Advanced Computer Architecture:
A Google Search Engine

Jeremy Bradley

Room 372. Office hour – Thursdays at 3pm. Email: jb@doc.ic.ac.uk

Course notes: http://www.doc.ic.ac.uk/∼jb/

Department of Computing, Imperial College London

Produced with prosper and $\LaTeX$

Response Time Summary

- Google processes between 1000 and 2000 requests per second on average
- Google has a response time requirement that it should process all requests within 0.5s
- Start by treating system as black box with just these constraints

Little's Law

- Used to estimate response time of a processing system:

$$\mathbb{E}(N) = \lambda \times \mathbb{E}(R)$$

- $\mathbb{E}(N)$ = mean number of jobs in system
- $\mathbb{E}(R)$ = mean response time for a job (time elapsed from the instant job enters system until its completion)
- λ = arrival rate of jobs/requests
- Only applies for system in steady-state
- No creating/destroying of jobs

A Basic Processing System

- Job arrival rate = λ jobs per second
- Size of buffer = m cells
- Probability that buffer has n jobs in at steady-state = π_n

$\Rightarrow$ mean no. of jobs = $\mathbb{E}(N) = \sum_{n=0}^{m} n\pi_n$

$\Rightarrow$ mean response time = $\mathbb{E}(R) = \frac{1}{\lambda}\sum_{n=0}^{m} n\pi_n$

Google Specifics (I)

- We know that in December 2000, λ was roughly 1000 requests per second
- Assume 10,000 mile worst case distance from a cohosting site
- Using speed of light $\sim$ 300,000 kms^{-1}

$\Rightarrow$ at least 0.1 second network latency per request

Google Specifics (II)

- We know that Google set itself the target that mean response time should be < 0.5s including network latency: i.e. $\mathbb{E}(R) < 0.4$
- By Little's Law: $\mathbb{E}(N) = \lambda \times \mathbb{E}(R)$

$\Rightarrow$ $\mathbb{E}(R) = \mathbb{E}(N)/\lambda < 0.4$

  - Dec. 2000: $\lambda = 1000$, $\mathbb{E}(N) < 0.4 \times 1000 = 400$
  - Feb. 2004: $\lambda = 2000$, $\mathbb{E}(N) < 0.4 \times 2000 = 800$

- i.e. In December 2000, average number of queued requests at any moment should have been < 400

Google as a single server (I)

- Take a Google site to be a single server in an M/M/1 queue
- $\mathbb{E}(N)$ for an M/M/1 queue is:

$$\frac{\rho}{1-\rho}$$

- where $\rho =$ arrival rate $\times$ av. time in service
- μ is the service rate of the whole site (in this case)

$\Rightarrow$ $\rho = \lambda/\mu$ is called server utilisation

Google as a single server (II)

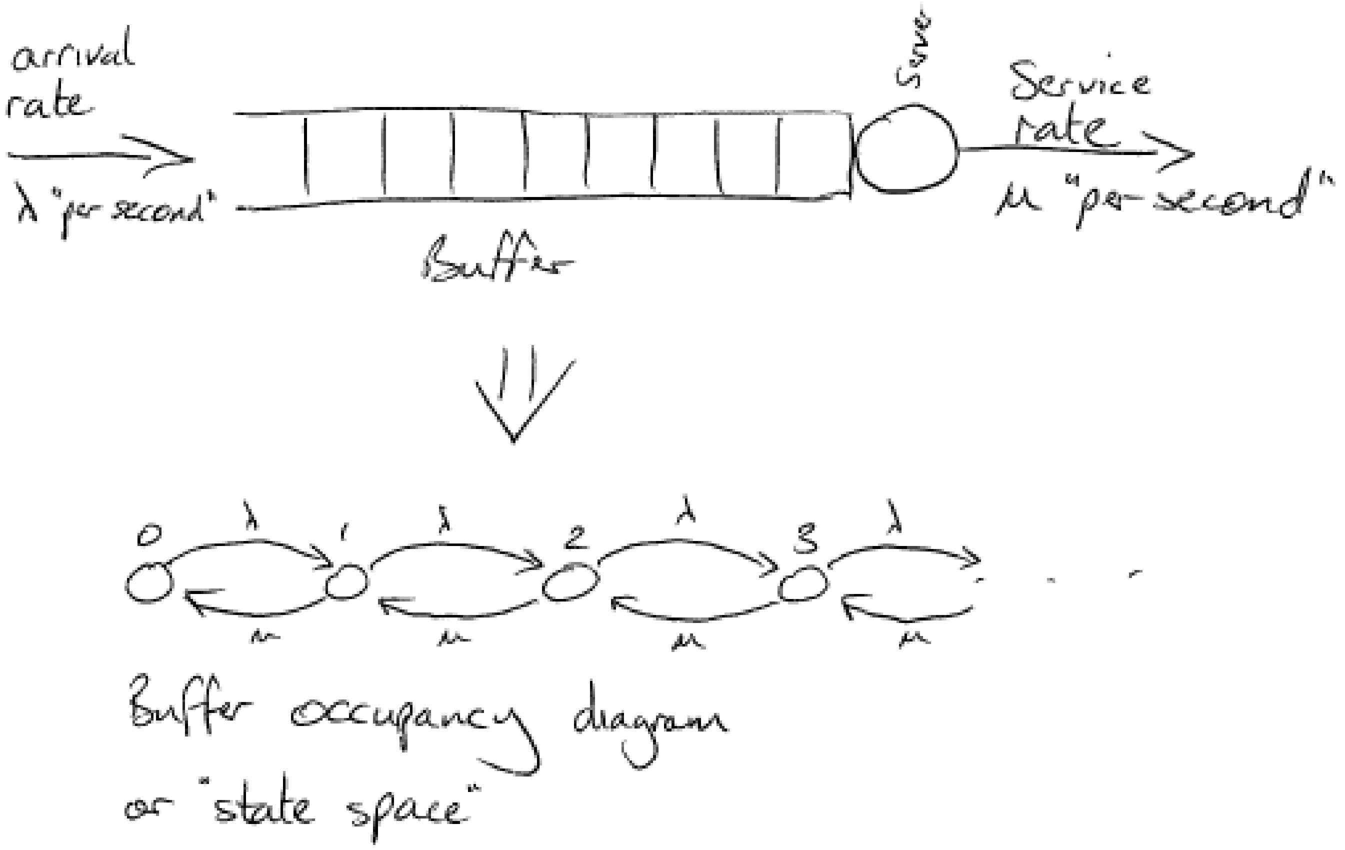
- We need $\mathbb{E}(N) < L$ for some value L, so:

$$\mu > \frac{L+1}{L}\lambda$$

  - i.e. if $L = 400$, $\lambda = 1000$ then $\mu > 1002.5$ per second
  - i.e. if $L = 800$, $\lambda = 2000$ then $\mu > 2002.5$ per second
- i.e. an overall Google cluster had to be able to service requests at greater than 1002.5 per second assuming an incoming load of 1000 per second, if the number of waiting jobs was to be kept below 400 and the system response time was to be kept below 0.4s

An M/M/1 Queue Diagram

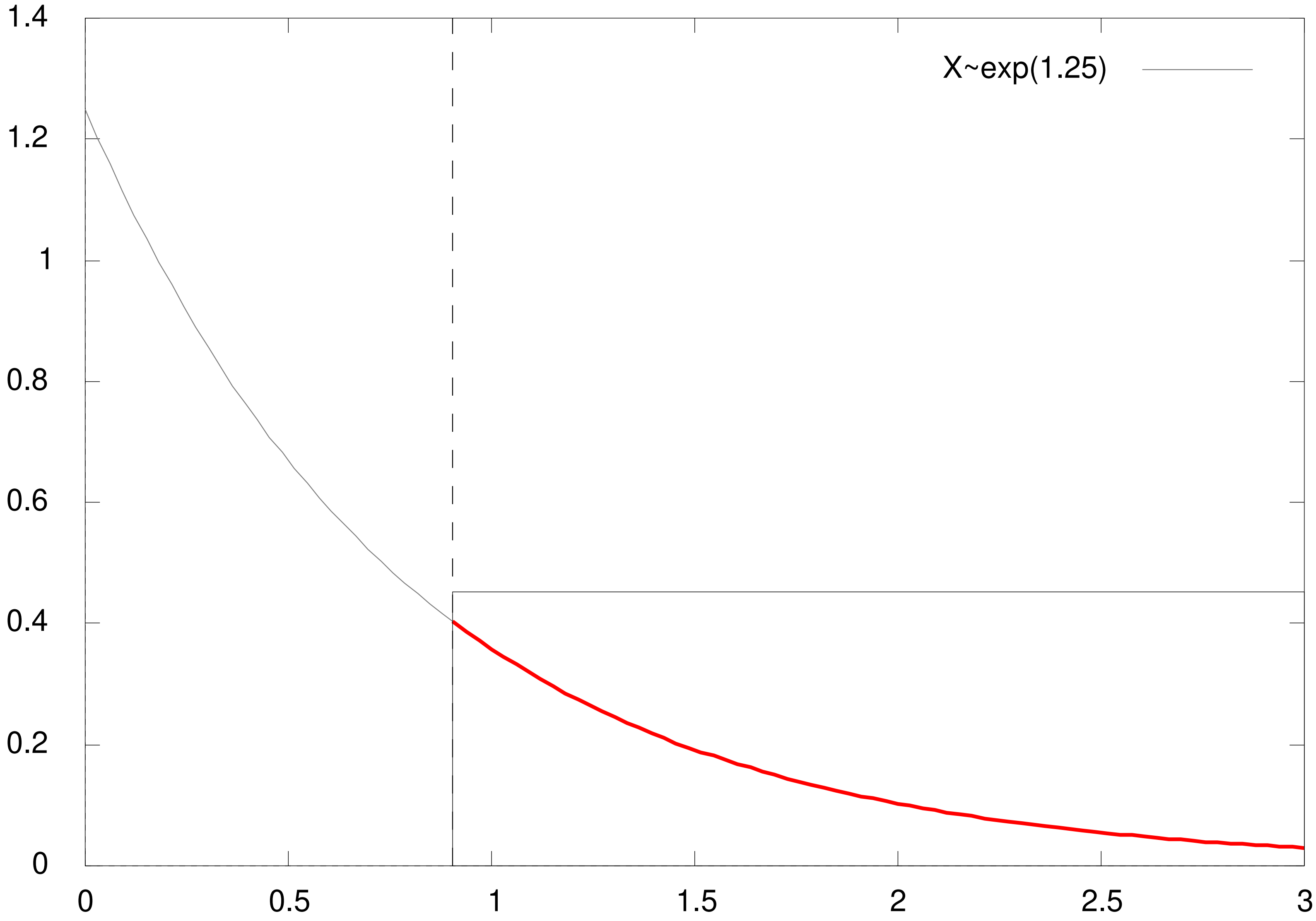
- Arrival rate: λ, Service rate: μ (see Figs. 8.2/8.1 [Tri])
- Markovian arrivals/services, infinite buffer, one server



Why Markov?

- Why is a Markovian arrival a good model?
  - **Good reason**: random interleaved independent streams of requests tend to be Markovian
  - **OK reason**: if you know nothing about the distribution, then this is the least biased distribution to use
  - **Bad reason**: the maths is easier
- Markovian (or exponential) distribution is memoryless. i.e. it can be interrupted anywhere and the remainder is also Markovian

Markov property



Small bit of queueing theory

- Going to show that mean queue length, $\mathbb{E}(N)$, on M/M/1 queue at steady-state is $\rho/(1-\rho)$
- As $\mathbb{E}(N) = \sum_{k=0}^{\infty} k\pi_k$, we need to find π_k:
  - Derive steady-state equations from time-varying equations
  - Solve steady-state equations to get π_k
  - Calculate M/M/1 mean queue length, $\mathbb{E}(N)$
- (In what follows, remember $\rho = \lambda/\mu$)

Small bit of queueing theory

- Write down time-varying equations for M/M/1 queue:
  - At time t, in state $k = 0$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\pi_0(t) = -\lambda\pi_0(t) + \mu\pi_1(t)$$

  - At time, t, in state $k \geq 1$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\pi_k(t) = -(\lambda+\mu)\pi_k(t)+\lambda\pi_{k-1}(t)+\mu\pi_{k+1}(t)$$

Steady-state for M/M/1

- At steady-state, $\pi_k(t)$ are constant (i.e. π_k) and $\frac{\mathrm{d}}{\mathrm{d}t}\pi_k(t) = 0$ for all k

$\Rightarrow$ Balance equations:

  - $-\lambda\pi_0 + \mu\pi_1 = 0$
  - $-(\lambda + \mu)\pi_k + \lambda\pi_{k-1} + \mu\pi_{k+1} = 0 \quad : k \geq 1$

- Rearrange balance equations to give:
  - $\pi_1 = \frac{\lambda}{\mu}\pi_0 = \rho\pi_0$
  - $\pi_{k+1} = \frac{\lambda+\mu}{\mu}\pi_k - \frac{\lambda}{\mu}\pi_{k-1} \quad : k \geq 1$

- Solution: $\pi_k = \rho^k\pi_0$ (proof by induction)

Normalising to find π_0

- As these π_k are probabilities which sum to 1:

$$\sum_{k=0}^{\infty} \pi_k = 1$$

- i.e. $\sum_{k=0}^{\infty} \pi_k = \sum_{k=0}^{\infty} \rho^k \pi_0 = \frac{\pi_0}{1-\rho} = 1$

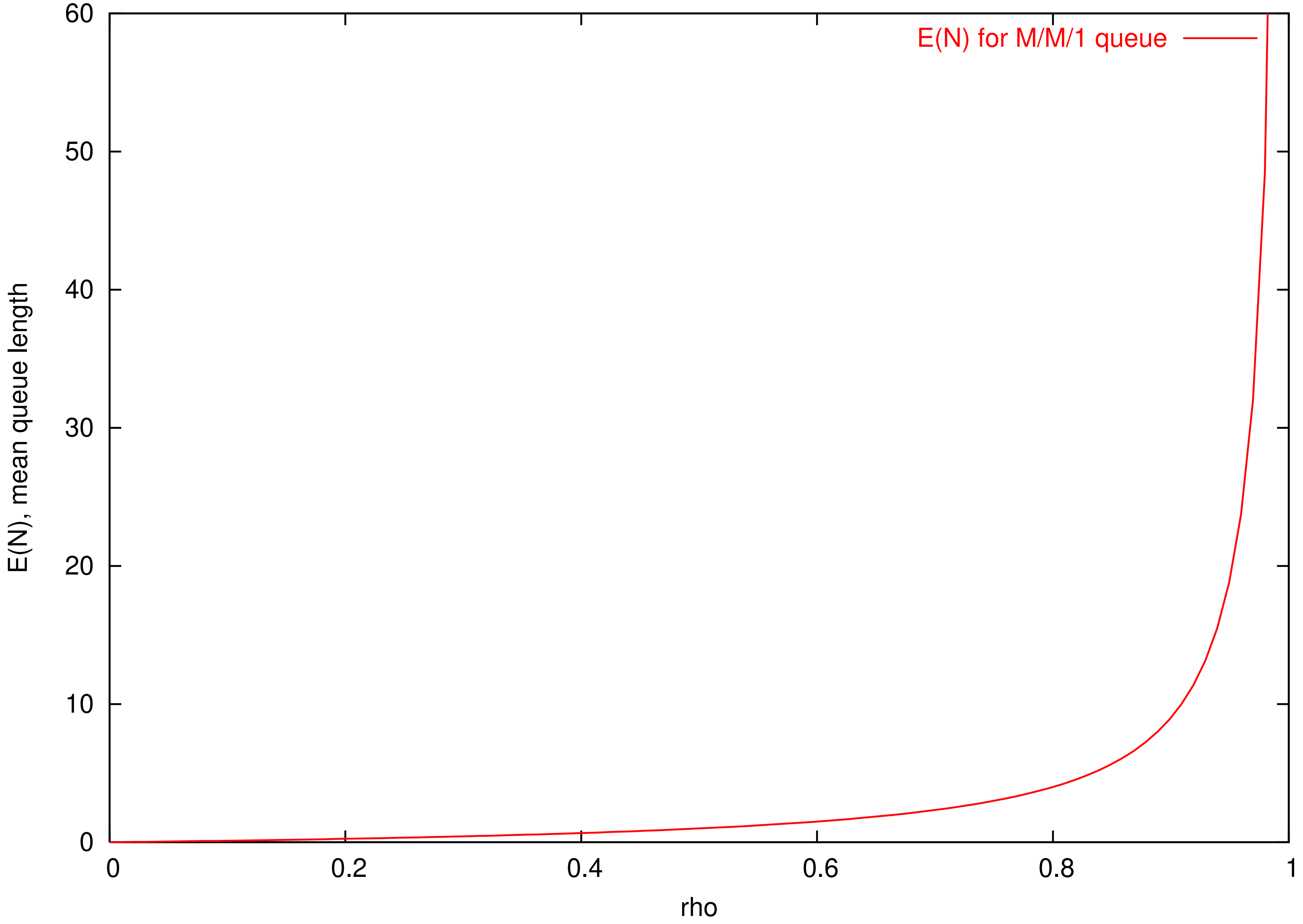
$\Rightarrow$ $\pi_0 = 1 - \rho$ as long as $\rho < 1$

M/M/1 Mean Queue Length

- N is queue length random variable
- N could be 0 or 1 or 2 or 3 ...
- Mean queue length is written $\mathbb{E}(N)$:

$$
\begin{aligned}
\mathbb{E}(N) &= 0.\mathbb{P}(\text{in state 0}) + 1.\mathbb{P}(\text{in state 1}) + 2.\mathbb{P}(\text{in state 2}) + \cdots \\
&= \sum_{k=0}^{\infty} k\pi_k \\
&= \pi_0 \sum_{k=0}^{\infty} k\rho^k = \pi_0\rho \sum_{k=0}^{\infty} k\rho^{k-1} = \pi_0\rho \sum_{k=0}^{\infty} \frac{\mathrm{d}}{\mathrm{d}\rho}\rho^k \\
&= \pi_0\rho \frac{\mathrm{d}}{\mathrm{d}\rho} \sum_{k=0}^{\infty} \rho^k = \pi_0\rho \frac{\mathrm{d}}{\mathrm{d}\rho} \left(\frac{1}{1-\rho} \right) \\
&= \frac{\pi_0\rho}{(1-\rho)^2} = \frac{\rho}{1-\rho} \quad \square
\end{aligned}
$$

M/M/1 Mean Queue Length

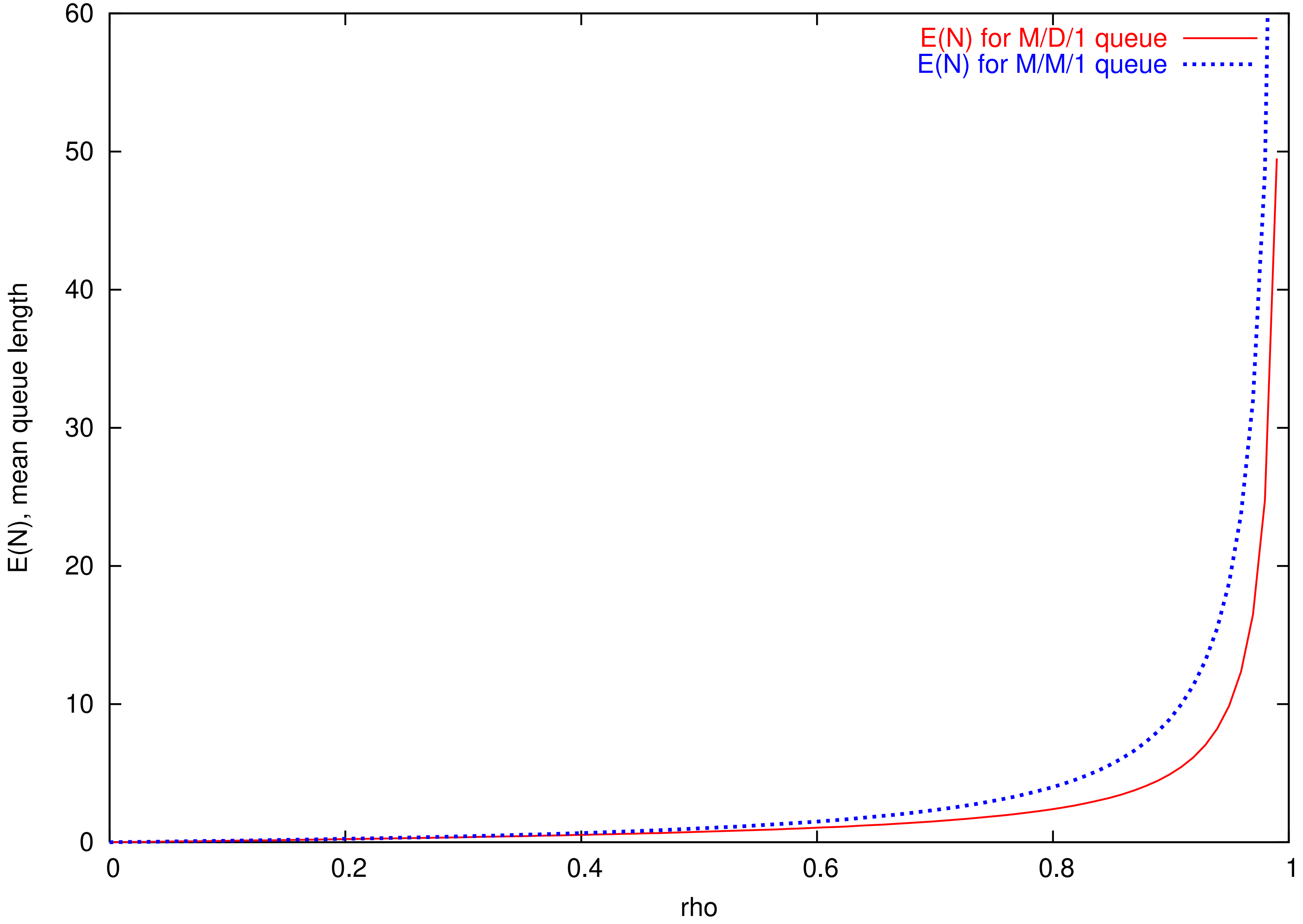


M/D/1 Queue Model

- M/M/1 queue has Markovian arrivals – a plausable assumption
- Markovian servicing – less likely from a computer-driven system
- M/D/1 might be a better approximation – deterministic servicing e.g. a service 10 times a second
- For M/D/1:

$$\mathbb{E}(N) = \rho + \frac{\rho^2}{2(1-\rho)}$$

M/D/1 Queue versus M/M/1 queue



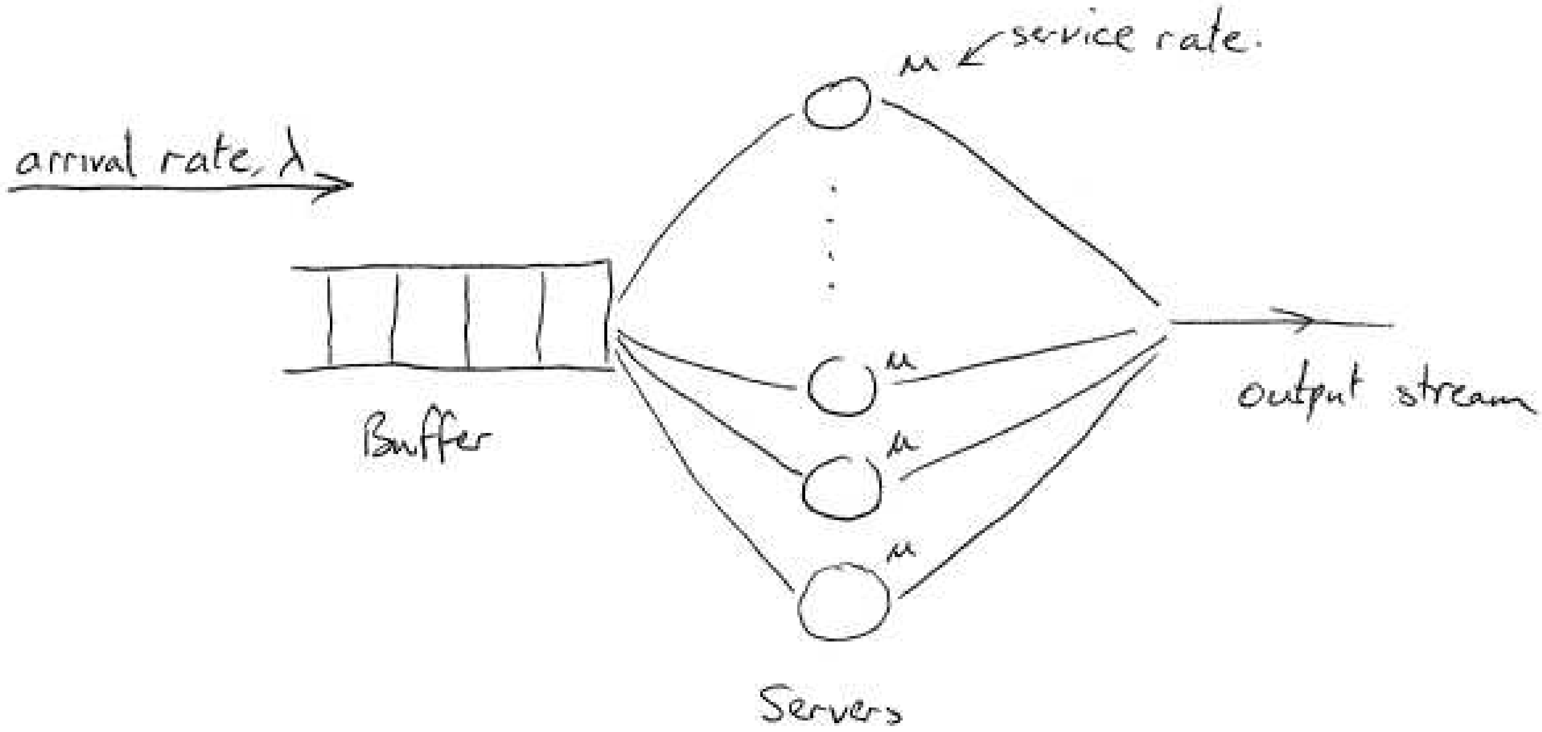
M/D/1: Back to Google

- What does this mean for service rate?
  - $\rho + \frac{\rho^2}{2(1-\rho)} < L$

  $\Rightarrow \rho < L + 1 - \sqrt{L^2 + 1}$
  - Dec. 2000: $\lambda = 1000$, $\mathbb{E}(N) < 400 \Rightarrow \mu > 1001.252$
  - Feb. 2004: $\lambda = 2000$, $\mathbb{E}(N) < 800 \Rightarrow \mu > 2001.251$
  - c.f. $\mu > 1002.5, 2002.5$ for M/M/1 queue
- i.e. we can get away with dealing with requests slightly more slowly if we can guarantee a different overall service discipline

Modelling many servers: M/M/n

- A cluster would be better represented by a many-server model (Fig. 8.5 [Tri])
- An M/M/n model has a single buffer and n servers



Mean queue length for M/M/n

- The mean queue length for an M/M/n queue is given by:

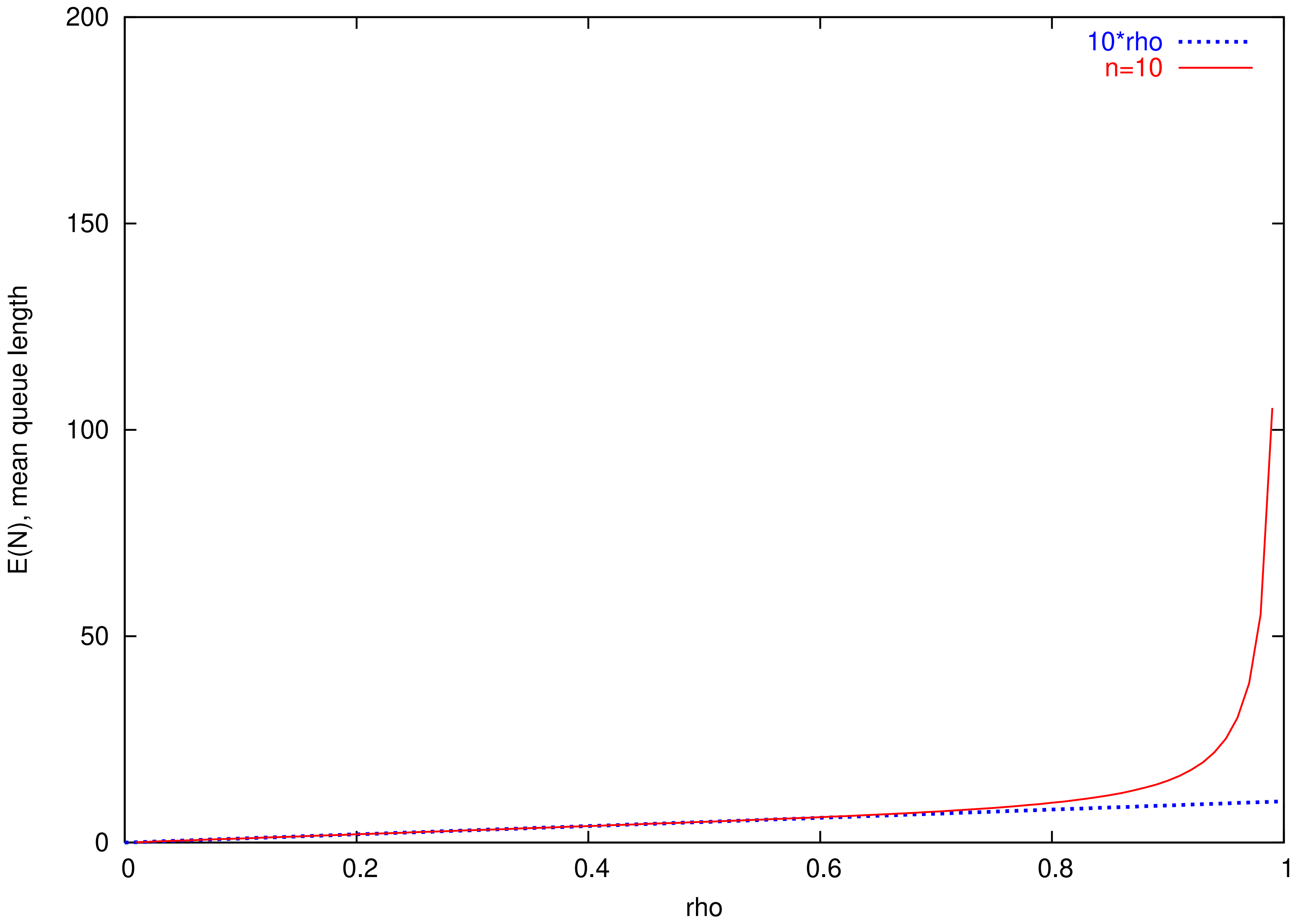
$$\mathbb{E}(N) = n\rho + \rho\frac{(n\rho)^n}{n!}\frac{\pi_0}{(1-\rho)^2}$$

- where $\rho = \lambda/(n\mu)$ and:

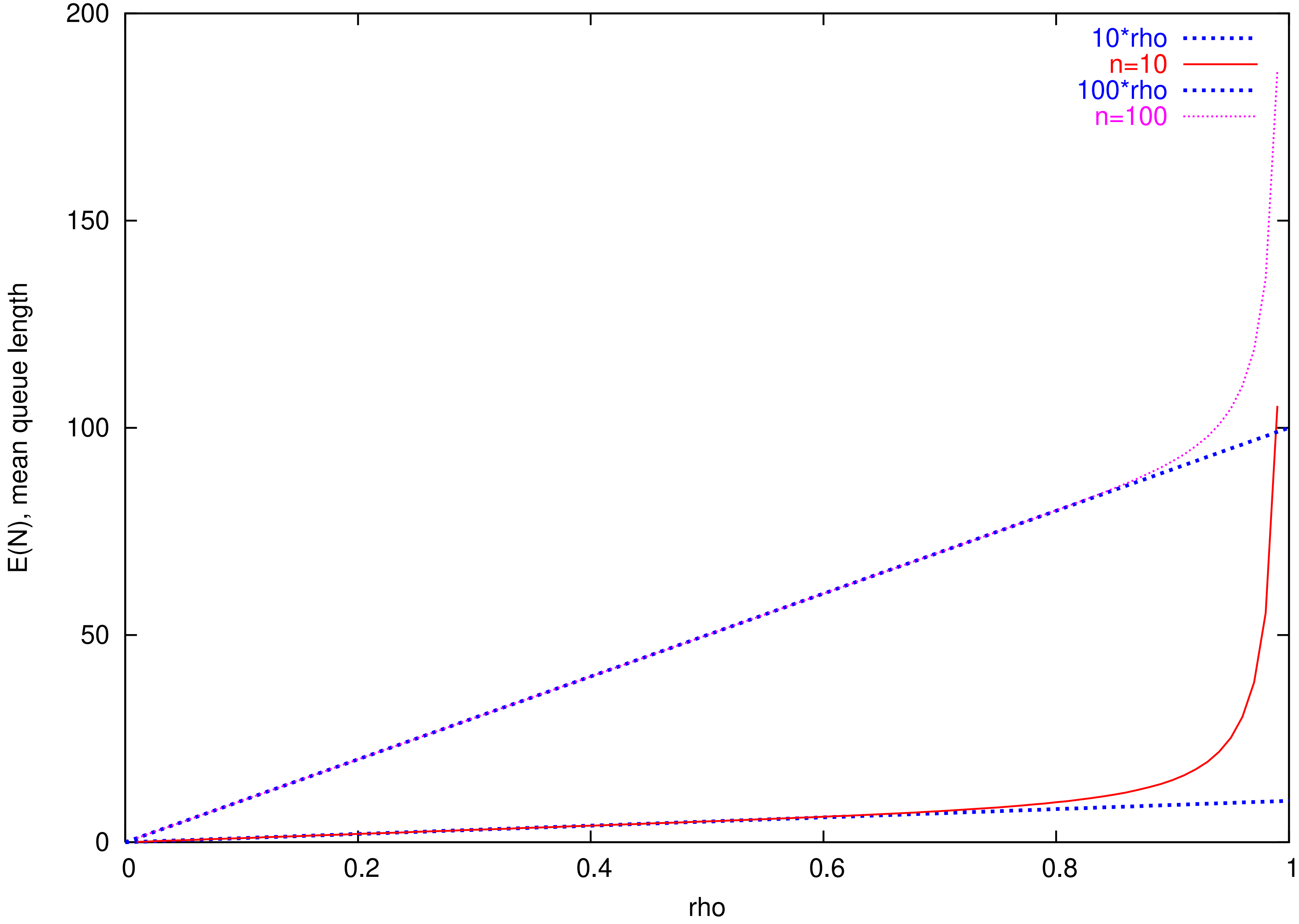
$$\pi_0 = \left[\sum_{k=0}^{n-1}\frac{(n\rho)^k}{k!} + \rho\frac{(n\rho)^n}{n!}\frac{1}{1-\rho}\right]^{-1}$$

- For further details: Trivedi 2002

Mean queue length: 10 servers



Mean queue length: +100 servers



Approximating $\mathbb{E}(N)$

- The mean queue length for an M/M/n queue is given by:

$$\mathbb{E}(N) = n\rho + \underbrace{\rho\frac{(n\rho)^n}{n!}\frac{\pi_0}{(1-\rho)^2}}_{\text{almost 0 if } \rho \text{ small}}$$

- In fact we can say that for small ρ:

$$\mathbb{E}(N) \sim n\rho$$

- if $E(N) < L$ for some λ then $\mu > \lambda/L$ as long as the number of processors exceeds the average buffer size.

How fast is a single processor?

- Dec. 2000: $\lambda = 1000$, $\mathbb{E}(N) < 400 \Rightarrow \mu > 2.5$
- Feb. 2004: $\lambda = 2000$, $\mathbb{E}(N) < 800 \Rightarrow \mu > 2.5$
- i.e. fast enough to service 2.5 requests per second

- But how good a model is M/M/n for representing a cluster computer?
- If you only need, say, 2000 processors to cope with peek demand over the mean – what are the other 3000 doing?