Lecture 33 — Applying Queueing Theory

Jeff Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

December 2, 2017

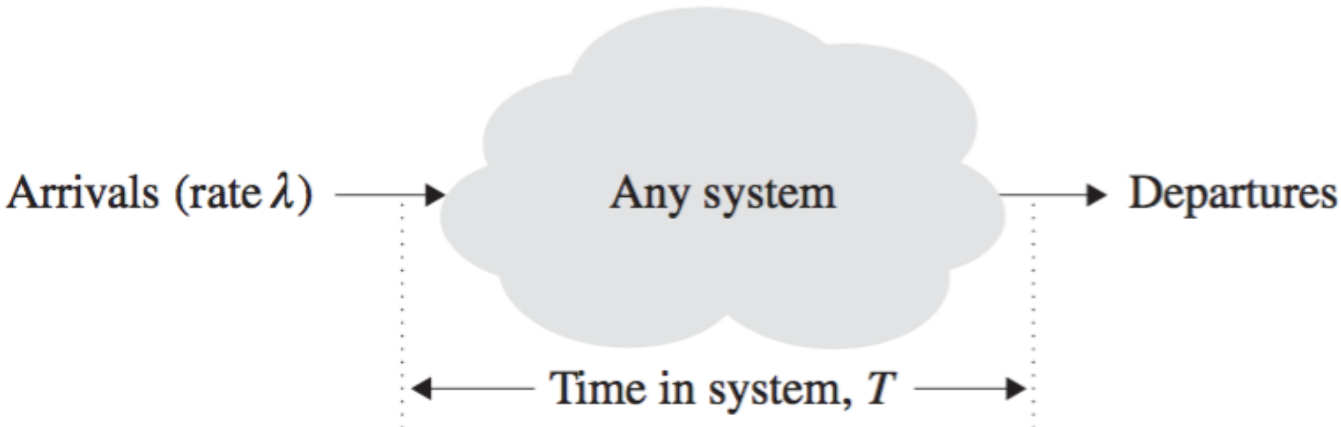
Little's Law

Little's Law says the average number of jobs in the system equals the product the average arrival rate into the system and the average time in the system.

Let's start with an open system. The law, written more formally:

$$E[N] = \lambda E[T]$$

Little's Law


Arrivals (rate λ)
Any system
Departures
Time in system, T

Observations on Little's Law

Note that we don't need to know anything about the arrival process the service time distribution, network topology, etc.

It seems intuitive that this is the case (or it should).

Restaurant Analogy

Imagine a fast food restaurant: they make money by quick turnaround time.

They get people out of the place quickly (low $E[T]$) and accordingly they don't require a lot of seating (low $E[N]$).

A sit down restaurant is the opposite though; people leave slowly (high $E[T]$) and therefore the restaurant needs lots of seating (more $E[N]$).

FCFS Version

If you prefer to think of this in a single FCFS queue version, imagine a customer arrives and sees $E[N]$ jobs ahead of her in the queue.

The expected time for each customer to complete is $1/\lambda$, because the average rate of completions is λ.

So we can approximate $E[T]$ as being roughly $\frac{1}{\lambda}E[N]$.

Little's Law for Closed Systems

Remember that for closed systems, we have a rule that says there is N jobs in process at any given time (the multiprocessing level of the system).

If the system is ergodic, then $N = X \cdot E[T]$.

This assumes that there is zero think time.

Users Ruin Everything

If we do have to deal with the vagaries of users and this time, then we care more about the response time $E[R]$.

So for a terminal-driven system, the expected response time is $E[R] = \frac{N}{X} - E[Z]$.

Probabilistic processes are described according to their models, which will probably one of the three:

1. Deterministic (D)
2. Markov (M)
3. General (G)

Markov Processes

We’re going to focus on Markov processes.

It means that the number of arrivals follow the Poisson distribution.

The inter-arrival times follow the exponential distribution.

Service times follow the exponential distribution too.

Kendall Notation

Those letters we saw are part of Kendall notation.

It has six symbols, written in a specific order, separated by slashes.

The order is $\alpha/\sigma/m/\beta/N/Q$.

Kendall Notation

Symbol	**Meaning**
α	The type of distribution (Markov, General, Deterministic)
σ	The type of probability distribution for service time
m	Number of servers
β	Buffer size
N	Allowed population size (finite or infinite)
Q	Queueing policy

Why is abbreviation such a long word?

We often leave off the last three, assuming that there is an infinite buffer, infinite population, and a FIFO queueing policy.

If that is the case, then we have only three values.

Those three then produce the “M/M/1” and “M/M/k” symbols.

“M/M/1” means a Markov arrival process, exponential queueing system, and a single server.

When there are k servers, then, of course the 1 is replaced with the k.

Utilization

We should also think about utilization, denoted ρ.

It is a fraction between 0 and 1 and it is simply the amount of time that the server is busy.

We talked about this earlier in an informal way, but now we can actually calculate it!

$\rho = \lambda \times s$.

M/M/1 Formulae

For M/M/1 systems:

The completion time average T_q is $\frac{s}{(1-\rho)}$

The average length of the queue W is $\frac{\rho^2}{1-\rho}$.

Queuing Theory Example

We have a server that completes a request, on average, in 10 ms.

The time to complete a request is exponentially distributed.

Over a period of 30 minutes, 117 000 jobs arrive.

So this is a M/M/1 situation.

How long did it take to complete the average request?

What is the average length of the queue?

Queueing Theory Example

The service time s is given as $0.01s$, the arrival rate is 65 requests per second.

So we can calculate $\rho = 0.01 \times 65 = 0.65$.

So we have what we need to plug and chug using the formulae from above.

The time to complete the average request is 28.6 ms.

The average length of the queue is 1.21.

Queueing Theory Example

What about the number of jobs in the system?

The value Q gives the average number of jobs, including the waiting jobs and the ones being served.

The probability that there are exactly x jobs in the system at any time is given by the formula: $(1 - \rho)\rho^x$.

Queueing Theory Example

The probability that the number of jobs is less than or equal to n is then given by:

$$\sum_{i=0}^{n}(1-\rho)\rho^i.$$

For more than n at a time, from $n+1$ to infinity... Or...

$$1-\sum_{i=0}^{n}(1-\rho)\rho^i.$$

M/M/k

Now let us take it to multiple servers.

We will say jobs arrive at a single queue and then when a server is ready it will take the first job from the front of the queue.

The servers are identical and jobs can be served by any server.

So far, so simple. Sadly, the math just got harder.

The server utilization for the server farm is now $\rho = \lambda s/N$.

To make our calculations a little easier, we want an intermediate value K:

$$K = \frac{\sum_{i=0}^{N-1} \frac{(\lambda s)^i}{i!}}{\sum_{i=0}^{N} \frac{(\lambda s)^i}{i!}}$$

K is always less than 1. It has no intrinsic meaning.

What is the probability that all servers are busy?

We represent this as C, the probability a new job will have to wait in the queue.

$$C = \frac{1 - K}{1 - \frac{\lambda s K}{N}}$$

The M/M/k formulae:

$$T_q = \frac{Cs}{k(1-\rho)} + s \qquad W = C\frac{\rho}{1-\rho}$$

Printer Example

Suppose we have a printer that can complete an average print job in 2 min.

Every 2.5 minutes, a user submits a job to the printer.

How long does it take to get the print job on average?

We're starting with a single printer, so the system is M/M/1.

Service time s is 2 minutes; the arrival rate λ is $1/2.5 = 0.4$.

So $\rho = \lambda \times s = 0.4 \times 2 = 0.8$.

So $T_q = s/(1 - \rho) = 2/(1 - 0.8) = 10$.

Ten minutes to get the print job. Ouch.

Executive Decision

Here we have an opportunity to use the predictive power of queueing theory.

Management is convinced that ten minute waits for print jobs is unreasonable, so we have been asked to decide what to do.

Should we buy a second printer of the same speed, or should we sell the old one and buy a printer that is double the speed?

New Printer

The faster printer calculation is easy enough.

Now $s = 1.0$ and λ remains 0.4, making $\rho = 0.4$.

So rerunning the calculation: $T_q = s/(1 - \rho) = 1/(1 - 0.4) = 1.67$.

1:40 is a lot less time than 10:00!

Two Printer Solution

The two printer solution is more complicated. So let us calculate K as the intermediate value.

$$K = \frac{\sum_{i=0}^{N-1} \frac{(\lambda s)^i}{i!}}{\sum_{i=0}^{N} \frac{(\lambda s)^i}{i!}} = \frac{\frac{(\lambda s)^0}{0!} + \frac{(\lambda s)^1}{1!}}{\frac{(\lambda s)^0}{0!} + \frac{(\lambda s)^1}{1!} + \frac{(\lambda s)^2}{2!}} = 0.849057$$

Calculate!

Now we can calculate C as 0.22857 and T_q as 2.57 minutes.

Two observations jump out at us:

(1) we doubled the number of printers, but now jobs are completed almost four times faster; and

(2) the single fast printer is better, if utilization is low.

Conditions

That is an important condition: if utilization is low.

At some point will the two printers be a better choice than the single fast one?

What if both printers are used to the max (100% load)...?

The basic process is:

1. Convert to common time units.
2. Calculate the visitation ratios V_i.
3. Calculate the device utilization ρ_i.
4. Calculate the CPU service time.
5. Calculate the device time.
6. Find the bottleneck device.
7. Calculate the maximum transaction rate.
8. Calculate the average transaction time.

Let us execute this process on a web server system that serves 9 000 pages per hour. Here are the known values:

Device	**Data/Hour**	λ	S	V	ρ	$V \times S$
Webpages	9 000					
CPU					42%	
Disk 1	108 000		11ms			
Disk 2	72 000		16ms			
Network	18 000		23ms			

Step one is to convert to common time units; in this case, seconds.

Let's also look at the λ values - reported counts divided by seconds in the reporting period.

Device	**Data/Hour**	λ	S	V	ρ	$V \times S$
Webpages	9 000	2.5				
CPU					42%	
Disk 1	108 000	30	0.011s			
Disk 2	72 000	20	0.016s			
Network	18 000	5	0.023s			

The visitation ratio is the number of times a device is used in each transaction; divide use by number of transactions to get V_i (you could also log this sort of thing).

The visitation ratio of the CPU is the sum of all other visitation ratios.

Device	**Data/Hour**	λ	S	V	ρ	$V \times S$
Webpages	9 000	2.5		1		
CPU	207 000	57.5		23	42%	
Disk 1	108 000	30	0.011s	12		
Disk 2	72 000	20	0.016s	8		
Network	18 000	5	0.023s	2		

Next, calculate device utilization: $\rho = \lambda \times s$. That is, arrival rate times service time.

Device	**Data/Hour**	λ	S	V	ρ	$V \times S$
Webpages	9 000	2.5		1		
CPU	207 000	57.5		23	42%	
Disk 1	108 000	30	0.011s	12	0.33	
Disk 2	72 000	20	0.016s	8	0.32	
Network	18 000	5	0.023s	2	0.115	

We can also get the service time of the CPU by rearrangement of the utilization formula to $s = \rho/\lambda$.

Device	**Data/Hour**	λ	S	V	ρ	$V \times S$
Webpages	9 000	2.5		1		
CPU	207 000	57.5	0.0073s	23	0.42	
Disk 1	108 000	30	0.011s	12	0.33	
Disk 2	72 000	20	0.016s	8	0.32	
Network	18 000	5	0.023s	2	0.115	

And the device time is the final thing we can fill in for this table: $V_i \times S_i$ (just like the column header says!).

Device	**Data/Hour**	λ	S	V	ρ	$V \times S$
Webpages	9 000	2.5		1		
CPU	207 000	57.5	0.0073s	23	0.42	0.168
Disk 1	108 000	30	0.011s	12	0.33	0.132
Disk 2	72 000	20	0.016s	8	0.32	0.128
Network	18 000	5	0.023s	2	0.115	0.046

Is It Useful?

Did we need to complete the whole table? Probably not.

In a practical sense what we cared about the most was the ρ column – utilization.

The bottleneck device, i.e., the one that limits our maximum throughput, is the one that is the busiest.

Thus, the one with the largest utilization.

This application appears to be CPU bound; it has the highest utilization at 42%, well ahead of disk 1 and disk 2.

Predicting the Maxes

Having identified the bottleneck device as the CPU, we can make a prediction about the maximum rate of transactions (web page requests) we can serve.

$\frac{1}{S_i V_i}$ or in this example, 5.95.

This is also called saturation.

If λ exceeds this saturation point, we will not be able to keep up with incoming requests.

With this table we can also calculate the average transaction time: it is the sum of the $S_i V_i$ columns.

In this example, it is 0.474 seconds.

It Gets Worse!

The typical assumption is that we know the service times for each device.

Unfortunately this is not true; usually performance monitoring gives us the average size of a device queue.

So we had better apply queuing theory here.

The average size of a device's queue is W, and for a queue with characteristics M/M/1 then $W = \frac{\rho^2}{1-\rho}$.

Combining the known W with the average arrival rate λ, we can work out the service time.

$$W = \frac{(\lambda s)^2}{1 - \lambda s}$$

$$s = \frac{-w \pm \sqrt{w^2 + 4w}}{2\lambda}$$