2 Operational Laws

2.1 Introduction

Operational laws are simple equations which may be used as an abstract representation or model of the average behaviour of almost any system. One of the advantages of the laws is that they are very general and make almost no assumptions about the behaviour of the random variables characterising the system[1]. Another advantage of the laws is their simplicity: this means that they can be applied quickly and easily by almost anyone.

Based on a few simple observations of the system the performance analyst can, by applying these simple laws, derive more information. Using this information as input to further laws the analyst gradually builds up a more complete picture of the behaviour of the system. Note that although we will talk in this section about operational laws in the context of systems, the laws are equally applicable to the observations obtained from models and we will have occasion to use the laws in this way later in the course.

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. Each request generates a *job* or *customer* within the system. When the job has been processed the system responds to the environment with the completion of the corresponding request.

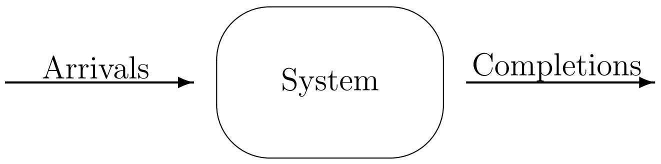

Figure 1: An Abstract System

If we observed such an abstract system we might measure the following quantities:

T, the length of *time* we observe the system;

A, the number of request *arrivals* we observe;

C, the number of request *completions* we observe;

B, the total amount of time during which the system is *busy* ($B \leq T$);

N, the average number of jobs in the system.

From these observed values we can derive the following four important quantities:

$\lambda = A/T$, the *arrival rate*;

$X = C/T$, the *throughput* or *completion rate*,

$U = B/T$, the *utilisation*;

$S = B/C$, the *mean service time* per completed job.

[1]In contrast, Markovian analysis relies on very strong assumptions about the distribution function of the random variables which are used.

We will assume that the system is *job flow balanced.* This means that the number of arrivals is equal to the number of completions during an observation period, i.e. $A = C$. Obviously this assumption is not true in all observation periods, but it is a *testable assumption* because an analyst can always test whether the assumption holds. Note that if the system is job flow balanced the arrival rate will be the same as the completion rate, that is, $\lambda = X$.

2.2 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. Let X be the throughput, as above. Then Little's law states that

$$\boxed{N = XW}$$

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.

2.3 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.

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 ith resource to be the ratio of the number of completions at that resource to the number of system completions

$$V_i \equiv C_i/C.$$

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

$$\boxed{X_i = XV_i}$$

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.

2.4 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 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 ith resource is called the *service demand*, D_i:

$$D_i = S_i V_i$$

The utilisation of a resource, the percentage of time that the ith resource is in use processing to a job, is denoted U_i. The *utilisation law* states that

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

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 $40 \times 0.0225 = 90\%$.

2.5 General Residence Time Law

One method of computing the mean residence 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 $N_i = X_i W_i$, where N_i is the mean number of jobs at the resource and W_i is the average residence time of the resource.

From the forced flow law we know that $X_i = X V_i$. Thus we can deduce that

$$N_i / X = V_i W_i.$$

The total number jobs in the system is clearly the sum of the number of jobs at each resource, i.e. $N = N_1 + \cdots + N_M$ 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 law*:

$$\boxed{W = \sum_{i=1}^{M} W_i V_i}$$

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):

$$W_{diskA} = \frac{N_{diskA}}{X_{diskA}} = \frac{5}{15/1000} = \frac{5000}{15} \qquad W_{diskB} = \frac{N_{diskB}}{X_{diskB}} = \frac{4}{10/1000} = \frac{4000}{10}$$

Then

$$\begin{aligned} W &= W_{CPU} V_{CPU} + W_{diskA} V_{diskA} + W_{diskB} V_{diskB} \\ &= 30 \times 126 + \frac{5000}{15} \times 75 + \frac{4000}{10} \times 50 = 3780 + 25000 + 20000 = 48780 milliseconds \end{aligned}$$

2.6 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:

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

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.

2.7 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.

$$W \geq D$$

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, $U_b = XD_{max}$. Therefore, since $U_b \leq 1$

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

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, $X \leq 1/D_{max}$ we obtain:

$$R = N/X - Z \geq ND_{max} - Z$$

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

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

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 $X \leq N/D$.

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

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.

2.8 Assumptions

As mentioned in the introduction, the operational laws do not rely on many assumptions. Indeed the only explicit assumption we have made is that the system is *job flow balanced*—the same number of requests are completed by the system as arrive at the system. We are also implicitly assuming that this holds at each of the resources or devices within a system. A consequence of this is that jobs are not created or destroyed anywhere in the system. This is sometimes called *conservation of work.*

We have also assumed that the system is *homogeneous*, that is, that the behaviour of jobs or resources within a system does not depend on the global state of the system. For example, a job cannot alter the number of visits it makes to a particular resource because that resource is heavily loaded.

Jane Hillston ⟨Jane.Hillston@ed.ac.uk⟩. January 21, 2017.