NAME: GBENRO TIMOTHY

MATRIC NO: 15/ENG02/027

COURSE: CSC 410 – COMPUTER SYSTEM PERFORMANCE EVALUATION

ASSIGNMENT: Assignment 1

QUESTION

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.
6. You have been presented with some systems performance evaluation report; the first was done using measurement technique only, the second was done with simulation technique only, the third was done using analytical technique only, then the fourth was done using measurement and simulation technique only, while the fifth was done using simulation and analytical technique only and the sixth was done measurement and analytical technique only. You are the director of information technology infrastructure in your firm and your firm is about to acquire the information infrastructure concerned in this report.
    a. What specific motive will you consider imperative in general?
    b. Why do you consider this metric important?
    c. If they are not in this report, what will you do in such a report?
    d. If they are part of the report, what is the first action to take with that report and why?
    e. Which reports or combination of report will you adopt and why?

ANSWER

1. Several 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 by using the operational method of computer system analysis. The operational method, which differs significantly from the conventional stochastic modeling approach, is based on a set of concepts that corresponds naturally and directly to observed properties of real computer systems. Operational laws are simple equations which may be used as an abstract representation or model of the average behavior of almost any system. The laws are very general and make almost no assumptions about the behavior of the random variables characterizing the system. These laws are simple and this means that they can be applied quickly and easily by almost anyone.

2.

   a. **Little's Law**
      The best known and most 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. Let X be the throughput, as above. Then Little's law states that **N = XW**
      The average number of jobs in a system is equal to the product of the throughput of the system and the average time spent in that system by a job. 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. At different levels of detail, different definitions of "request" are appropriate. For example, when considering a disk, it is natural to define a

request to be a disk access, and to measure throughput and residence time on this basis. When considering an entire transaction processing system, on the other hand, it is natural to define a request to be a user-level transaction, and to measure throughput and residence time on this basis. Each such transaction may generate several disk accesses. We will return to this idea of systems, or subsystems, within a system in the following subsection.

Example: Consider a disk that serves 40 requests/second (X = 40) and suppose that on average there are 4 requests present in the disk system (waiting to be served or in service) (N = 4). Then Little's law tells us that the average time spent at the disk by a request must be 4/40 = 0.1 seconds. If we know that each request requires 0.0225 seconds of disk service we can then deduce that the average queueing time is 0.0775 seconds.

b. **Forced Flow Law**

It is often natural to regard a system as being made up of several devices or resources. Each of these resources may be treated as a system 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.

Suppose that during an observation interval we count not only completions external to the system, but also the number of completions at each resource within the system. We define the visit count, V_i, of the $_i$th resource to be the ratio of the number of completions at that resource to the number of system completions $\mathbf{V_i \equiv C_i/C}$. More intuitively, we might think of this as the average number of visits that a system-level job makes to that resource. For example, if, during an observation interval, we measure 10 system completions and 150 completions at a specific disk, then on the average each system-level request requires 15 disk operations. The forced flow law captures the relationship between the different

components within a system. It states that the throughputs or flows, in all parts of a system must be proportional to one another. In other words, it relates the throughput at the individual resources (**$X_i = C_i/T$**) to the throughput at the complete system (**$X = C/T$**). It is stated as follows **$X_i = XV_i$**
The throughput at the ith resource is equal to the product of the throughput of the system and the visit count at that resource. An informal interpretation of this law is that, since the visit count defines the number of visits to a resource or device that each job needs in order to complete its processing, the resource must keep up a correspondingly scaled completion rate to ensure that the system completion rate is maintained.
Example: Consider a robotic workcell within a computerised manufacturing system which processes widgets. Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press. We know that the lathe processes 8 widgets in a minute and we want to know the throughput of the press. The throughput of the workcell will be proportional to the lathe throughput, i.e. $X = X_{lathe}/V_{lathe} = 8/4 = 2$. The throughput of the press will be $X_{press} = X \times V_{press} = 2 \times 2 = 4$. Thus the press throughput is 4 widgets per minute.

c. **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 $_i$th resource the amount of processing, or service, time it requires is S_i. 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. The total amount of service that a system job generates at the $_i$th resource is called the service demand, D_i:
$D_i = S_iV_i$
The utilisation of a resource, the percentage of time that the ith resource is in use processing to a job, is denoted Ui. The utilisation law states that

$\mathbf{U_i = X_i S_i = X D_i}$

The utilisation of a resource is equal to the product of the throughput of that resource and the average service requirement at that resource.

Example: Consider again the disk that is serving 40 requests/second, each of which requires 0.0225 seconds of disk service. The utilisation law tells us that the utilisation of the disk must be 40×0.0225 = 90%.

d. **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 $_i$th resource we see that $N_i = X_i W_i$, where N_i is the mean number of jobs at the resource and W_i is the average response time of the resource.

From the forced flow law we know that $\mathbf{X_i = XV_i}$**.** Thus we can deduce that

$\mathbf{N_i/X = V_i W_i}$**.**

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. We know from Little's law that **W = N/X** and from this we arrive at the general residence time, or general response time law:

$$W = \sum_{i=1}^{M} Wi\ Vi$$

The average residence time of a job in the system will be the sum of the product of its average residence time at each resource and the number of visits it makes to that resource.

Example: A web service running on an application server requires 126 bursts of CPU time and makes 75 I/O requests to disk A and 50 I/O requests to disk B. On average each CPU burst requires 30 milliseconds (waiting + processing time). Monitoring has shown that the throughput of disk A is 15 requests per second and the average number in the buffer is 4 whilst at disk B the throughput is 10 requests per second and the average number in the buffer is 3. Using Little's Law we calculate the residence time at each of the disks (remembering that the number in the system is the number in the buffer +1):

$W_{diskA} = N_{diskA} / X_{diskA} = 5 /15/1000 = 5000/15$

$W_{diskB} = N_{diskB} / X_{diskB} = 4 /10/1000 = 4000/10$

Then

$W = W_{CPU}V_{CPU} + W_{diskA}V_{diskA} + W_{diskB}V_{diskB} = 30\times126 + (5000/15)\times75 + (4000/10)\times50 = 3780 + 25000 + 20000 =$ 48780milliseconds

e. **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 non processing 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. Thus, the residence time of the job, as calculated by Little's law as the time from arrival to completion, is greater than the system's response time. The interactive response time law reflects this: it calculates the response time, R as follows:

$\mathbf{R = N/X - Z}$

The response time in an interactive system is the residence time minus the think time. Note that if the think time is zero, $Z = 0$ and $R = W$, then the interactive response time law simply becomes Little's law.

Example: Suppose that the library catalogue system, has 64 interactive users connected via web browsers, that the average think time is 30 seconds, and that system throughput is 2 interactions/second. Then the interactive response time law tells us that the response time must be 64/2−30 = 2 seconds.

f. **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 $\max_i\{D_i\}$, denoted D_{max}. 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. The total amount of processing that a job requires is D, the total service demand,

$$D = \sum_{i=1}^{M} D_i$$

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.** $\mathbf{W \geq D}$

The throughput of a system will always be limited by the throughput at the slowest resource (think of the forced flow law); this is the bottleneck device. By the utilisation law, at this resource, let's call it b, $\mathbf{U_b = X_{Dmax}}$. Therefore, since $\mathbf{U_b \leq 1}$

$\mathbf{X \leq 1/D_{max}}$

It follows that if we wish to improve throughput we should first concentrate on this resource—improving throughput at other resources in the system might have little effect on the overall performance. Using Little's law or the interactive response time law, we can derive a tighter bound on the response time which applies when the system is heavily loaded (i.e. the mean number of jobs, N, is high). Applying the interactive response time law to the throughput bound, $\mathbf{X \leq 1/D_{max}}$ we obtain: $\mathbf{R = N/X - Z \geq ND_{max} - Z}$

Applying Little's law we obtain **$W \geq ND_{max}$**. Thus the asymptotic bound for residence time or response time is: **$W \geq \max\{D, ND_{max}\}$**

$R \geq \max\{D, ND_{max} - Z\}$

Similarly the bound on the throughput of an interactive system may be made tighter when the system is lightly loaded (i.e. the mean number of jobs, N, is small). From the interactive response time law:

$X = N/(R + Z) \leq N/(D + Z)$

Applying Little's law (when $Z = 0$) we obtain **$X \leq N/D$.**

$X \leq \min\{1/D_{max}, N/(D + Z)\}$

Notice that the bottleneck depends on both resource parameters (X_i or S_i) and the workload parameters (V_i). If we change the number of visits that each job makes to a resource we might move the bottleneck.

g. **Service Demand Law**

Service demand is a fundamental concept in performance modeling. The notion of service demand is associated both with a resource and a set of requests using the resource. The *service demand*, denoted as D_i, is defined as the total average time spent by a typical request of a given type obtaining service from resource *i*. Throughout its existence, a request may visit several devices, possibly multiple times. However, for any given request, its service demand is the sum of all service times during all visits to a given resource. When considering various requests using the same resource, the service demand at the resource is computed as the average, for all requests, of the sum of the service times at that resource. Note that, by definition, service demand does not include queuing time since it is the sum of service times. If different requests have very different service times, using a multiclass model is more appropriate. In this case, define $D_{i,r}$, as the service demand of requests of class *r* at resource *i*.

Service demands are important because, along with workload intensity parameters, they are input parameters for QN models. Fortunately, there is an easy way to obtain service demands from resource utilizations and system throughput. By multiplying the utilization U_i of a resource by the measurement interval *T* one obtains the total time the resource was busy. If this time is divided by the total number of completed requests, C_0, the average amount of time that the resource was busy serving each request is derived. This is precisely the service demand.

h. **Throughput law:**
Throughput can be best described as the rate at which a system generates its products or services per unit of time. Businesses often measure their throughput using a mathematical equation known as Little's law, which is related to inventories and process time: time to fully process a single product.
Using Little's Law, one can calculate throughput with the equation:

I = R * T
where:

I is the number of units contained within the system, inventory.

T is the time it takes for all the inventory to go through the process, flow time.

R is the rate at which the process is delivering throughput, flow rate or throughput.

If you solve for **R**, you will get: **R = I / T**

3. Residence time is quantified in terms of frequency distribution of the residence time in the set (mean residence time) and there is focus on things being discrete while
Focused flow relates throughputs at individual resources within a system to the overall system throughput and there is focus on continuity.

4. The queue discipline is the method by which customers are selected from the queue for processing by the service mechanisms

(also called *servers*). The queue discipline is normally first-come-first-served (FCFS), where the customers are processed in the order in which they arrived in the queue, such that the head of the queue is always processed next. Various queuing disciplines can be used to control which packets get transmitted (bandwidth allocation) and which packets get dropped (buffer space). The queuing discipline also affects the latency experienced by a packet, by determining how long a packet waits to be transmitted. Examples of the common queuing disciplines are first-in- first-out (FIFO) queuing, priority queuing (PQ), and weighted-fair queuing (WFQ).

The idea of FIFO queuing is that the first packet that arrives at a router is the first packet to be transmitted. Given that the amount of buffer space at each router is finite, if a packet arrives and the queue (buffer space) is full, then the router discards (drops) that packet. This is done without regard to which flow the packet belongs to or how important the packet is.

Priority Queuing is a simple variation of the basic FIFO queuing. The idea is to mark each packet with a priority; the mark could be carried, for example, in the IP Type of Service (ToS) field. The routers then implement multiple FIFO queues, one for each priority class. Within each priority, packets are still managed in a FIFO manner. This queuing discipline allows high- priority packets to cut to the front of the line.

The idea of the fair queuing (FQ) discipline is to maintain a separate queue for each flow currently being handled by the router. The router then services these queues in a round- robin manner. WFQ allows a weight to be assigned to each flow (queue). This weight effectively controls the percentage of the link's bandwidth each flow will get. We could use ToS bits in the IP header to identify that weight.

Last-come-first-served (LCFS) - the customers are processed such that the customer at the back of the queue is always processed next.

Random selection - customers are selected from the queue randomly when another customer is required for processing.

Pre-emption selection - customers can be pre-empted during service if a higher priority customers joins the queue, and their service suspended or terminated.

5. In a computer network, queuing problems may involve the router and the transmissions it receives: If the traffic is more than the router can process efficiently, packets back up just like customers in a checkout line. If the computer runs multiple operations that demand more service from the central processor unit than it can provide efficiently, that's another type of queuing problem; if a database receives more calls for information than it can handle, that also creates a queue.

   Models

   The goal of queuing theory is to develop formulas that predict the amount of service needed to eliminate queues without the service sitting idle a lot of the time. The first step is to develop a model for the system in question. All queuing models include a representation of the service - cashiers or the router, for instance - and the probable demands on the service at any given time. The level of demand varies not only with the number of requests for service but how long each request takes to process.

   Calculations

   Queuing theory involves a number of calculations. One of the simpler ones is Little's Theory, which states that the number of customers on hand at a given time depends on the rate at which they arrive, multiplied by the time it takes to process them. If a network bottleneck causes a router to take twice as long forwarding data packets but the packets still arrive at the same rate, the number of data packets the router deals with at one time is now double. That often causes a backlog until someone resolves the problem or the arrival rate slows.

6a**.** To check that the report provided is authentic and is correct based on the established standards for the chosen technique.

6b**.** It is important because the content of the report will be of no use and importance if the content itself is not accurate and standard.

6c**.** I will make have to make do with assumptions.

6d**.** To proof that the reports findings are accurate by cross referencing with other established rules and making use of other methods to confirm the results.

6e**.** I will adopt the sixth report which was done using measurement and analytical technique because no method is sufficient in itself and it may not be accurate until proved otherwise by some other method.