Advanced Topics in Software Engineering: Operational Laws

Prof. Michele Loreti

Advanced Topics in Software Engineering
Corso di Laurea in Informatica (L31)
Scuola di Scienze e Tecnologie

Operational Laws

- Operational laws are simple equations which may be used as an abstract representation or model of the average behaviour of almost any system.

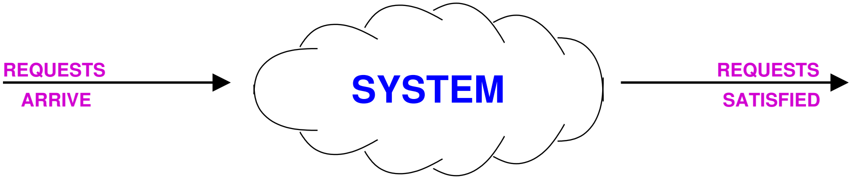
Operational Laws

- Operational laws are simple equations which may be used as an abstract representation or model of the average behaviour of almost any system.
- The laws are very general and make almost no assumptions about the behaviour of the random variables characterising the system.

Operational Laws

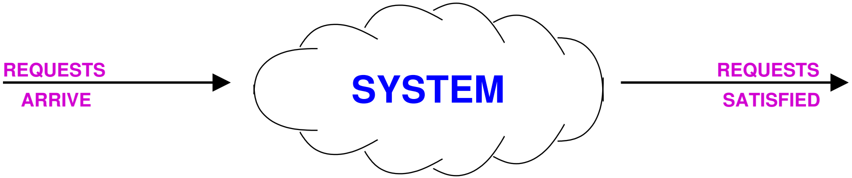
- Operational laws are simple equations which may be used as an abstract representation or model of the average behaviour of almost any system.
- The laws are very general and make almost no assumptions about the behaviour of the random variables characterising the system.
- Another advantage of the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

Observable variables



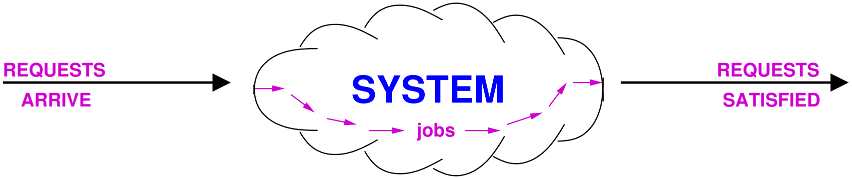
- Operational laws are based on observable variables — values which we could derive from watching a system over a finite period of time.

Observable variables



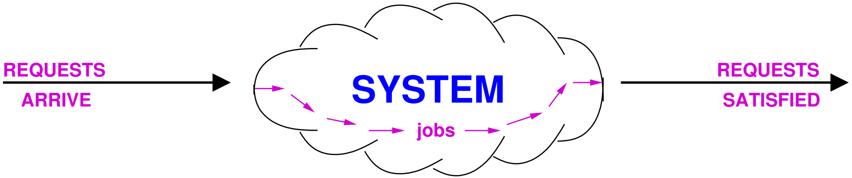
- Operational laws are based on observable variables — values which we could derive from watching a system over a finite period of time.
- We assume that the system receives requests from its environment.

Observable variables



- Operational laws are based on observable variables — 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.

Observable variables



- Operational laws are based on observable variables — 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.

Observations and measurements

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

T, the length of time we observe the system;

Observations and measurements

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;

Observations and measurements

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;

Observations and measurements

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 ($B \leq T$);

Observations and measurements

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.

Four important quantities

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

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

Four important quantities

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

$$\lambda = A/T, \text{ the arrival rate;}$$
$$X = C/T, \text{ the throughput or completion rate,}$$

Four important quantities

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

$$\lambda = A/T,\ \text{the arrival rate;}$$
$$X = C/T,\ \text{the throughput or completion rate,}$$
$$U = B/T,\ \text{the utilisation;}$$

Four important quantities

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

$$\lambda = A/T, \text{ the arrival rate;}$$
$$X = C/T, \text{ the throughput or completion rate,}$$
$$U = B/T, \text{ the utilisation;}$$
$$S = B/C, \text{ the mean service time per completed job.}$$

Job flow balance

- 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. $A = C$.

Job flow balance

- 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. $A = C$.
- This is a testable assumption because an analyst can always test whether the assumption holds.

Job flow balance

- 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. $A = C$.
- This 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, $\lambda = X$.

Little's Law

$$N = XW$$

Little's Law

The average number of jobs N in a system is equal to the product of the throughput of the system X and the average time W spent in that system by a job.

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) ($N = 4$).

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) ($N = 4$).

Little's law tells us that the average time spent at the disk by a request must be $4/40 = 0.1$ seconds.

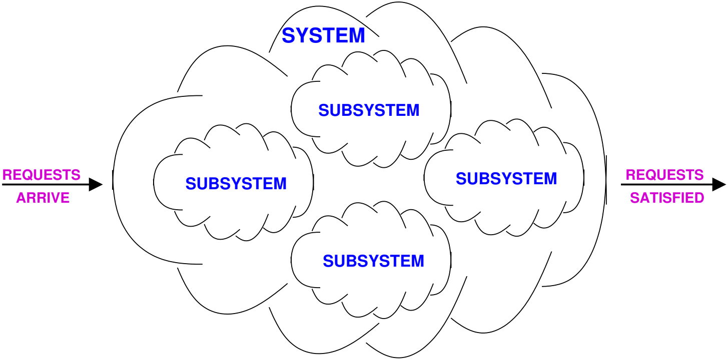
Example

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) ($N = 4$).

Little's law tells us that the average time spent at the disk by a request must be $4/40 = 0.1$ seconds.

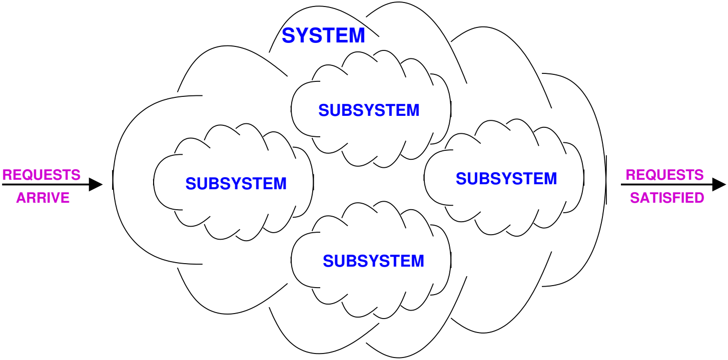
If we know that each request requires 0.0225 seconds of disk service we can then deduce that the average queueing time is 0.0775 seconds.

Subsystems within Systems



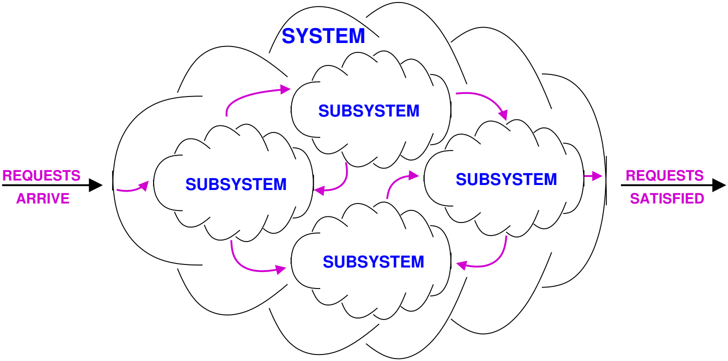
- A system may be regarded as being made up of a number of devices or resources.

Subsystems within Systems



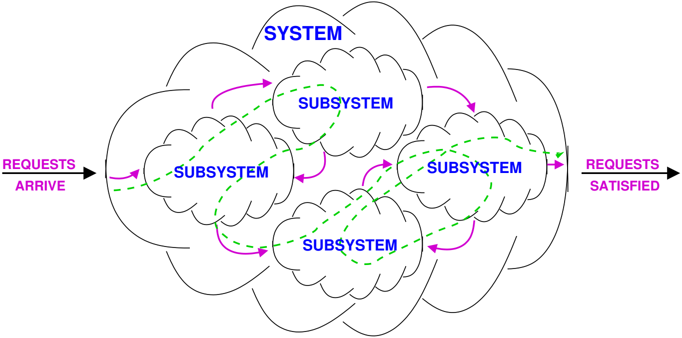
- A system may be regarded as being made up of a number of devices or resources.
- Each of these may be treated as a system in its own right from the perspective of operational laws.

Subsystems within Systems



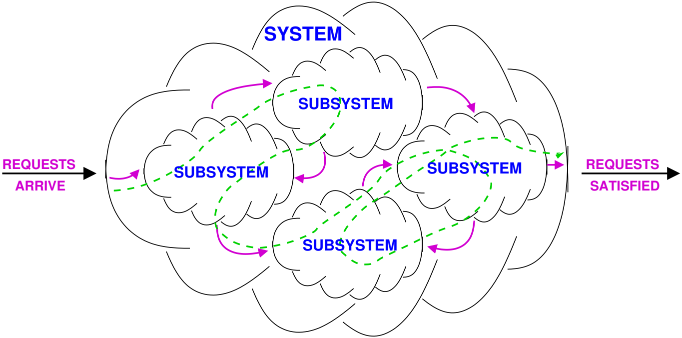
An external request generates a job within the system; this job may then circulate between the resources until all necessary processing has been done; as it arrives at each resource it is treated as a request, generating a job internal to that resource.

Subsystems within Systems



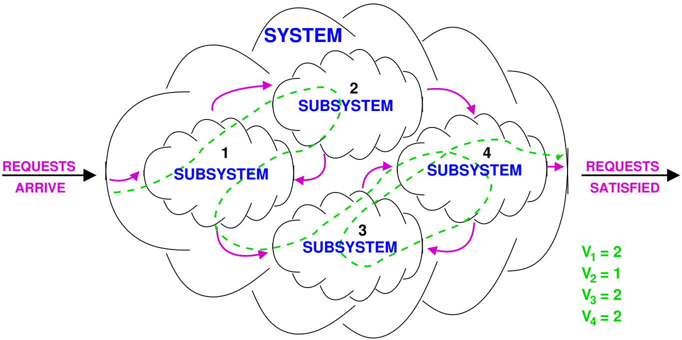
An external request generates a job within the system; this job may then circulate between the resources until all necessary processing has been done; as it arrives at each resource it is treated as a request, generating a job internal to that resource.

Visit count



In an observation interval we can count not only completions external to the system, but also the number of completions at each resource within the system.

Visit count



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 $V_i \equiv C_i/C$.

Visit count: example

For example, if, during an observation interval, we measure...

... 10 system completions

... and 150 completions at a specific disk

Visit count: example

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...

Visit count: example

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.

Forced Flow Law

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.

Forced Flow Law

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.

$$X_i = XV_i$$

Forced Flow Law

Forced Flow Law

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.

$$X_i = XV_i$$

Forced Flow Law

The throughput at the ith resource is equal to the product of the throughput of the system and the visit count at that resource.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.

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.

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.

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. $X = X_{lathe}/V_{lathe} = 8/4 = 2$.

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 $X_{press} = X \times V_{press} = 2 \times 2 = 4$.

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.

Utilisation Law

- If we know the amount of processing each job requires at a resource then we can calculate the utilisation of the resource.

Utilisation Law

- If we know the amount of processing 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.

Utilisation Law

- If we know the amount of processing 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.)

Utilisation Law

- If we know the amount of processing 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:

$$D_i = S_i V_i$$

Utilisation Law

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

Utilisation Law

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

$$U_i = X_i S_i = X D_i$$

Utilisation Law

Utilisation Law

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

$$U_i = X_i S_i = X D_i$$

Utilisation Law

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

Utilisation Example

- Consider again the disk that is serving 40 requests/second, each of which requires 0.0225 seconds of disk service.

Utilisation 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 $40 \times 0.0225 = 90\%$.

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.

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.

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 $N_i = X_i W_i$, where N_i is the mean number of jobs at the resource and W_i is the average response time of the resource.

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 $N_i = X_i W_i$, 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 $X_i = XV_i$. Thus we can deduce that

$$N_i/X = V_i W_i.$$

General Residence Time Law

The total number of 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. From Little's Law that $W = N/X$ and so:

General Residence Time Law

The total number of 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. From Little's Law that $W = N/X$ and so:

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

General Residence Time Law

General Residence Time Law

The total number of 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. From Little's Law that $W = N/X$ and so:

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

General Residence 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.

General Residence Time Law: Example

- A program running on a compute server requires 126 bursts of CPU time and makes 75 I/O requests to disk A and 50 I/O requests to disk B.

General Residence Time Law: Example

- A program running on a compute 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).

General Residence Time Law: Example

- A program running on a compute 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.

General Residence Time Law: Example (continued)

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 $+1$):

$$W_{diskA} = \frac{N_{diskA}}{X_{diskA}}$$

General Residence Time Law: Example (continued)

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):

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

General Residence Time Law: Example (continued)

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 $+1$):

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

General Residence Time Law: Example (continued)

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):

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

$$W_{diskB} = \frac{N_{diskB}}{X_{diskB}}$$

General Residence Time Law: Example (continued)

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 $+1$):

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

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

General Residence Time Law: Example (continued)

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):

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

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

General Residence Time Law: Example (concluded)

Then

$$W = W_{CPU} V_{CPU} + W_{diskA} V_{diskA} + W_{diskB} V_{diskB}$$

General Residence Time Law: Example (concluded)

Then

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

General Residence Time Law: Example (concluded)

Then

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

General Residence Time Law: Example (concluded)

Then

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

Interactive Response Time Law

- Back when most processing was done on shared mainframes, think time, Z, was quite literally the length of time that a programmer spent thinking before submitting another job.

Interactive Response Time Law

- Back when most processing was done on shared mainframes, think time, Z, was quite literally the length of time that a programmer spent thinking before submitting another job.
- More generally in interactive systems, 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.

Interactive Response Time Law

- Back when most processing was done on shared mainframes, think time, Z, was quite literally the length of time that a programmer spent thinking before submitting another job.
- More generally in interactive systems, 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.
- 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.

Example

- 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.

Example

- 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.

Think time, residence time, response time

- The think time represents the time between processing being completed and the job becoming available as a request again.

Think time, residence time, response time

- 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.

Interactive Response Time Law

The interactive response time law reflects this: it calculates the response time, R as follows:

Interactive Response Time Law

The interactive response time law reflects this: it calculates the response time, R as follows:

$$R = N/X - Z$$

Interactive Response Time Law

Interactive Response Time Law

The interactive response time law reflects this: it calculates the response time, R as follows:

$$R = N/X - Z$$

Interactive Response Time Law

The response time in an interactive system is the residence time minus the think time.

Interactive Response Time Law

The interactive response time law reflects this: it calculates the response time, R as follows:

$$R = N/X - Z$$

Interactive Response Time Law

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.

Interactive Response Time Law: Example

- Suppose that the library catalogue system has 64 interactive users connected via Browsers, that the average think time is 30 seconds, and that system throughput is 2 interactions/second.

Interactive Response Time Law: Example

- Suppose that the library catalogue system has 64 interactive users connected via 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.

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}.

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.

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.

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.

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,

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

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,

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

- 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. $W \geq D$

Throughput, utilisation and overall performance

- 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.

Throughput, utilisation and overall performance

- 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, $U_b = XD_{max}$.

Throughput, utilisation and overall performance

- 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 $U_b \leq 1$

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

Throughput, utilisation and overall performance

- 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.

Obtaining a tighter bound

- 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).

Obtaining a tighter bound

- 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:

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

Obtaining a tighter bound

- 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 $W \geq ND_{max}$.

Asymptotic bound

Thus the asymptotic bound for residence time or response time is:

$$\underbrace{W \geq \max\{D, ND_{max}\}}_{\textbf{Residence Time Bound}}$$

$$\underbrace{R \geq \max\{D, ND_{max} - Z\}}_{\textbf{Response Time Bound}}$$

Bound on a lightly loaded system

- 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).

Bound on a lightly loaded system

- 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:

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

Bound on a lightly loaded system

- 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 $X \leq N/D$.

Bound on a lightly loaded system

- 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 $X \leq N/D$.

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

Throughput Bound (lightly loaded system)

- Notice that the bottleneck depends on both resource parameters (X_i or S_i) and the workload parameters (V_i).

- 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.

Assumptions

- As mentioned in the introduction, the operational laws do not rely on many assumptions.
- 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.

To be continued. . .