Multiserver Capacity Equation

Done Right ☺

Dr. Neil J. Gunther

Performance Dynamics

August 10, 2009

Alistair's Capacity Equation

$$Delay\ (\text{sec}) = \frac{Traffic\ (\text{reqs/sec})}{Capacity\ (\#\text{machines})} \tag{1}$$

This equation says 2 things:

1. Delay is directly proportional to the traffic (i.e., throughput)
2. Delay is inversely proportional to the system capacity

Example

Think of a grocery store. More customer traffic, longer lines means longer delays. If we add more capacity/checkout lanes, the delay should be shorter.

Seems to make intuitive sense, but is it correct?

Dimensional Analysis

- Delay clearly has the dimensional units of *time*: [time]
- Traffic rate has dimensional units of *inverse time*: [1/time]
- Pure quantities or numbers have no dimensional units: [1]

Example

A rate represents some number of things completed per unit time. Speed or velocity is a rate measured in miles completed per hour (in the USA). Number of miles is a dimensionless engineering quantity:

$$[speed] = \frac{miles}{hour} \equiv \left[\frac{1}{time}\right]$$

All rates have the dimensional units of inverse time.

Throughput is a rate

Problem

In dimensional form, Alistair's eqn.(1) becomes:

$$[Delay] = [Throughput] \tag{2}$$

where:

- Delay has the dimensional units of time: [*time*]
- Traffic throughput has dimensions of inverse time: $[1/time]$
- Machine capacity is dimensionless: [1]

But eqn.(2) fails dimensional analysis because:

$$[time] = \left[\frac{1}{time}\right] \tag{3}$$

It's like saying: $seconds = 1/seconds$. Can't haz!

What to Do?

We could make things dimensionally consistent by inverting the right-hand side of eqn.(3). In other words:

$$[time] = \left[\frac{1}{time}\right]^{-1} = [rate]^{-1} = \frac{1}{rate} \tag{4}$$

Since *two inverses make an upright* ☺ :

$$Delay = \frac{k}{Throughput} \tag{5}$$

which says the Delay is inversely proportional to Throughput. In symbols:

$$R = \frac{k}{X} \tag{6}$$

where k is a number (constant of proportionality)

Correct But Very Sophisticated Formula

I'm going to "simplify" the discussion with these symbols:

Definitions

- R: average delay or response time [*time*]
- λ: arrival rate of requests per unit time [$1/time$]
- m: number of **servers** or machines (dimensionless)
- S: the service time [*time*]
- ρ: server utilization or traffic intensity $m\rho = \lambda S$ (dimensionless) [1]
  Note: $0 < \rho < 1$ (can't be more than 100% utilized)

$$R = \frac{S}{1 - (\lambda S/m)^m} = \frac{S}{1 - \rho^m} \tag{7}$$

Dimensionally correct: [*time*] = [*time*]

Grocery Store with $m = 1$ Lane

$$R = \frac{S}{1 - \lambda S} \tag{8}$$

Note the arrival rate (λ) or throughput is in the denominator.

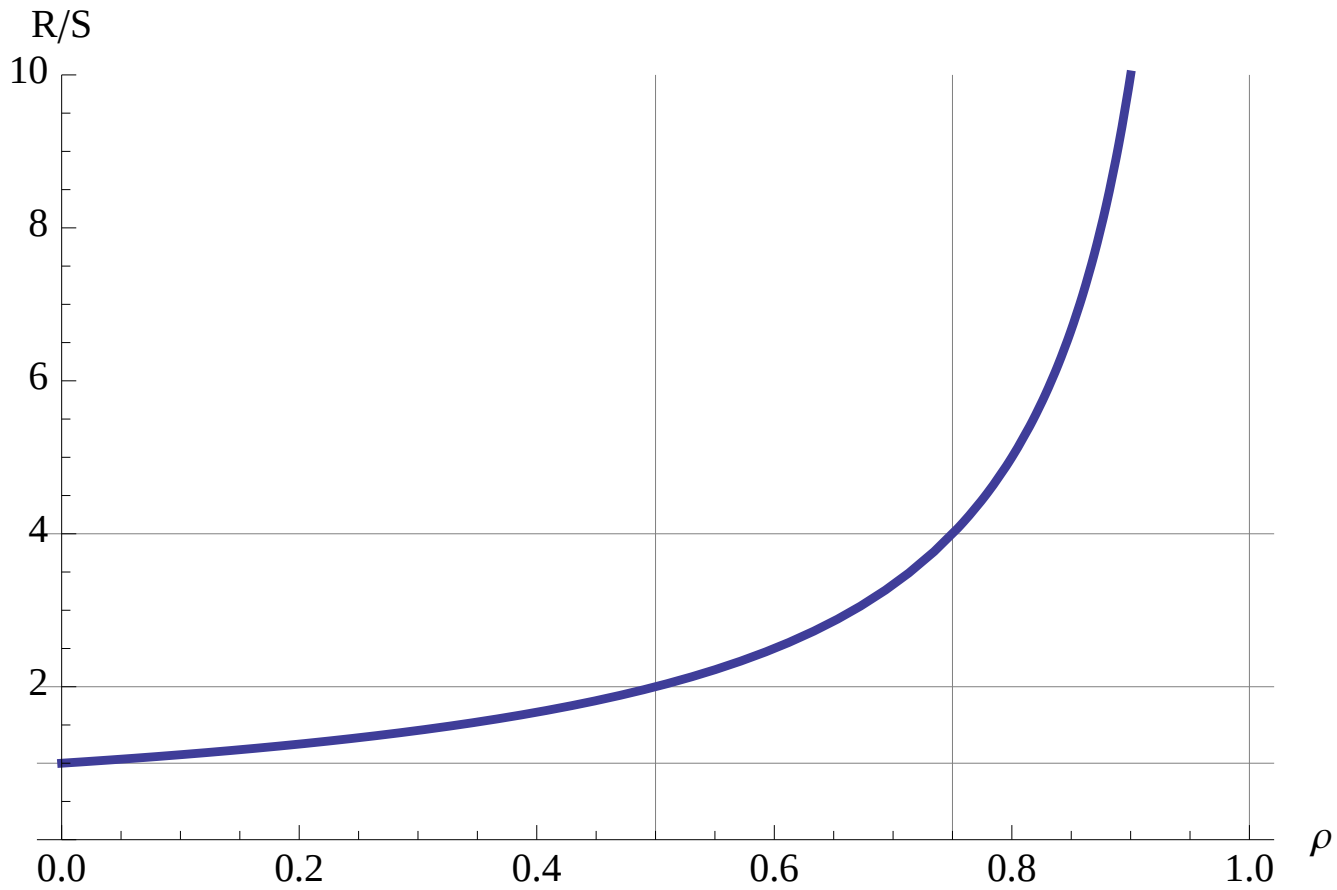
Example

Slack period: $\lambda \simeq 0$. Amount of traffic in the store is small. $R = S$ in eqn.(8). Time to get through checkout is just the time S for the cashier to ring up your groceries.

Peak period: $\lambda \simeq 1/S$. Then, $\lambda S \simeq 1$, the difference $(1 - \lambda S) \to 0$ in denominator of eqn.(8) so, $R \to \infty$. In heavy traffic, your time to get through the checkout grows unbounded, i.e., will take a hell of a long time!

Equations are good, plots are better ...

What Happens Between Light and Heavy Traffic?



The relationship between R and λ ($\rho = \lambda S$) is nonlinear (because of the dimensional inverse) and therefore looks like a curve—a very special kind of curve.

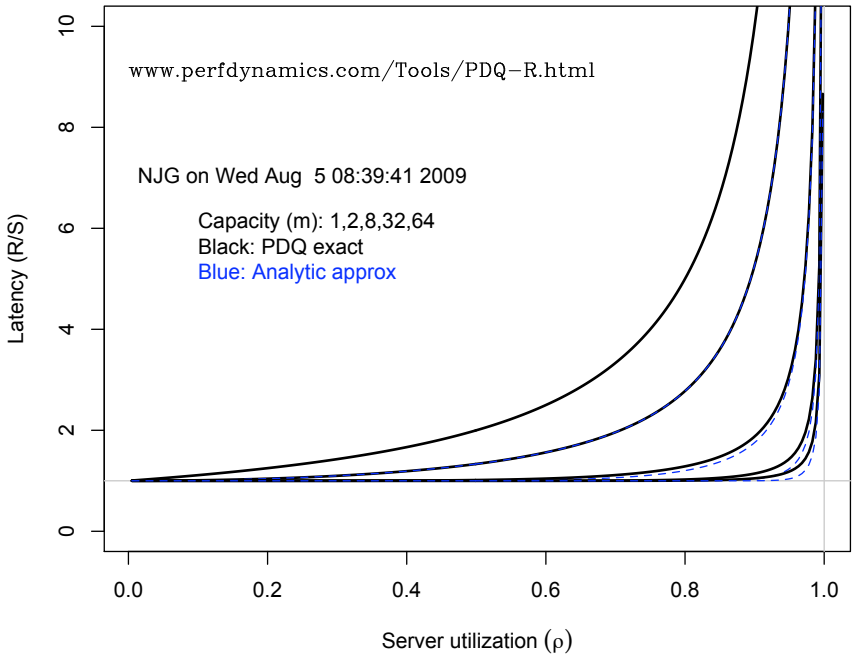
What Happens with More Server Capacity?

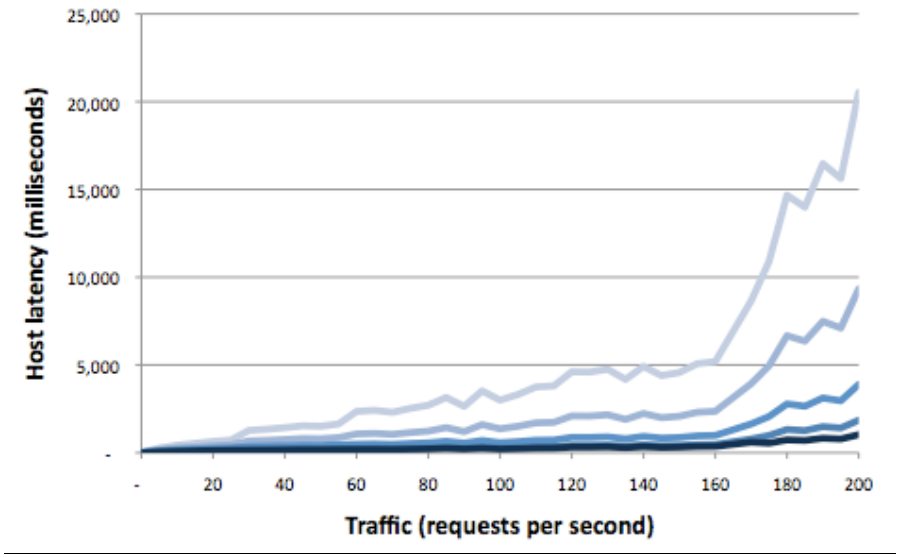
Use PDQ performance tool to get exact solutions.

```
library(pdq)

# PDQ globals
stime<-0.499
servers<-c(1,2,8,32,64)
node<-"qnode"
work<-"qwork"

for (m in servers) {
        arrivrate<-0
        xc<-0
        yc<-0

        for (i in 1:200) {
                Init("")

                arrivrate<-arrivrate + 0.01
                aggArrivals<-m * arrivrate # remember Erlang!
                CreateOpen(work, aggArrivals)
                CreateMultiNode(m, node, CEN, FCFS)
                SetDemand(node, work, stime)

                Solve(CANON)
                xc[i]<-as.double(aggArrivals*stime/m) # rho
                yc[i]<-GetResidenceTime(node,work,TRANS)/stime # R/S
        }

        if (m == 1) {
                # draw plot frame and first curve
```

PDQ (Pretty Damn Quick) Output

By increasing server capacity (m), the response time remains close to service time S (horizontal line at $Latency = 1$) at higher server utilizations or traffic rates.





Similar to latency profile seen in measured performance data.

My Proposed Equation

On July 7, 2009 I came up with this exact and correct multiserver equation:

Definition

$$R = \frac{N}{m\,\lambda} \tag{9}$$

This is very close to eqn.(1) in form and spirit.

$k \rightarrow N$ is the previous "constant" of proportionality. It's really a function of the request rate: $N = N(\lambda)$, ... I lied. ☺

In words:

$$\text{Delay} = \frac{\text{Total requests}}{(\#\text{ machines}) \times (\text{machine Throughput})} \tag{10}$$

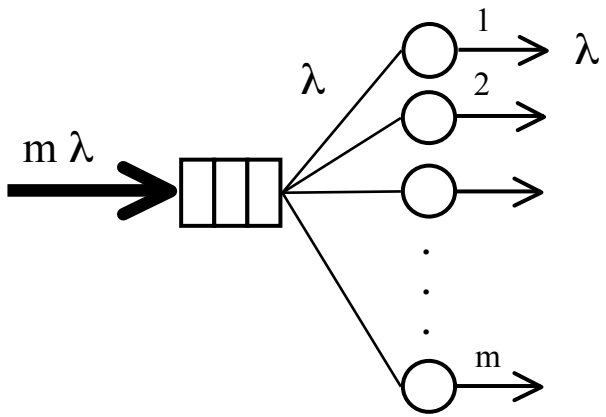
Doesn't blind the audience with too much science. ☺

Equations are good, diagrams are better ... (next slide) ...

Diagram for Neil's Equation

Definitions

1. Circles represent machines servicing requests
2. Squares represent waiting requests



$$R = \frac{N}{m\,\lambda}$$

- Aggregate arrivals or total throughput: $m\lambda$
- Total number of requests already in the system: N
- Throughput of each machine λ is in *denominator*
- Satisfies: delay $= k/$throughput, per dimensional eqn.(5)
- Nonlinearity (slide 10) is hidden in the $N(\lambda)$ function

Coordinates

- www.perfdynamics.com
- perfdynamics.blogspot.com
- twitter.com/DrQz
- njgunther@perfdynamics.com