Computer Systems Modelling
Operational Analysis

Simon Crosby*

Michaelmas Term 1996

1 Operational Analysis

Operational Analysis is an approach to the modelling of systems which is based on the idea of **observation** rather than a probablistic description of system behaviour. Many results of operational analysis can be derived from a stochastic approach, but relying on observation means that one can have a better feel for the assumptions being made. Operational analysis, or so its proponents argue, makes very weak assumptions about the system being modelled.

Operational analysis involves more than observation. It is also concerned with quantities "directly related" to observables. Some of these direct relations require certain assumptions to be made, others do not.

A very good summary can be found in Leung, an in depth description in Lazowska. Beware the differences in notation.

We will begin by examining some fundamental quantities and operational laws and then move on to some analysis techniques.

1.1 Fundamental Operational Quantities and Laws

Our approach is based on observation. Suppose we examine a system for some time recording the customer arrivals and departures.

We use the following definitions for our observables.

- T is the length of time we observe the system

*These notes also include material by R.J. Gibbens and I.M. Leslie

- A is the number of arrivals observed
- C is the number of departures (or completions of service) observed.

If the system under observation is a single resource, then we can also measure:

- B, the time for which the resource was busy.

From these quantities we can define the following quantities

- Arrival rate, $\lambda \triangleq \frac{A}{T}$
- Throughput, $X \triangleq \frac{C}{T}$.

For a single resource, we can also define:

- Utilization, $U \triangleq \frac{B}{T}$
- Average service requirement, $S \triangleq \frac{B}{C}$

Our first "law" is just an algebraic identity

$$U \triangleq \frac{B}{T} = \frac{C}{T}\frac{B}{C} = XS$$

known as the *Utilization Law*.

Our next law requires a further quantity to be observed

- W, the job-time product which is sum of the durations of all customers in the system over the observation period.

From this we can define

- Average number of requests in the system, $N \triangleq \frac{W}{T}$
- Average residence time per request, $R \triangleq \frac{W}{C}$

Again, simple manipulation gives

$$N \triangleq \frac{W}{T} = \frac{C}{T}\frac{W}{C} = XR$$

which is *Little's Law* (sometimes called *Little's Result*).

Little's Law makes very weak assumptions about the system which makes it applicable to a wide range of systems and can be applied recursively as we break systems down into subsystems and then those subsystems into individual resources.

For example, suppose we examine a system for $T = 10$ seconds and observe 4 customers who spend 10 seconds in the system and one customer who spends 5 seconds then we would have $W = 4 \times 10 + 1 \times 5 = 45$ seconds. If also $A = C = 5$ then $X = C/T = 5/10 = 0.5$

customers per second, $\lambda = 5/10 = 0.5$ customers per second, $N = 45/10 = 4.5$ customers and $R = 45/5 = 9.0$ seconds per customer.

A simple example of this is shown in Figure 1. This is a queueing network model of an interactive system. Each device has its own queue associated with it. We have a fixed number, M, of users logged on. We consider a customer to be at the terminal if they are thinking, that is the system is waiting for them to do something. We invent a think time, Z, which is the average time a user spends between receiving a prompt (say) and typing a command in (say). Otherwise a customer is somewhere inside the central subsystem either being served or waiting to be served.

We first consider the central subsystem, that is, the system without the terminals. The population of customers in the central subsystem is finite (but variable) since there is a fixed number of users logged on.

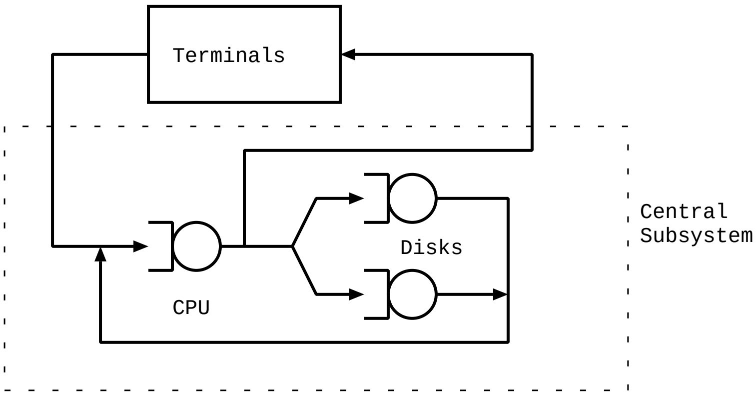

Figure 1: Simple Interactive System

We can use Little's Law to relate some observables in the central subsystem, but we should be careful to understand what the quantities are.

- N is the number of customers in the central subsystem, clearly between 0 and M
- X is the rate at which customers complete interactions with the central subsystem
- R is the average time a customer spends in the central subsystem which corresponds to our notion of response time.

So, for example if we observe that system throughput is 0.5 interactions per second, and we find on average 7.5 users in the subsystem (that is they are not "thinking") then from Little's Law,

$$R = \frac{N}{X} = \frac{7.5}{0.5} = 15 \text{ seconds}.$$

If we consider the whole system including the terminals, then our system throughput remains the same, but our N is now the total number of users, that is $N = M$, and residence time is now response time plus think time.

If we now totally abuse our notation (at least we are being honest!) as others do, and let R be response time rather than residence time, we get the interactive system version of Little's Law

$$M = X(R + Z)$$

where R is the response time and Z is the think time. The residence time is $R + Z$ (which we were calling R above).

So, if we had 10 users logged on, 5 seconds average think time and an average response time of 15 seconds,

$$X = \frac{M}{R+Z} = \frac{10}{15+5} = 0.5 \text{ interactions/second}\,.$$

Note that under heavy load (that is, either M is large or Z is small) the utilization is close to one ($U \approx 1$) so that using the Utilzation Law the throughput, $X \approx \frac{1}{S}$ and hence

$$R = \frac{M}{X} - Z \approx \frac{1}{S}M - Z$$

so that the response time grows approximately linearly with the number of users M.

1.2 Visit Counts and Forced Flow

As our example above shows, we are in a strong position to deal with multiple servers, each having their own queue. We can expand our notation so that if we number our devices, we denote the throughput at a device i as X_i. The same applies to the other quantities. We shall suppose that there are $i = 1, 2, \ldots, K$ devices.

Our trivial example above shows that we can relate what is happening at the larger system level to what is happening inside the system. However, by making an assumption about customer behaviour we can relate this to system behaviour to derive behaviour in a stronger way.

Our assumption, and it is a huge assumption, is that the service required by a customer or request or job is an inherent property of the customer not of the state of the system.

(It is important to note that this does not assume that servers are load independent. For example, a disc controller that orders requests provides higher throughput at the disc as the load increases. As long as the performance of a server depends on the load at **that** device rather than the load elsewhere in the system our assumptions will hold.)

A **visit count** for a device is the number of completions at that device for every completion from the system

$$V_i \triangleq \frac{C_i}{C}$$

where C_i is the number of completions at device i. Since $X = \frac{C}{T}$, we have

$$X_i = \frac{C_i}{T} = \frac{C_i}{C}\frac{C}{T} = V_i X$$

which is known as the **Forced Flow Law** but which might be better called the Forced Flow Assumption.

What it is really saying is that given the characteristics of requests, V_i, and the throughput of the system, X then we know the throughput at every device, X_i. Of course, characterising requests is a nontrivial problem.

If devices are load independent, then we can define the service demand a customer makes on a device i,

$$D_i \triangleq V_i S_i\,,$$

and note

$$U_i = X_i S_i = (X V_i) S_i = X(V_i S_i) = X D_i\,.$$

Thus given the V_i and S_i for customers and the system throughput, X, we can determine the utilization at each device.

If we also know all the N_i, the average number of customers at device i, then we can apply Little's Law at each device

$$N_i = R_i X_i \Longrightarrow R_i = \frac{N_i}{X_i}$$

where R_i is the residence time a job spends at each visit to device i.

Summing the residence times at all devices gives

$$R = \sum_{i=1}^{K} V_i R_i = \sum_{i=1}^{K} \frac{V_i N_i}{X_i}$$

the residence time in the whole system, which might correspond to our notion of response time.

We can infer X_i from U_i (and vice versa) but can not find R_i without further assumptions on the service time distribution.

We can approximate it by assuming that a customer finds N_i customers at device i and must wait for all of them to be served for the average service time, S_i and then be served itself in time S_i. This is called a memoryless approximation (why?) leading to R_i^*

$$\begin{aligned} R_i^* &= N_i S_i + S_i \\ &= R_i^* X_i S_i + S_i \\ &= R_i^* U_i + S_i \end{aligned}$$

(where we have used Little's Law and the Utilization Law). Reducing this we get

$$R_i^* = \frac{S_i}{1 - U_i}$$

and

$$\begin{aligned} N_i &= R_i^* X_i \\ &= \frac{S_i X_i}{1 - U_i} \\ &= \frac{U_i}{1 - U_i} \end{aligned}$$

which looks remarkably like the M/M/1 queueing system.

1.3 Bottleneck Analysis

A bottleneck in a system is a hindrance to movement or progress. If the forced flow assumption holds, then we can show that at high loads, system performance is determined by the device with the highest utilization: the bottleneck.

The ratio of the completion rates of any two devices is, using the Forced Flow Law

$$\frac{X_i}{X_j} = \frac{V_i X}{V_j X} = \frac{V_i}{V_j}$$

given by the ratio of their visit ratios and since $U_i = X_i S_i$, we have a similar property for utilizations

$$\frac{U_i}{U_j} = \frac{X_i S_i}{X_j S_j} = \frac{V_i S_i}{V_j S_j}\,.$$

If the V_i are intrinsic properties of customers and each S_i is independent of the queue length at device i then the system has load independent parameters. In such cases, the throughput and utilization ratios are the same for all loads on the system. This observation can be used to determine asymptotes (as N increases) for system throughput X, and response time R. (Recall that symbols without subscripts denote quantities for the entire system.)

In general, $U_i \leq 1$ and $X_i \leq \frac{1}{S_i}$. We say a device i is **saturated** if its utilization approaches 100%, that is $U_i \to 1$. As $U_i \to 1$, we have that $X_i \to \frac{1}{S_i}$. In other words, the device is completing work at its capacity at an average of one request in every time period S_i. Devices which operate near full utilization tend to have very long queues; a saturated device is a bottleneck device. We will use the subscript b to denote a device capable of saturating.

Every network has at least one bottleneck, for as the number of customers in the system, N, becomes large, the jobs will form at least one long queue. Since the utilization ratios are fixed, the device i with the largest V_iS_i will be the first to achieve 100% utilization as N increases:

$$V_bS_b = \max\{V_1S_1, ..., V_KS_K\}$$

so that the bottleneck(s) is (are) determined by device (the S_i) and workload (the V_i and S_i) properties.

As N becomes large, we have that $U_b \to 1$ and $X_b \to \frac{1}{S_b}$. Now, $XV_b = X_b$ so that,

$$X_{\max} = \frac{X_b}{V_b} \to \frac{1}{V_bS_b}$$

is the maximum possible value of system throughput as N increases.

The total of all service required by a job while in the system is given by

$$R_{\min} = \sum_{i=1}^{K} V_iS_i$$

which denotes the smallest possible value of mean response time. In fact, $R_{\min}$ is the response time when $N = 1$; this implies that $X = \frac{1}{R_{\min}}$ when $N = 1$ and that for $N > 1$, $X \leq \frac{N}{R_{\min}}$.

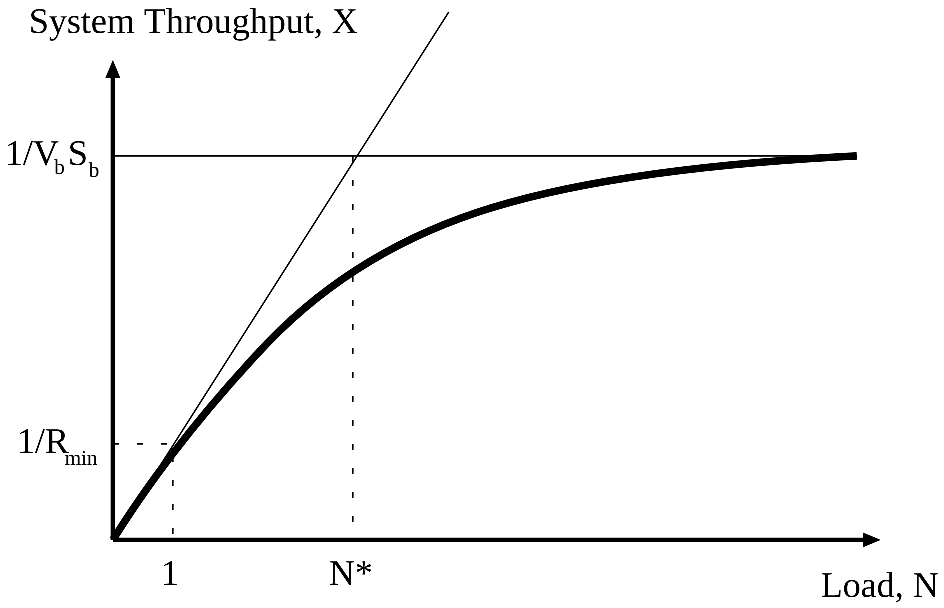

Figure 2: System Throughput

As a function of the load, N, the system throughput, X, rises monotonically from $\frac{1}{R_{\min}}$ at $N = 1$ to the asymptote $\frac{1}{V_bS_b}$ and stays below the straight line $X = \frac{N}{R_{min}}$ as shown in Figure 2.

If we were to hypothesise that k jobs (where $k \leq K$, the total number of devices in the system) always manage to avoid each other in the network so that $X = \frac{k}{R_{\min}}$, we would require that

$$\frac{k}{R_{min}} \leq \frac{1}{V_b S_b}$$

or $k \leq N^*$ where

$$N^* = \frac{R_{\min}}{V_b S_b} = \frac{\sum_{i=1}^{K} V_i S_i}{V_b S_b}\,.$$

Thus, N^* represents a load beyond which queueing is certain to exist somewhere in the network.

Consider these results in terms of response time for the interactive system described earlier, with M terminals and a average think time Z, and the mean response time

$$R = \frac{M}{X} - Z\,.$$

When $M = 1$, we must have $R = R_{\min}$. The throughput, X, cannot exceed $\frac{1}{V_b S_b}$, so

$$R \geq M V_b S_b - Z \geq M V_i S_i - Z, \qquad i = 1, ..., K$$

by the choice of device b and where the first inequality approaches equality for large M.

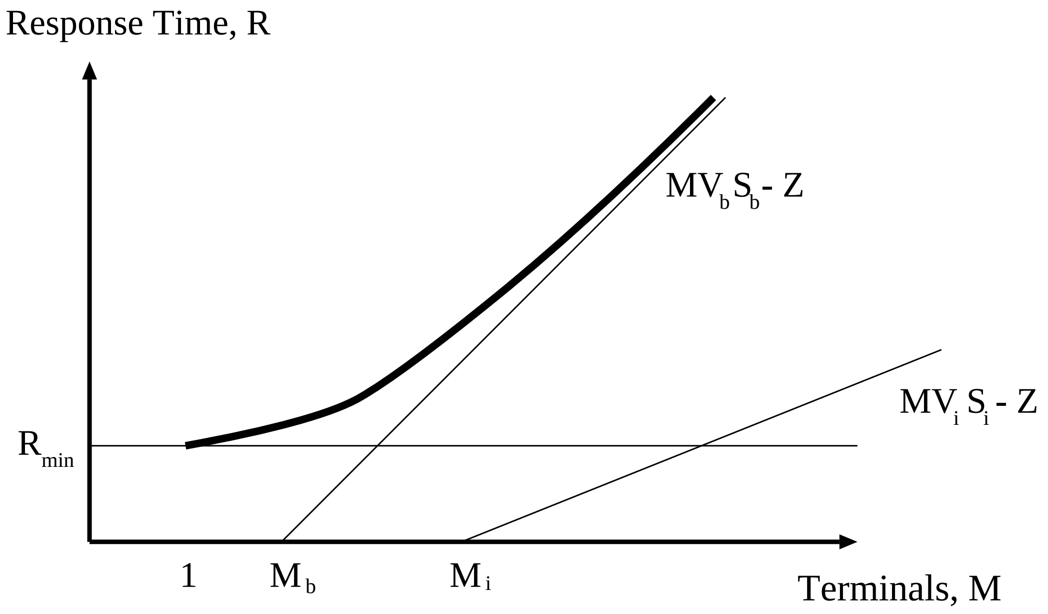

Figure 3: System Response Time

As shown in Figure 3, the response time asymptote intersects the horizontal axis at

$$M_b = \frac{Z}{V_b S_b}\,.$$

The response time asymptote crosses the minimum response time $R_{\min}$ at M_b^* where

$$M_b^* V_b S_b - Z = R_{\min}$$

so that

$$M_b^* = \frac{R_{min} + Z}{V_b S_b} = N^* + M_b\,, \quad \text{say}\,.$$

When there are more than M_b^* terminals, queueing is certain to exist in the central subsystem.

Note that if we let $Z = 0$ then the response times are those of a transaction system.

Figure 3 also shows a bound, $MV_iS_i - Z$ for some arbitrary device i. There is such a bound for each device but all will be below the bottleneck device bound except for the bottleneck itself.

In summary,

- the largest of the products V_iS_i determines the bottleneck device, b.
- the sum of these products determines the smallest possible response time $R_{\min}$.
- queueing cannot be avoided when N exceeds $N^* = \frac{R_{\min}}{V_b S_b}$.
- queueing cannot be avoided in an interactive systems when $N^* + \frac{Z}{V_b S_b}$ terminals are logged on.

Diagrams like Figure 3 may give some gross insight into the effects of proposed changes. Reducing V_iS_i for a device which is not a bottleneck will not affect the bottleneck and will not move the $\frac{1}{V_b S_b}$ asymptote. It may improve $R_{\min}$, but it will not improve response time at high loads. On the other hand, reducing V_iS_i for the bottleneck device(s) will raise the $\frac{1}{V_b S_b}$ asymptote on the throughput graph, lower the $MV_bS_b - Z$ asymptote on the response time graph and reduce $R_{\min}$. However, asymptotes will remain relevant only as long as V_bS_b remains the largest of the V_iS_i; too much improvement just moves the bottleneck elsewhere.

Example: interactive system

Consider the interactive system shown in Figure 1. The visit counts and average service times have been measured as follows and suppose that $Z = 20$ seconds.

number	device	S_i (seconds)	V_i	$D_i = V_iS_i$
1	CPU	0.05	20	1.00
2	disk	0.08	11	0.88
3	fast disk	0.04	8	0.32
			$R_{\min}$	2.20

Question: Is an 8 second response time feasible with 30 users logged on? If not, what changes are required to achieve it?

The V_iS_i products are $V_1S_1 = 1$ second, $V_2S_2 = 0.88$ seconds and $V_3S_3 = 0.32$ seconds. The minimum response time is 2.2 seconds (since $1 + 0.88 + 0.32 = 2.20$) and the CPU is clearly the bottleneck. The response time curve is shown in Figure 4.

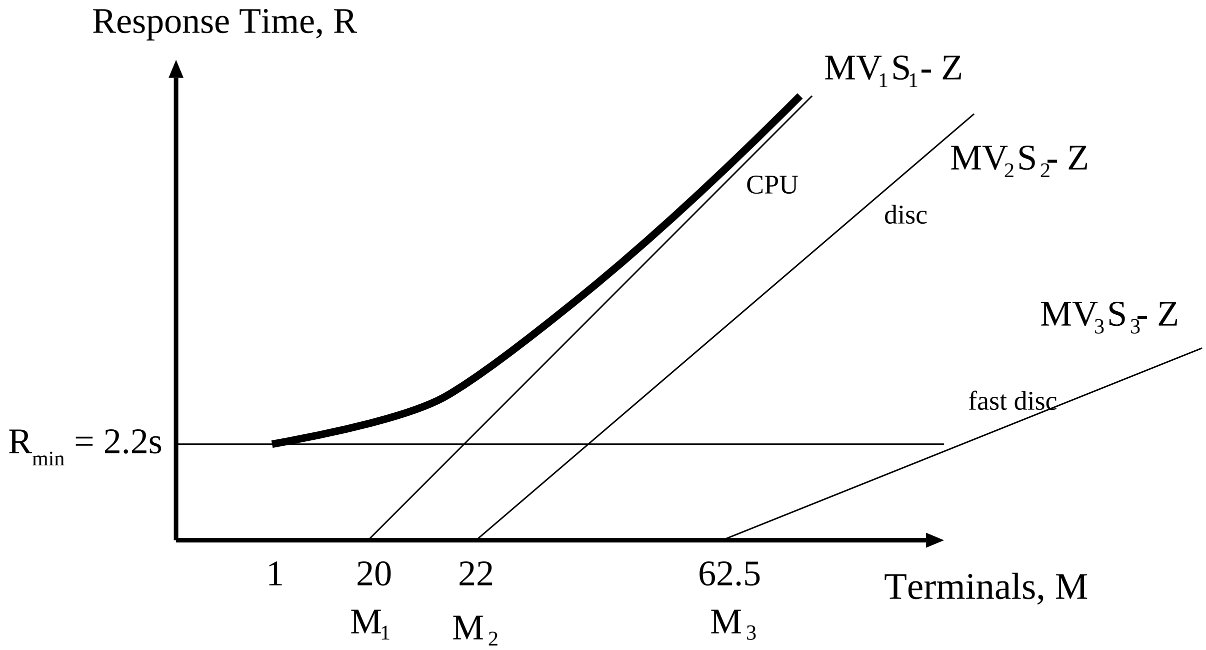

Figure 4: Reponse time for example

For $M = 30$, the response time asymptote requires that

$$R \geq 30 \times 1 - 20 = 10 \quad \text{seconds}\,.$$

So the answer is no. In order to make 8 second response time feasible, we need to speed up the CPU, so that the new service time obeys

$$MV_1S_1' - Z \leq 8$$

or

$$S_1' \leq \frac{20 + 8}{30 \times 20} = .047 \quad \text{seconds}$$

which is a 7% speed up in the CPU. Then $V_1S_1' = .93$ is still the largest product so the CPU is still the bottleneck.

Question: Is 10 second response time feasible when 50 users are logged in? If not, how much CPU speedup is required?

If the CPU becomes extremely fast, that is $S_1 \to 0$, the disk will become bottleneck. We then have

$$R \geq MV_2S_2 - Z\,.$$

For $M = 50$, this is

$$R \geq 50 \times 0.88 - 20 = 24 \quad \text{seconds}$$

so a 10 second response time is not feasible with $M = 50$ and no amount of CPU speedup is capable of achieving it.

1.4 Balanced System Bounds

Bottleneck analysis is an example of asymptotic analysis: as the number of customers grows the system performance approaches one of the asymptotes. There are of course other sorts of bounds; ones which we do not expect system performance to reach but which nonetheless bound performance.

Bottleneck analysis is good at high loads (and indeed at low loads if no queueing is going on). Mid-range is harder. Balanced system bounds provide **tighter** bounds at mid-range loads than bottleneck analysis.

The basic idea is that we compare the system we are modelling with a balanced system. A system is **balanced** if for any load the utilizations of all devices are equal. Thus, in a balanced system all devices approach saturation together. Under certain assumptions balanced systems exhibit the following important property that

$$U_i(N) = \frac{N}{N + K - 1}$$

where N is the number of customers and so the system throughput is given by

$$X(N) = \frac{U_i}{D_i} = \frac{N}{N + K - 1} \times \frac{1}{D_i}\,.$$

So, in a balanced system we only need to know how many customers there are to determine device utilizations. The assumptions required to prove this (which we omit) are that arrivals into the system are not related to the distribution of customers inside the system, that there is flow balance and that a customer sees the time average queue length when arriving at a device.

For example,

$$\begin{array}{lll} N = 1 & K = 2 & U_1 = U_2 = \frac{1}{2} \\ N = 1 & K = 100 & U_1 = U_2 = \ldots = U_{100} = \frac{1}{100} \\ N = 100 & K = 2 & U_1 = U_2 = \frac{100}{101} \\ N = 2 & K = 2 & U_1 = U_2 = \frac{2}{3}\,. \end{array}$$

The approach we take is to look at our system and find $D_{\max}$, $D_{\min}$, D_{av} and $D = D_{\text{tot}}$ which are respectively, the maximum demand at any device, the minimum device demand, the average demand at each device and the total demand across all devices. So, we have $D_{\text{av}} = D/K$. We can then construct imaginary balanced systems which are related to our system.

pess1: balanced system with K devices each with a demand of $D_{\max}$

opt1: balanced system with K devices each with a demand of $D_{\min}$

The throughput of system **pess1** is

$$\frac{N}{N+K-1} \times \frac{1}{D_{\max}}$$

and the throughput of system **opt1** is

$$\frac{N}{N+K-1} \times \frac{1}{D_{\min}}$$

so for the system being modelled we have

$$\frac{N}{N+K-1} \times \frac{1}{D_{\max}} \leq X(N) \leq \frac{N}{N+K-1} \times \frac{1}{D_{\min}}$$

and since $N = XR$,

$$(N+K-1)D_{\max} \geq R(N) \geq (N+K-1)D_{\min}\,.$$

This will give tighter bounds on the mid-range performance than the bottleneck bounds. However, we can do even better using the following line of argument.

Given a system with total load D and K devices what is the best performance that the system can achieve? We claim that the best performance is achieved by spreading the load out evenly among all the devices.

opt2: D_{av} at each of the K devices with throughput

$$\frac{N}{N+K-1} \times \frac{1}{D_{\text{av}}} = \frac{N}{D+(N-1)D_{\text{av}}}\,.$$

What is the worst system that can be obtained subject to the constraints that D and $D_{\max}$ remain fixed? We claim that the worst system is obtained by placing $D_{\max}$ at as many devices as possible and 0 at the rest.

pess2: $D_{\max}$ at each of the $\frac{D}{D_{\max}}$ devices (we shall ignore fractional devices here for simplicity we should really use the ceiling of $\frac{D}{D_{\max}}$) with throughput

$$\frac{N}{N+K-1} \times \frac{1}{D_{\max}} = \frac{N}{D+(N-1)D_{\max}}\,.$$

So that for the system being modelling

$$\frac{N}{D + (N-1)D_{\text{max}}} \le X(N) \le \frac{N}{D + (N-1)D_{\text{av}}}$$

and

$$D + (N-1)D_{\text{max}} \ge R(N) \ge D + (N-1)D_{\text{av}}\,.$$

The throughput curve for the bottleneck and balanced system bounds is shown in Figure 5. The lower balanced system bound asymptotically approaches the bottleneck upper bound at high loads, that is as $N \to \infty$,

$$\frac{N}{D + (N-1)D_{\text{max}}} \to \frac{1}{D_{\text{max}}}\,.$$

Note that at high loads the bottleneck bounds are the limiting high bound, not the balanced system bounds!

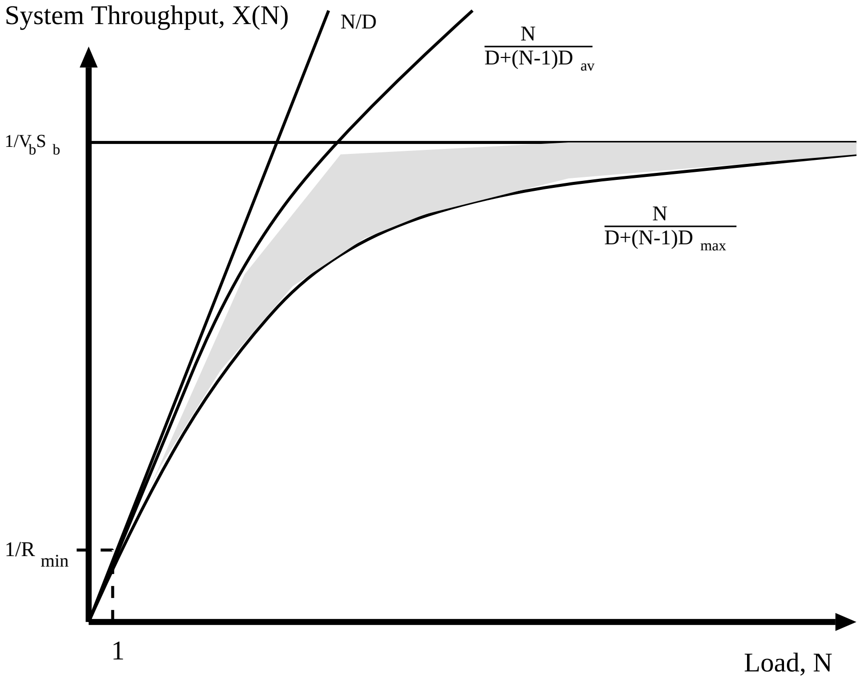

Figure 5: Balanced Systems Bounds

The asymptotic bottleneck bound $\frac{1}{D_{\text{max}}}$ and the optimistic balanced bound intersect at $N^\dagger$ where

$$\frac{1}{D_{\text{max}}} = \frac{N^\dagger}{D + (N-1)D_{\text{av}}}\,.$$

So that,

$$N^\dagger = \frac{D - D_{\text{av}}}{D_{\text{max}} - D_{\text{av}}}\,.$$

1.5 Measurement

A system model will have parameters. In our examination of operational analysis, these parameters are the service demands and visit counts that customers make on and to devices.

When comparing designs one often makes informed guesses about what these parameters might be. When modelling an existing system, however, one observes and measures the system. Even informed guesses are better when they are based on real measurements.

We call devices which measure system performance **monitors**. Monitors may be based on hardware external to the system being measured (a logic analyser for example) or maybe a piece of code running on the system being measured. The terms hardware monitor, software monitor and hybrid monitor are often used — the notion of a hybrid monitor is extremely fuzzy but which might mean the combination of a hardware monitor with code running of the system being measured.

Whatever form a monitor takes, we can describe it in terms of monitor **artifact** and monitor **power**.

Monitor artifact describes the side effects that operation of the monitor has on the system being observed. Hardware monitors should have no artifact, software monitors will have artifact depending on how much processing they do and hybrid monitor artifact will depend on the structure of the monitor.

Monitor power has at least 5 dimensions:

- domain
  - the class of activities observable
- input rate
  - the maximum frequency at which events can be recognized and recorded
- input width
  - the amount of information the monitor can extract and process when a monitored event occurs
- recording capacity
  - the amount of memory available for storing extracted information
  - for hardware monitors (in particular):
    - event activity can be measured by a single counter
    - average duration can be derived from two values: accumulated duration of all occurrences of the activity and the total number of occurrences
    - distributional information requires much more memory

- monitor resolution
  - the resolution of the time clock from which the monitor derives timing information

An Example: CPU Utilization

Many processors actually have a signal which can be monitored when they are idle (as long as they execute a HALT instruction). A hardware monitor can be attached to this signal and a complete distribution of idle time can be built up.

A software monitor can monitor CPU activity by generating interrupts and interrogating the state of the systems to see if, but for the interrupt, the system would be idle.

Alternatively a dummy task, which is always runnable can be run at the lowest priority and the OS statistics can be used to see how much time it spent running. This needs a preemptive scheduling discipline — why?

Dummy tasks can be quite elaborate in order to measure the **effective** CPU which is idle. For example, if there is a lot of DMA activity going on a dummy task which is running out of cache will run quickly; we might be more interested in how quickly a task which has significant cache misses would run. We could tune the behaviour of the dummy task to be like the behaviour of the real tasks.

Operating statistics, especially those designed for accounting, are notoriously inaccurate. This has changed over the years as emphasis has shifted from accounting to performance monitoring. However, specific experiments designed to measure parameters for a specific model still tend to be the best way to proceed.

A Final Word

In a sense, the philosophy behind operational analysis is "measure and apply common sense". We have spent some time talking about the common sense aspects. Measuring systems, when you understand something of their internal structure, is almost always illuminating. It is very common for the measurement phase of a modelling study to show up enormous implementation inefficiencies which are corrected — the main benefit of the model in such cases is to point out the quantities that one should measure. Informed measurement is almost useful in itself.