CS 147:
Computer Systems Performance Analysis
Introduction to Queueing Theory

CS 147:
Computer Systems Performance Analysis
Introduction to Queueing Theory

Overview

Introduction and Terminology
- Poisson Distributions

Fundamental Results
- Stability
- Little's Law

M/M/*
- M/M/1
- M/M/m
- M/M/m/B

More General Queues

Overview

Introduction and Terminology
Poisson Distributions
Fundamental Results
Stability
Little's Law
M/M/*
M/M/1
M/M/m
M/M/m/B
More General Queues

What is a Queueing System?

- A *queueing system* is any system in which things arrive, hang around for a while, and leave
- Examples
  - A bank
  - A freeway
  - A (computer) network
  - A beehive
- The things that arrive and leave are *customers* or *jobs*
- Customers leave after receiving *service*
- Most queueing systems have (surprise!) a *queue* that can store (delay) customers awaiting service

What is a Queueing System?
- A *queueing system* is any system in which things arrive, hang around for a while, and leave
- Examples
  - A bank
  - A freeway
  - A (computer) network
  - A beehive
- The things that arrive and leave are *customers* or *jobs*
- Customers leave after receiving *service*
- Most queueing systems have (surprise!) a *queue* that can store (delay) customers awaiting service

Parameters of a Queueing System

Arrival Process Injects customers into system
- Usually statistical
- Convenient to specify in terms of *interarrival time* distribution
- Most common is *Poisson* arrivals

Service Time Also statistical

Number of Servers Often 1

System Capacity Equals number of servers plus queue capacity. Often assumed infinite for convenience

Population Maximum number of customers. Often assumed infinite

Service Discipline How next customer is chosen for service. Often FCFS or priority

Parameters of a Queueing System

Arrival Process Injects customers into system
- Usually statistical
- Convenient to specify in terms of *interarrival time* distribution
- Most common is *Poisson* arrivals

Service Time Also statistical

Number of Servers Often 1

System Capacity Equals number of servers plus queue capacity. Often assumed infinite for convenience

Population Maximum number of customers. Often assumed infinite

Service Discipline How next customer is chosen for service. Often FCFS or priority

Arrival and Service Distributions

- Customer arrivals are random variables
  - Next disk request from many processes
  - Next packet hitting Google
  - Next call to Chipotle
- Same is true for service times
- What distribution describes it?
  - May be complicated (fractal, Zipf)
  - We often use Poisson for tractability

Arrival and Service Distributions

- Customer arrivals are random variables
  - Next disk request from many processes
  - Next packet hitting Google
  - Next call to Chipotle
- Same is true for service times
- What distribution describes it?
  - May be complicated (fractal, Zipf)
  - We often use Poisson for tractability

The Poisson Distribution

- Probability of exactly k arrivals in $(0, t)$ is $\mathrm{P}_k(t) = (\lambda t)^k e^{\lambda t}/k!$
  - λ is *arrival rate* parameter
- More useful formulation is *Poisson arrival distribution*:
  - PDF $A(t) = \mathrm{P}[\text{next arrival takes time} \leq t] = 1 - e^{-\lambda t}$
  - pdf $a(t) = \lambda e^{-\lambda t}$
  - Also known as *exponential* or *memoryless* distribution
  - Mean $=$ standard deviation $= \lambda$
- Poisson distribution is *memoryless*
  - Assume P[arrival within 1 second] at time $t_0 = x$
  - Then P[arrival within 1 second] at time $t_1 > t_0$ is also x
    - I.e., no memory that time has passed
  - Often true in real world
    - E.g., when I go to Von's doesn't affect when you go

The Poisson Distribution

- Probability of exactly k arrivals in $(0, t)$ is $\mathrm{P}_k(t) = (\lambda t)^k e^{\lambda t}/k!$
  - λ is *arrival rate* parameter
- More useful formulation is *Poisson arrival distribution*:
  - PDF $A(t) = \mathrm{P}[\text{next arrival takes time} \leq t] = 1 - e^{-\lambda t}$
  - pdf $a(t) = \lambda e^{-\lambda t}$
  - Also known as *exponential* or *memoryless* distribution
  - Mean $=$ standard deviation $= \lambda$
- Poisson distribution is *memoryless*
  - Assume P[arrival within 1 second] at time $t_0 = x$
  - Then P[arrival within 1 second] at time $t_1 > t_0$ is also x
    - I.e., no memory that time has passed
  - Often true in real world
    - E.g., when I go to Von's doesn't affect when you go

Splitting and Merging Poisson Processes

- Merging streams of Poisson events (e.g., arrivals) is Poisson

$$\lambda = \sum_{i=1}^{k} \lambda_i$$

- Splitting a Poisson stream randomly gives Poisson streams; if stream i has probability p_i, then

$$\lambda_i = p_i \lambda$$

Splitting and Merging Poisson Processes

- Merging streams of Poisson events (e.g., arrivals) is Poisson

$$\lambda = \sum_{i=1}^{k} \lambda_i$$

- Splitting a Poisson stream randomly gives Poisson streams; if stream i has probability p_i, then

$$\lambda_i = p_i \lambda$$

Kendall's Notation

A/S/m/B/K/D defines a (single) queueing system compactly:

- *A* Denotes arrival distribution, as follows:
  - *M* Exponential (*M*emoryless)
  - E_k Erlang with parameter k
  - *D* Deterministic
  - *G* Completely general (very hard to analyze!)
- *S* Service distribution, same as arrival
- *m* Number of servers
- *B* System capacity; ∞ if omitted
- *K* Population size; ∞ if omitted
- *D* Service discipline, FCFS if omitted

Kendall's Notation
A/S/m/B/K/D defines a (single) queueing system compactly:
A Denotes arrival distribution, as follows:
M Exponential (Memoryless)
E_k Erlang with parameter k
D Deterministic
G Completely general (very hard to analyze!)
S Service distribution, same as arrival
m Number of servers
B System capacity; ∞ if omitted
K Population size; ∞ if omitted
D Service discipline, FCFS if omitted

Examples of Kendall's Notation

- **D/D/1** Arrivals on clock tick, fixed service times, one server
- **M/M/m** Memoryless arrivals, memoryless service, multiple servers (good model of a bank)
- **M/M/m/m** Customers go away rather than wait in line
- **G/G/1** Modern disk drive

Examples of Kendall's Notation
D/D/1 Arrivals on clock tick, fixed service times, one server
M/M/m Memoryless arrivals, memoryless service, multiple servers (good model of a bank)
M/M/m/m Customers go away rather than wait in line
G/G/1 Modern disk drive

Common Variables

- τ Interarrival time. Usually varies per customer, e.g., $\tau_1, \tau_2, \ldots$
- λ Mean arrival rate: $1/\overline{\tau}$
- s_i Service time for job i, sometimes called x_i
- μ Mean service rate per server, $1/\overline{s}$
- ρ Traffic intensity or system load = $\lambda/m\mu$. This is the most important parameter in most queueing systems
- w_i Waiting time, or time in queue: interval between arrival and beginning of service
- r_i Response time $= w_i + s_i$

Common Variables

- τ Interarrival time. Usually varies per customer, e.g., $\tau_1, \tau_2, \ldots$
- λ Mean arrival rate: $1/\overline{\tau}$
- s_i Service time for job i, sometimes called x_i
- μ Mean service rate per server, $1/\overline{s}$
- ρ Traffic intensity or system load = $\lambda/m\mu$. This is the most important parameter in most queueing systems
- w_i Waiting time, or time in queue: interval between arrival and beginning of service
- r_i Response time $= w_i + s_i$

Stability

- A system is stable iff $\lambda < m\mu \equiv \rho < 1$
- Otherwise, system can't keep up and queue grows to ∞
- Exception: in D/D/m, $\rho = 1$ is OK

Stability

- A system is stable iff $\lambda < m\mu \equiv \rho < 1$
- Otherwise, system can't keep up and queue grows to ∞
- Exception: in D/D/m, $\rho = 1$ is OK

Little's Law

- Let $n =$ Number of jobs in system
- Then $n = \lambda \bar{r}$
- Likewise, if $n_q =$ Number of jobs in queue, then $n_q = \lambda \bar{w}$
- True regardless of distributions, queueing disciplines, etc., as long as system is in equilibrium
- May seem obvious:
  - If ten people are ahead of you in line, and each takes about 1 minute for service, you're going to be stuck there for 10 minutes
- Not proved until 1961
- Often useful for calculating queue lengths:
  - Packet takes 2s to arrive, you're sending 100 pps

  $\Rightarrow$ Mean queue length $= 100\ \text{pkt}/\text{s} \times 2\text{s} = 200$ pkts

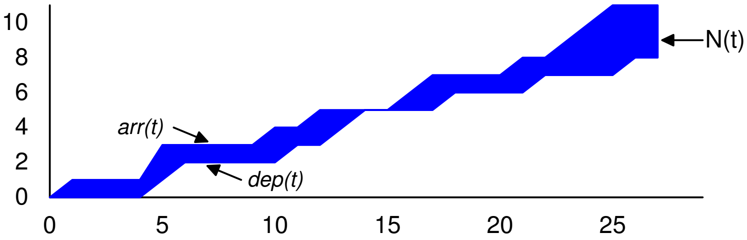
Little's Law

- Let $n =$ Number of jobs in system
- Then $n = \lambda \bar{r}$
- Likewise, if $n_q =$ Number of jobs in queue, then $n_q = \lambda \bar{w}$
- True regardless of distributions, queueing disciplines, etc., as long as system is in equilibrium
- May seem obvious:
  - If ten people are ahead of you in line, and each takes about 1 minute for service, you're going to be stuck there for 10 minutes
- Not proved until 1961
- Often useful for calculating queue lengths:
  - Packet takes 2s to arrive, you're sending 100 pps

  $\Rightarrow$ Mean queue length $= 100\ \text{pkt}/\text{s} \times 2\text{s} = 200$ pkts

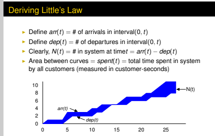
Deriving Little's Law

- Define $arr(t)$ = # of arrivals in interval$(0, t)$
- Define $dep(t)$ = # of departures in interval$(0, t)$
- Clearly, $N(t)$ = # in system at time$t$ = $arr(t) - dep(t)$
- Area between curves = $spent(t)$ = total time spent in system by all customers (measured in customer-seconds)



2015-06-15

CS147
└Fundamental Results
 └Little's Law
  └Deriving Little's Law



Deriving Little's Law (continued)

- Define average arrival rate during interval t, in customers/second, as $\lambda_t = arr(t)/t$
- Define T_t as system time/customer, averaged over all customers in $(0, t)$
  - Since $spent(t)$ = accumulated customer-seconds, divide by arrivals up to that point to get $T_t = spent(t)/arr(t)$
- Mean tasks in system over $(0, t)$ is accumulated customer-seconds divided by seconds: $\textit{Mean-tasks}_t = spent(t)/t$
- Above three equations give us:

$$\begin{aligned}\textit{Mean-tasks}_t &= spent(t)/t \\ &= T_t arr(t)/t \\ &= \lambda_t T_t\end{aligned}$$

Deriving Little's Law (continued)
- Define average arrival rate during interval t, in customers/second, as $\lambda_t = arr(t)/t$
- Define T_t as system time/customer, averaged over all customers in $(0, t)$
  - Since $spent(t)$ = accumulated customer-seconds, divide by arrivals up to that point to get $T_t = spent(t)/arr(t)$
- Mean tasks in system over $(0, t)$ is accumulated customer-seconds divided by seconds: $\textit{Mean-tasks}_t = spent(t)/t$
- Above three equations give us:
$$\begin{aligned}\textit{Mean-tasks}_t &= spent(t)/t \\ &= T_t arr(t)/t \\ &= \lambda_t T_t\end{aligned}$$

Deriving Little's Law (continued)

- We've shown that $\textit{Mean-tasks}_t = \lambda_t T_t$
- Assuming limits of λ_t and T_t exist, limit of $\textit{mean-tasks}_t$ also exists and gives Little's result:

$$\textit{Mean tasks in system} = \textit{arrival rate} \times \textit{mean time in system}$$

Deriving Little's Law (continued)
- We've shown that $\textit{Mean-tasks}_t = \lambda_t T_t$
- Assuming limits of λ_t and T_t exist, limit of $\textit{mean-tasks}_t$ also exists and gives Little's result:

$$\textit{Mean tasks in system} = \textit{arrival rate} \times \textit{mean time in system}$$

The M/M/1 Queue

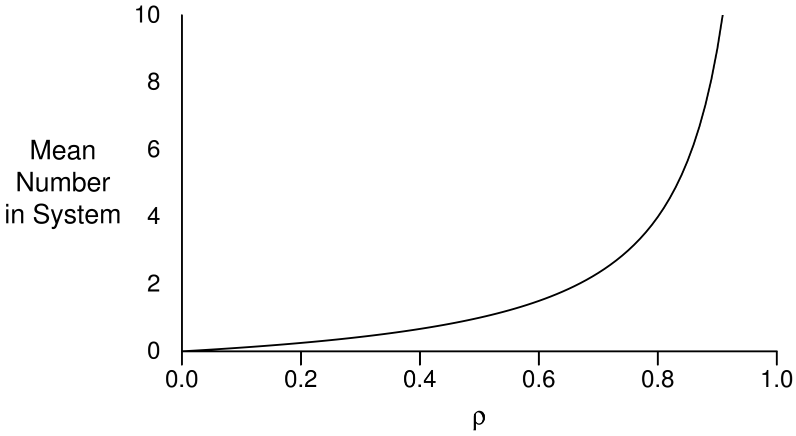
- Remember this one if you don't remember anything else
- Assumptions are sometimes realistic, sometimes not
  - Never infinite customers or capacity
  - Service times aren't truly Poisson
  - Interarrival times more likely to be Poisson
- Still provides surprisingly good analysis
- M/M/1's characteristics are clue to many other queues
- Primary results (in equilibrium):
  - Mean number in system $\overline{n} = \rho/(1-\rho)$
  - Mean time in system $\overline{r} = (1/\mu)/(1-\rho) = 1/\mu(1-\rho) = \overline{s}/(1-\rho)$

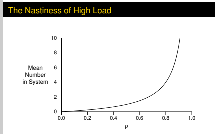
The M/M/1 Queue

- Remember this one if you don't remember anything else
- Assumptions are sometimes realistic, sometimes not
  - Never infinite customers or capacity
  - Service times aren't truly Poisson
  - Interarrival times more likely to be Poisson
- Still provides surprisingly good analysis
- M/M/1's characteristics are clue to many other queues
- Primary results (in equilibrium):
  - Mean number in system $\overline{n} = \rho/(1-\rho)$
  - Mean time in system $\overline{r} = (1/\mu)/(1-\rho) = 1/\mu(1-\rho) = \overline{s}/(1-\rho)$

Nearly all useful results in queueing theory apply only to systems in equilibrium.

The Nastiness of High Load





The system breaks down completely at $\rho > 0.95$.

The reason for the breakdown is variance: at high load, a burst fills the queue and it takes a long time to drain, giving plenty of time for another burst to arrive.

More M/M/1 Results

- Variance is $\rho/(1-\rho)^2$, so standard deviation is $\sqrt{\rho}/(1-\rho)$
- q-percentile of time in system is $\bar{r}\ln[100/(100-q)]$
  - 90^{th} percentile is $2.3\bar{r}$
- Mean *waiting* time is $\overline{w} = \frac{1}{\mu}\frac{\rho}{1-\rho}$
- q-percentile of waiting time is

$$\max\left(0, \frac{\overline{w}}{\rho}\ln[100\rho/(100-q)]\right)$$

- Mean jobs served in a busy period: $1/(1-\rho)$
- Probability of n jobs in system $p_n = (1-\rho)\rho^n$
- Probability of $> n$ jobs in system: ρ^n

More M/M/1 Results

- Variance is $\rho/(1-\rho)^2$, so standard deviation is $\sqrt{\rho}/(1-\rho)$
- q-percentile of time in system is $\bar{r}\ln[100/(100-q)]$
  - 90^{th} percentile is $2.3\bar{r}$
- Mean *waiting* time is $\overline{w} = \frac{1}{\mu}\frac{\rho}{1-\rho}$
- q-percentile of waiting time is

$$\max\left(0, \frac{\overline{w}}{\rho}\ln[100\rho/(100-q)]\right)$$

- Mean jobs served in a busy period: $1/(1-\rho)$
- Probability of n jobs in system $p_n = (1-\rho)\rho^n$
- Probability of $> n$ jobs in system: ρ^n

M/M/1 Example

- Web server gets 1200 requests/hour w/ Poisson arrivals
- Typical request takes 1s to serve
- $\rho = 1200/3600 = 0.33$
- Mean requests in service = $0.33/0.67 = 0.5$
- Mean response time $\bar{r} = (1/1)/(1 - 0.33) = 1.5$s
- 90$^{\text{th}}$ percentile response time = 3.4s

M/M/1 Example

- Web server gets 1200 requests/hour w/ Poisson arrivals
- Typical request takes 1s to serve
- $\rho = 1200/3600 = 0.33$
- Mean requests in service = $0.33/0.67 = 0.5$
- Mean response time $\bar{r} = (1/1)/(1 - 0.33) = 1.5$s
- 90$^{\text{th}}$ percentile response time = 3.4s

This slide has animations.

M/M/1 Example

- Web server gets 1200 requests/hour w/ Poisson arrivals
- Typical request takes 1s to serve
- $\rho = 1200/3600 = 0.33$
- Mean requests in service = $0.33/0.67 = 0.5$
- Mean response time $\bar{r} = (1/1)/(1 - 0.33) = 1.5$s
- 90^{th} percentile response time = 3.4s
- But if Slashdot hits…

M/M/1 Example

- Web server gets 1200 requests/hour w/ Poisson arrivals
- Typical request takes 1s to serve
- $\rho = 1200/3600 = 0.33$
- Mean requests in service = $0.33/0.67 = 0.5$
- Mean response time $\bar{r} = (1/1)/(1 - 0.33) = 1.5$s
- 90^{th} percentile response time = 3.4s
- But if Slashdot hits…

This slide has animations.

M/M/1 Example (cont'd)

- Suppose Slashdot raises request rate to 3500/hr
- Now $\rho = 3500/3600 = 0.972$
- Mean requests in service = $0.972/(1 - 0.972) = 34.7$
- $\bar{r} = 1/0.028 = 35.7$ seconds
- 90^{th} percentile response time $= 82.8$s

M/M/1 Example (cont'd)

- Suppose Slashdot raises request rate to 3500/hr
- Now $\rho = 3500/3600 = 0.972$
- Mean requests in service = $0.972/(1 - 0.972) = 34.7$
- $\bar{r} = 1/0.028 = 35.7$ seconds
- 90^{th} percentile response time $= 82.8$s

This slide has animations.

M/M/1 Example (cont'd)

- Suppose Slashdot raises request rate to 3500/hr
- Now $\rho = 3500/3600 = 0.972$
- Mean requests in service = $0.972/(1 - 0.972) = 34.7$
- $\bar{r} = 1/0.028 = 35.7$ seconds
- 90^{th} percentile response time $= 82.8$s
- And don't even *think* about 4000 requests/hr

M/M/1 Example (cont'd)

- Suppose Slashdot raises request rate to 3500/hr
- Now $\rho = 3500/3600 = 0.972$
- Mean requests in service = $0.972/(1 - 0.972) = 34.7$
- $\bar{r} = 1/0.028 = 35.7$ seconds
- 90^{th} percentile response time $= 82.8$s
- And don't even *think* about 4000 requests/hr

This slide has animations.

M/M/m

- Multiple servers, one queue
- $\rho = \lambda/(m\mu)$
- We'll need probability of empty system:

$$p_0 = \frac{1}{\dfrac{(m\rho)^m}{m!(1-\rho)} + \displaystyle\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}}$$

- Probability of queueing:

$$\varrho = P(\geq m \text{ jobs}) = \frac{(m\rho)^m}{m!(1-\rho)} p_0$$

M/M/m
- Multiple servers, one queue
- $\rho = \lambda/(m\mu)$
- We'll need probability of empty system:
$$p_0 = \frac{1}{\dfrac{(m\rho)^m}{m!(1-\rho)} + \displaystyle\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}}$$
- Probability of queueing:
$$\varrho = P(\geq m \text{ jobs}) = \frac{(m\rho)^m}{m!(1-\rho)} p_0$$

For $m = 1$, $\varrho = \rho$

M/M/m (cont'd)

- Mean jobs in system: $\overline{n} = m\rho + \rho\varrho/(1-\rho)$
- Mean time in system:

$$\overline{r} = \frac{1}{\mu}\left(1 + \frac{\varrho}{m(1-\rho)}\right)$$

- Mean waiting time: $\overline{w} = \varrho/[m\mu(1-\rho)]$
- q-percentile of waiting time:

$$\max\left(0, \frac{\overline{w}}{\varrho} \ln \frac{100\varrho}{100-q}\right)$$

M/M/m (cont'd)
- Mean jobs in system: $\overline{n} = m\rho + \rho\varrho/(1-\rho)$
- Mean time in system:
$$\overline{r} = \frac{1}{\mu}\left(1 + \frac{\varrho}{m(1-\rho)}\right)$$
- Mean waiting time: $\overline{w} = \varrho/[m\mu(1-\rho)]$
- q-percentile of waiting time:
$$\max\left(0, \frac{\overline{w}}{\varrho} \ln \frac{100\varrho}{100-q}\right)$$

$m\times$M/M/1 vs. M/M/m

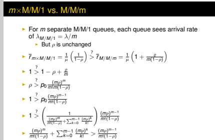
- For m separate M/M/1 queues, each queue sees arrival rate of $\lambda_{M/M/1} = \lambda/m$
  - But ρ is unchanged
- $\overline{r}_{m\times M/M/1} = \frac{1}{\mu}\left(\frac{1}{1-\rho}\right) \overset{?}{>} \overline{r}_{M/M/m} = \frac{1}{\mu}\left(1 + \frac{\varrho}{m(1-\rho)}\right)$
- $1 \overset{?}{>} 1 - \rho + \frac{\varrho}{m}$
- $\rho \overset{?}{>} p_0 \frac{(m\rho)^m}{m!m(1-\rho)}$
- $1 \overset{?}{>} p_0 \frac{(m\rho)^{m-1}}{m!(1-\rho)}$
- $1 \overset{?}{>} \left(\frac{1}{\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}}\right) \frac{(m\rho)^{m-1}}{m!(1-\rho)}$

$m\times$M/M/1 vs. M/M/m
- For m separate M/M/1 queues, each queue sees arrival rate of $\lambda_{M/M/1} = \lambda/m$
  - But ρ is unchanged
- $\overline{r}_{m\times M/M/1} = \frac{1}{\mu}\left(\frac{1}{1-\rho}\right) \overset{?}{>} \overline{r}_{M/M/m} = \frac{1}{\mu}\left(1 + \frac{\varrho}{m(1-\rho)}\right)$
- $1 \overset{?}{>} 1 - \rho + \frac{\varrho}{m}$
- $\rho \overset{?}{>} p_0 \frac{(m\rho)^m}{m!m(1-\rho)}$
- $1 \overset{?}{>} p_0 \frac{(m\rho)^{m-1}}{m!(1-\rho)}$
- $1 \overset{?}{>} \left(\frac{1}{\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}}\right) \frac{(m\rho)^{m-1}}{m!(1-\rho)}$

This slide has animations.

$m\times$M/M/1 vs. M/M/m

- For m separate M/M/1 queues, each queue sees arrival rate of $\lambda_{M/M/1} = \lambda/m$
  - But ρ is unchanged
- $\bar{r}_{m\times M/M/1} = \frac{1}{\mu}\left(\frac{1}{1-\rho}\right) \stackrel{?}{>} \bar{r}_{M/M/m} = \frac{1}{\mu}\left(1 + \frac{\varrho}{m(1-\rho)}\right)$
- $1 \stackrel{?}{>} 1 - \rho + \frac{\varrho}{m}$
- $\rho \stackrel{?}{>} p_0 \frac{(m\rho)^m}{m!m(1-\rho)}$
- $1 \stackrel{?}{>} p_0 \frac{(m\rho)^{m-1}}{m!(1-\rho)}$
- $1 \stackrel{?}{>} \left(\frac{1}{\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}}\right) \frac{(m\rho)^{m-1}}{m!(1-\rho)}$
- $\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} > \frac{(m\rho)^{m-1}}{m!(1-\rho)}$



This slide has animations.

$m\times$M/M/1 vs. M/M/m

- For m separate M/M/1 queues, each queue sees arrival rate of $\lambda_{M/M/1} = \lambda/m$
  - But ρ is unchanged
- $\bar{r}_{m\times M/M/1} = \frac{1}{\mu}\left(\frac{1}{1-\rho}\right) \overset{?}{>} \bar{r}_{M/M/m} = \frac{1}{\mu}\left(1 + \frac{\varrho}{m(1-\rho)}\right)$
- $1 \overset{?}{>} 1 - \rho + \frac{\varrho}{m}$
- $\rho \overset{?}{>} p_0 \frac{(m\rho)^m}{m!m(1-\rho)}$
- $1 \overset{?}{>} p_0 \frac{(m\rho)^{m-1}}{m!(1-\rho)}$
- $1 \overset{?}{>} \left(\frac{1}{\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!}}\right)\frac{(m\rho)^{m-1}}{m!(1-\rho)}$
- $\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} > \frac{(m\rho)^{m-1}}{m!(1-\rho)}$

$m\times$M/M/1 vs. M/M/m
- For m separate M/M/1 queues, each queue sees arrival rate of $\lambda_{M/M/1} = \lambda/m$
  - But ρ is unchanged
- $\bar{r}_{m\times M/M/1} = \frac{1}{\mu}\left(\frac{1}{1-\rho}\right) \overset{?}{>} \bar{r}_{M/M/m} = \frac{1}{\mu}\left(1 + \frac{\varrho}{m(1-\rho)}\right)$
- $1 \overset{?}{>} 1 - \rho + \frac{\varrho}{m}$
- $\rho \overset{?}{>} p_0 \frac{(m\rho)^m}{m!m(1-\rho)}$
- $1 \overset{?}{>} p_0 \frac{(m\rho)^{m-1}}{m!(1-\rho)}$
- $1 \overset{?}{>} \left(\frac{1}{\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!}}\right)\frac{(m\rho)^{m-1}}{m!(1-\rho)}$
- $\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} > \frac{(m\rho)^{m-1}}{m!(1-\rho)}$

This slide has animations.

$m\times$M/M/1 vs. M/M/m

- For m separate M/M/1 queues, each queue sees arrival rate of $\lambda_{M/M/1} = \lambda/m$
  - But ρ is unchanged
- $\overline{r}_{m\times M/M/1} = \frac{1}{\mu}\left(\frac{1}{1-\rho}\right) \overset{?}{>} \overline{r}_{M/M/m} = \frac{1}{\mu}\left(1 + \frac{\varrho}{m(1-\rho)}\right)$
- $1 \overset{?}{>} 1 - \rho + \frac{\varrho}{m}$
- $\rho \overset{?}{>} p_0 \frac{(m\rho)^m}{m!m(1-\rho)}$
- $1 \overset{?}{>} p_0 \frac{(m\rho)^{m-1}}{m!(1-\rho)}$
- $1 \overset{?}{>} \left(\frac{1}{\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}}\right) \frac{(m\rho)^{m-1}}{m!(1-\rho)}$
- $\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{\color{red}{k=0}}^{m-1} \frac{(m\rho)^k}{k!} > \frac{(m\rho)^{m-1}}{m!(1-\rho)}$

2015-06-15

CS147
└M/M/*
  └M/M/m
    └$m\times$M/M/1 vs. M/M/m

This slide has animations.

Running Some Numbers

- Assume 5 servers, $\rho = 0.5$, $\mu = 1$
- Then $\bar{r}_{m\times M/M/1} = 1/(1-\rho) = 2$
- $\varrho = \frac{(m\rho)^m}{m!(1-\rho)} p_0 = \frac{(2.5)^5}{5!(0.5)} p_0 = \frac{97.7}{60} p_0 = 1.63 p_0$
- $p_0 = \frac{1}{\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}} = \frac{1}{1.63+1+\frac{2.5^1}{1}+\frac{2.5^2}{2}+\frac{2.5^3}{3!}+\frac{2.5^4}{4!}}$
- $p_0 = \frac{1}{1.63+1+2.5+3.13+2.60+1.63} = \frac{1}{12.49} = 0.08$
- So $\varrho = 1.63(0.08) = 0.13$
- And $\bar{r}_{m/M/m} = 1 + \frac{\varrho}{m(1-\rho)} = 1 + \frac{0.13}{5(1-0.5)} = 1 + \frac{0.13}{2.5} = 1.05$
- In terms of previous slide's inequality, $\frac{97.7}{60}+1+2.5+3.13+2.60+1.63 = 12.49 > \frac{2.5^4}{5!(0.5)} = \frac{39.1}{60} = 0.65$

Running Some Numbers

- Assume 5 servers, $\rho = 0.5$, $\mu = 1$
- Then $\bar{r}_{m\times M/M/1} = 1/(1-\rho) = 2$
- $\varrho = \frac{(m\rho)^m}{m!(1-\rho)} p_0 = \frac{(2.5)^5}{5!(0.5)} p_0 = \frac{97.7}{60} p_0 = 1.63 p_0$
- $p_0 = \frac{1}{\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}} = \frac{1}{1.63+1+\frac{2.5^1}{1}+\frac{2.5^2}{2}+\frac{2.5^3}{3!}+\frac{2.5^4}{4!}}$
- $p_0 = \frac{1}{1.63+1+2.5+3.13+2.60+1.63} = \frac{1}{12.49} = 0.08$
- So $\varrho = 1.63(0.08) = 0.13$
- And $\bar{r}_{m/M/m} = 1 + \frac{\varrho}{m(1-\rho)} = 1 + \frac{0.13}{5(1-0.5)} = 1 + \frac{0.13}{2.5} = 1.05$
- In terms of previous slide's inequality, $\frac{97.7}{60}+1+2.5+3.13+2.60+1.63 = 12.49 > \frac{2.5^4}{5!(0.5)} = \frac{39.1}{60} = 0.65$

$m\times$M/M/1 vs. M/M/m (cont'd)

- A similar result holds for variance
- Conclusion: single queue, multiple server is always better than one queue per server
- Question 1: When is this false? (hint: multiple cores)
- Question 2: Why do so many movie theaters have multiple lines for popcorn?

$m\times$M/M/1 vs. M/M/m (cont'd)

- A similar result holds for variance
- Conclusion: single queue, multiple server is always better than one queue per server
- Question 1: When is this false? (hint: multiple cores)
- Question 2: Why do so many movie theaters have multiple lines for popcorn?

M/M/m/B

- Real systems have finite capacity
- Previous analysis applies only under light loads (relative to capacity)
- Considering limit has several effects:
  - Lost jobs (obviously)
  - Loss rate p_B becomes important parameter
  - Mean response time drops compared to M/M/m/∞ (Why?)

M/M/m/B

- Real systems have finite capacity
- Previous analysis applies only under light loads (relative to capacity)
- Considering limit has several effects:
  - Lost jobs (obviously)
  - Loss rate p_B becomes important parameter
  - Mean response time drops compared to M/M/m/∞ (Why?)

Extending the Results

- Unsurprisingly, generality equates to (mathematical) complexity
- Many special cases have been analyzed (e.g., Erlang distributions)
- Little's Law always applies
- Important cases:
  - M/G/1
  - M/D/1
  - G/G/m (but mostly intractable)

Extending the Results

- Unsurprisingly, generality equates to (mathematical) complexity
- Many special cases have been analyzed (e.g., Erlang distributions)
- Little's Law always applies
- Important cases:
  - M/G/1
  - M/D/1
  - G/G/m (but mostly intractable)