Name: OTEGBEYE OLORUNLEKE

Matric no: 16/sci01/038

Course code: CSC 410

1. Operational laws are simple equations which may be used as an abstract representation or model of the average behavior of almost any system. One of the advantages of the laws is that they are very general and make almost no assumptions about the behavior of based on a few simple observations of the system the performance analyst can, by applying these simple laws, derive more information. The foundation of the operational laws are observable variables. These are values which we could derive from watching a system over a finite period of time. We assume that the system receives requests from its environment.

2a.Little’s Law

The utilization law in fact is a special case of Little’s law, which we now will derive in a more general setting. Each step in the higher step function signifies the occurrence of an arrival at that instant; each step in the lower signifies a completion. At any instant, the vertical distance between the arrival and completion functions represents the number of requests present in the system. Over any time interval, the area between the arrival and completion functions represents the accumulated time in system during that interval, measured in request-seconds (or request-minutes, etc. Little’s law is important for three reasons. First, because it is so widely applicable (it requires only very weak assumptions), it will be valuable to us in checking the consistency of measurement data. Second, in studying computer systems we frequently will find that we know two of the quantities related by Little’s law (say, the average number of requests in a system and the throughput of that system) and desire to know the third (the average system residence time, in this case. Third, Little’s law is central to the algorithms for evaluating queueing network models.

b. The Forced Flow Law

In discussing Little’s law, we allowed our field of view to range from an individual resource to an entire system. 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 consider- ing an entire system, on the other hand, it is natural to define a request to be a user-level interaction, and to measure throughput and residence time on this basis.

The relationship between these two views of a system is expressed by the forced flow law, which states that the flows (throughputs) in all parts of a system must be proportional to one another. Suppose that during an observation interval we count not only system completions, but also the number of completions at each resource. We define the visit count of a resource to be the ratio of the number of completions at that resource to the number of system completions, or, more intuitively, to be the aver- age number of visits that a system-level request makes to that resource. If we let a variable with the subscript k refer to the k-th resource (a variable with no subscript continues to refer to the system as a whole).

c. 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: A = C, therefore A = X

The flow balance assumption can be tested over any measurement interval, and it can be strictly satisfied by careful choice of measurement inter: val.

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.

D. Utilization Law:

If we know the amount of processing that each job requires at a resource then we can calculate the utilization 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. The total amount of service that a system job generates at the ith resource is called the service demand, Di:

Di = SiVi

The utilisation of a resource, the percentage of time that the ith resource is in use processing to a job, is denoted Ui.

E. 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. 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:

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.

F. Interactive Response Time Law

The name of this law dates back to the time when most of the systems which were being modeled 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 workstations with a central file server to investigate the load on the file server, the think time might represent the average time that each workstation 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.

G. 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 utilization 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 = Mi=1 Di. 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

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.

3. For forced flow law Consider a robotic work cell within a computerized 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 work cell will be proportional to the lathe throughput, i.e. X = Xlathe/Vlathe = 8/4 = 2. The throughput of the press will be Xpress = X × Vpress = 2 × 2 = 4. Thus the press throughput is 4 widgets per minute while in Residence Time law 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.

4. Basic queuing 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.

Basic Queuing Disciplines

a. First-in-first-out (FIFO)

b. Last-in-first-out (LIFO)

c. Service in random order (SIRO)

d. Shortest processing time first (SPT)

e. Service according to priority (PR)

5a. Assess and improve your queue management strategy.

b. Implement digital queuing software.

c. Keep the rules of queuing fair and consistent.

d. Design your space to accomodate queues.

e. Inform customers of the duration of their wait.

f. Distract and entertain customers in a queue.

6i. Don’t trust the result of a simulation model until they have been validated by other performance evaluation techniques

II. It’s important so as to know if the system is correctly defined and if the goals are clearly stated.

iii. Such report is invalid

iv. Go through the report to check if there are any mistakes

V. The forth report (measurement and simulation) because measurement technique requires more effort and it is the most accurate; simulation method requires relatively lesser effort and it is averagely accurate while analytical method is very quick and often very less accurate.