are false then user needs to wait in queue)

How the program works:

first the program starts the server with initial time 0 via the default constructor

with the front and rear pointers pointing to the same initial server node.

each node then sends the Arrival time, worktime A and B variables to the next node to check whether it has to wait in queue or not, the new node then recalculates the new arrival time and worktime and the rear pointer is moved along the queue.



Running the server for a given time:

by using the timeEnqueue() method we can make the server run for a given time

(in seconds), the method loops creating new users randomly using exponentially distributed random variables (average interarrival = 20 , average service is 25) until the arrival of the last user is greater than the time given to the method which then the method terminates.

Note: the while loop in the method never actually becomes false because the method always terminates before the last iteration to ensure the last user doesn't enter after server closes.

Getting the average delay (waiting time):

we use the method delaySum() which takes a point to the first user of the server and then runs along the entire queue while summing all the delays of the users after which the method returns the sum.

we then use the getNum() static method which returns the static variable in the queue which shows how many users have used the server.

Usage of the program:

the main usage of the program is to show an accurate simulation of the load on a given server running on 2 computers, you just need to set the mean value of the interarrival time and service time to the expected load of the server, the program could also used to determine whether or not a certain server needs to be enlarged to handle extra load.

Further improvements on the program:

now that the main idea of the program was applied, the program could be further improved by increasing the number of computers so that it can be used on a larger scale, the algorithm of the program could be altered to be used in a real server queueing system