Simulation systems types

Introduction:

In a system, too much resource means too high costs. For example, in a computer network, the waiting time of jobs will decrease to zero in theory if the number of servers is larger than the jobs in the system. But it is infeasible because of the high costs of servers. Too little resource is also a problem in a system as it will lead to long waiting times. In [2], it provides an example in the National Health Service (NHS) system. The shortage of bed capacity will lead to high waiting times and delays of the patients. Therefore, we need to make optimal decisions on how much of the resources to deploy and how to access them. In this project, we will focus on a computer system, where jobs compete with each other to access the limited resources in the system [3]. In our case, the limited resource is the servers in the system. We need to find a way to optimize the resources (servers) in the system and figure out how the jobs should we assigned to the servers.

1- Queuing Systems

Queueing theory is the theoretic tool to model the resource sharing system and to optimize their use. Queueing theory is mainly seen as a branch of applied probability theory [4]. Its applications can be found in different areas,

including transportation system [5], the telecommunication applications [6], call centers [7], manufacturing [8], and computer system [9]. In this thesis, we mainly focus on the queueing system in a test execution engine that distributes test tasks among servers. Queues are common in computer systems. Typically, a queue has one service facility and a waiting room [9]. There may be one or more servers in the service facility. In general, a waiting room in the queue can be of finite or infinite capacity. The infinite waiting room means that the number of jobs waiting in the line will not be limited. Not all jobs in the queueing system need to be treated equally. The queueing systems in which some jobs get preferential treatment are called priority queueing systems [13]. In a priority queueing system, the queues are ordered and the higher priority jobs will be served first. It is assumed that all the jobs are divided into different priority classes, which are numbered from 1 to n. The higher priority class is denoted by the smaller number. For instance, the job with priority 1 will be handled before the job with priority 10. But for the jobs with the same priority, it still follows the service discipline FIFO (First-in, Firstout).

There are two basic classes of priority policies: the preemptive-resume policy, and the non-preemptive priority policy. The preemptive-resume policy means that a job of higher priority class has the right to interrupt the service of a lower

priority job [14]. On the other hand, the non preemptive priority policy means that when a server begins to handle a job, this process will not be stopped until the completion of this job. When applying the non-preemptive priority policy, the higher priority job can not interrupt another job's process. The service discipline indicates the manner in which the units are taken for service [15]. There are several different queueing disciplines: FIFO (First-in, First-out): The jobs will be served in the order of they arrive in the system. • LIFO (Last-in, First-out): The jobs will be served in the reverse order of they arrive in the system.

• SIRO (Service In Random Orer): The jobs will be served in random order.

• EDF (Earliest Deadline First): The job with the earliest deadline will be served first.

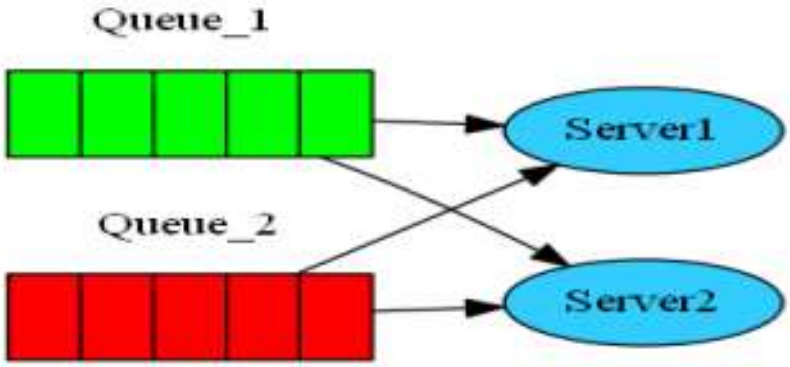
Flexible queuing systems:

The flexible queueing system is a queueing system with multiple classes of jobs and heterogeneous servers where jobs have the flexibility of being processed by more than one server and server posses the capability of processing more than one job class. Figure 2-1 provides an example of the

queueing system with full flexibility, which is a special kind of flexible queueing system. A queueing system with full flexibility means that in this system.

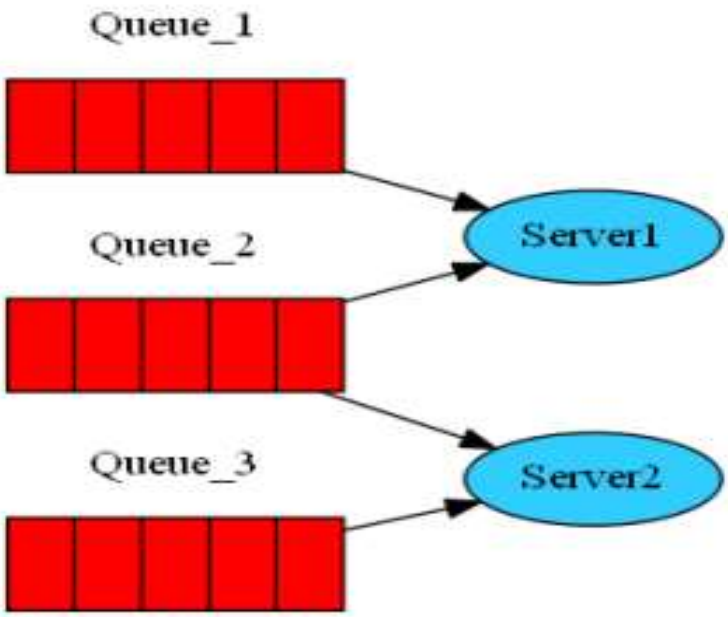
Each server has the capability of processing any job class.

It can be seen that server 1 and server 2 can handle the jobs from queue 1 and queue 2.



The job can be processed by one or more servers.

The job from queue 3 can only be processed by server 2 while the job from queue 3 can be processed by server 1 and server 2.



Discrete event simulation

Modeling and simulation are general tools of engineering. In practice, simulation is needed because in some cases, experimenting with the real-life system is not feasible due to the budget or the risks and analytic modeling is mathematically challenging to obtain. A general idea to build a simulation system is provided in. Firstly, we should select a source system, which is the real world system we are interested in. Second, we need to collect the data and behaviour from the source system,which will be fundamental for simulation. In our case, the database provided by Ericsson contains the information collected from the source system. The final step is to build a simulator. The simulator is the core part of the simulation framework, which has a similar behaviour as the source system. Discrete-event simulation and continuous simulation is

widely used, but the quantum simulation is a special way of simulation, as it is mainly used in highenergy physics, atomic physics or similar areas. The quantum simulation relies on the quantum computer because of the huge amount of memory required [20]. It is very hard to perform the quantum simulation on the classical computer. Continuous simulation is suitable for systems in which the states can change continuously [21]. In the continuous simulation, a continuous function will be applied using real numbers to represent a continuously changing system, which means that we can track the variables in continuous time. On the other hand, discrete event simulation is suitable for problems in which states change in discrete times and by discrete steps. In the discrete event simulation, the source system will be modelled as a sequence of events ordered by the event occurrence time. The state of the system will not change between two adjacent events.

Queuing system examples:

In a simple single server queue, we assume an infinite calling population from which people enter our system. We assume that the inter-arrival times of the customers and their service times (in minutes) are known(deterministic) and given in the following table:

Customer	Inter-arrival Time	Service Time
1	-	2
2	2	1
3	4	3
4	1	2
5	2	1
6	6	4

Duo to its fairness, we assume a FIFO (First In First Out) service policy, i.e. the customers are served with respect to their arrival order. Ex: Create a simulation system that contains for each customer: the arrival time, the time that service begins, the time that service ends (the departure time), the time spent in the queue and in the system by customers.

Solution:

We display the simulation system through the following table:

Customer	Inter-arrival time	Arrival time	Time service begins	Service time	Time service ends	Time spent in queue	Time spent in system
1	-	0	0	2	2	0	2
2	2	2	2	1	3	0	1
3	4	6	6	3	9	0	3
4	1	7	9	2	11	2	4
5	2	9	11	1	12	2	3
6	6	15	15	4	19	0	4
Average						0.67	2.83