NAME: AHMED LATEEFA OMOTOLA

MATRIC NUMBER: 16/SCI01/004

COURSE CODE: CSC 410

COURSE TITLE: Computer System Performance Evaluation

ASSIGNMENT

QUESTIONS

1. Explain the concepts of operational laws as applied to computer and network system performance evaluation.

2. Exhaustively describe at least eight operational laws that are widely employed in computer system performance evaluation.

3. Distinguish between the Forced Flow Law and the Residence Time Law from a systems perspective (not by definition).

4. Discuss some basic queuing models and basic queuing disciplines.

5. Discuss how to resolve some basic queuing problems.

ANWSERS

1. A number of laws are derived which establish relationships between throughput, response time, device utilization, space-time products and various other factors related to computer system performance. These laws are obtained through the operational method of computer system analysis. The operational method, which is formally introduced in this paper, differs significantly from the conventional stochastic modelling approach and is based on a set of concepts that correspond naturally and directly to observed properties of real computer systems. The operational laws presented in this paper apply with complete precision to all collections of observational data, and they are similar to fundamental laws found in other areas of engineering and applied science.
2. Operational Laws
3. Little’s Law: The best known and most commonly used operational law is Little’s law. It is named after the man who published the first formal proof of the law in 1961, although it had been widely used before that time. Little’s law is usually phrased in terms of the jobs in a system and relates the average number of jobs in the system N to the residence time W, the average time they spend in the system. Given a computer system, Little’s law can be applied at many different levels: to a single resource, to a subsystem or to the system as a whole. A little care may be necessary if the law is applied in this way, as the definitions of the number of jobs, throughput and residence time used at the different levels must be compatible with each other.
4. Forced Flow Law: It is often natural to regard a system as being made up of a number of devices or resources. Each of these resources may be treated as a system in its own right as far as the operational laws are concerned, with the rest of the system forming the environment of that resource. A request from the environment generates a job within the system; this job may then circulate between the resources until all necessary processing has been done; as it arrives at each resource it is treated as a request, generating a job internal to that resource.
5. Utilisation Law: If we know the amount of processing that each job requires at a resource then we can calculate the utilisation of the resource. Let us assume that each time a job visits the ith resource the amount of processing, or service, time it requires is Si. Note that service time is not necessarily the same as the residence time of the job at that resource: in general, a job might have to wait for some time before processing begins.
6. General Residence Time Law: One method of computing the mean residence or response time per job in a system is to apply Little’s law to the system as a whole. However, if the mean number of jobs in the system, N, or the system level throughput, X, are not known an alternative method can be used. Applying Little’s law to the ith resource we see that Ni = XiWi, where Ni is the mean number of jobs at the resource and Wi is the average response time of the resource. From the forced flow law, we know that Xi = XVi. Thus, we can deduce that Ni/X = ViWi. The total number jobs in the system is clearly the sum of the number of jobs at each resource, i.e. N = N1 + · · · + NM if there are M resources in the system.
7. Interactive Response Time Law: The name of this law dates back to the time when most of the systems which were being modelled were mainframes processing both interactive jobs and batch jobs. The think time, Z, was quite literally the length of time that a programmer spent thinking at his terminal before submitting another job. More generally interactive systems are those in which jobs spend time in the system not engaged in processing or waiting for processing: this may be because of interaction with a human user or may be for some other reason. For example, if we are studying a cluster of PCs with a central file server to investigate the load on the file server, the think time might represent the average time that each PC spends processing locally without access to the file server. At the end of this nonprecessing period the job generates a fresh request. The key feature of such a system is that the residence time can no longer be taken as a true reflection of the response time of the system. The think time represents the time between processing being completed and the job becoming available as a request again.
8. Bottleneck analysis: The resource within a system which has the greatest service demand is known as the bottleneck resource or bottleneck device, and its service demand is maxi {Di}, denoted Dmax. The bottleneck resource is important because it limits the possible performance of the system. This will be the resource which has the highest utilisation in the system. The residence time of a job within a system will always be at least as large as the total amount of processing that each job requires—this will be the time that the job takes even if it never has to wait for a resource.
9. The Flow Balance Assumption: Frequently it will be convenient to assume that systems satisfy the flow balance property, namely, that the number of arrivals equals the number of completions, and thus the arrival rate equals the throughput. The flow balance assumption can be tested over any measurement interval, and it can be strictly satisfied by careful choice of measurement interval. When used in conjunction with the flow balance assumption, little’s law and the forced flow law allow us to calculate device utilizations for systems whose workload intensities are described in terms of an arrival rate.
10. Residence time, service demand, contention- The residence time of a job within a system will always be at least as large as the total amount of processing that each job requires. The total amount of processing that a job requires is D, the total service demand M, D = \ Di, i =1

In general, there will be some contention in the system meaning that jobs have to wait for processing so the residence time will be larger than this, i.e. W ≥ D

1. Forced Flow Law: It is often natural to regard a system as being made up of a number of devices or resources. Each of these resources may be treated as a system in its own right as far as the operational laws are concerned, with the rest of the system forming the environment of that resource. A request from the environment generates a job within the system; this job may then circulate between the resources until all necessary processing has been done; as it arrives at each resource it is treated as a request, generating a job internal to that resource While Residence Time Law: One method of computing the mean residence or response time per job in a system is to apply Little’s law to the system as a whole. However, if the mean number of jobs in the system, N, or the system level throughput, X, are not known an alternative method can be used. Applying Little’s law to the ith resource we see that Ni = XiWi, where Ni is the mean number of jobs at the resource and Wi is the average response time of the resource.
2. Basic Queuing Disciplines-
	1. First-in-first-out (FIFO)- this means the oldest inventory items are recorded as sold first but do not necessarily mean that the exact oldest physical object has been tracked and sold. In other words, the cost associated with the inventory that was purchased first is the cost expensed first.
	2. Last-in-first-out (LIFO)- this describes a method for accounting for inventories. Under this system, the last unit added to an inventory is the first to be recorded as sold.
	3. Service in random order (SIRO)- Under this type of queue structure, the customer is chosen for service randomly and hence all the customers are equally likely to be selected. Therefore, the time of arrival of the customer has no consequence on the selection of the customer.
	4. Shortest processing time first (SPT)- Its principle is to order jobs according to their duration and schedule them by beginning by the shortest.

Basic Queueing Models

a. Calling population: the population of potential customers, may be assumed to be finite or infinite.

I. Finite population model: if arrival rate depends on the number of customers being served and waiting, e.g., model of one corporate jet, if it is being repaired, the repair arrival rate becomes zero.

II. Infinite population model: if arrival rate is not affected by the number of customers being served and waiting, e.g., systems with large population of potential customers.

b. System Capacity: a limit on the number of customers that may be in the waiting line or system.

I. Limited capacity, e.g., an automatic car wash only has room for 10 cars to wait in line to enter the mechanism.

II. Unlimited capacity, e.g., concert ticket sales with no limit on the number of people allowed to wait to purchase tickets.

C. Arrival Process: In terms of inter-arrival times of successive customers.

I. Random arrivals: inter-arrival times usually characterized by a probability distribution.

II. Poisson arrival process (with rate λ), where An represents the inter-arrival time between customer n − 1 and customer n, and is exponentially distributed (with mean 1/λ).

III. Scheduled arrivals: inter-arrival times can be constant or constant plus or minus a small random amount to represent early or late arrivals.

D. Arrival Processes - Finite population models: Customer is pending when the customer is outside the queueing system, e.g., machine-repair problem: a machine is “pending” when it is operating, it becomes “not pending” the instant it demands service form the repairman.

1. This could be by evaluating the system’s performance by either the measurement, simulation or analytical technique;
2. Measurement; measurement of the real system can be undertaken if the real system is accessible and the process of measurement will not disrupt the system or services. The impact of these disturbances or instructions are very serious.
3. Simulation; this technique drops from discreet event simulation philosophies a simplified model of the system and its nodes are implemented in a software and the performance of interest is measured as of a real system. But measurement side effects are usually not present.
4. Analytical; a mathematical model of the system is analysed numerically. This technique is viewed as a special form of simulation, it is often much quicker than simulation, but most times wide assumption needs to be made for numerical processes to be applicable. Analytical methods are usually used to gain insight during development phases and also to learn fundamental facts about a system.