CS147 -90-5102

CS 147: omputer Systems Performance Analysis Introduction to Queueing Theory

CS 147: Computer Systems Performance Analysis Introduction to Queueing Theory



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



which is a docucing of sterring
 A queueing system is any system in which things arrive, I around for a while, and leave
Examplea A bank A bank A bank A betway A (computer) network A banhve
 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 (delaw) systems replice contine

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 Process Trijects customers into system
 Usually statistical
 Convenient to specify in terms of interavival to
distribution
 Most common is Poisson arrivals
Service Time Also statistical
Number of Servers: Often 1
System Capacity Equals number of servers plus queue capaci 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

CS147 L Introduction and Terminology CS147 L Introduction and Terminology

rival and Service Distributions

Customer arrivals are random variables
 Next diak request from many processes
 Next packet hiting Google
 Next call to Chipotle
 Same is true for service times
 What distribution describes it?
 May be complicated (truck), Zp()
 We often use Poisson (truck)

The Poisson Distribution

- Probability of exactly k arrivals in (0, t) is $P_k(t) = (\lambda t)^k e^{\lambda t} / k!$
 - λ is *arrival rate* parameter
- ▶ More useful formulation is *Poisson arrival distribution*:
 - ▶ PDF $A(t) = P[\text{next arrival takes time } \leq t] = 1 e^{-\lambda t}$
 - pdf $a(t) = \lambda e^{-\lambda t}$
 - Also known as exponential or memoryless distribution
 - Mean = standard deviation = λ
- Poisson distribution is *memoryless*
 - Assume P[arrival within 1 second] at time $t_0 = x$
 - Then P[arrival within 1 second] at time t₁ > t₀ 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 P_k(t) = (λt)^k e^{λt} λ is arrival rate parameter
 More useful formulation is Poisson arrival distribution: PDF A(t) = P[next arrival takes time ≤ t] = 1 − e^{-3t} pdf a(t) = 3e^{-3t} Also known as exponential or memorylase distribution Mean = standard division = 3
 Poisson distribution is memoryless Assume Pjarnval within 1 second] at time t₀ = x Then Pjarnval within 1 second] at time t₁ > t₁ is also x I.e., no memory that time has passed
 Often true in real world E.o. when I on to World down? affect when you on

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

 $\lambda_i = p_i \lambda$

$$\begin{array}{c} \mathsf{CS147} \\ \downarrow \\ \mathsf{Introduction and Terminology} \\ \downarrow \\ \mathsf{Poisson Distributions} \\ \downarrow \\ \mathsf{Splitting and Merging Poisson Processes} \end{array} \qquad \begin{array}{c} \mathsf{Splitting and Merging Poisson Processes} \\ \mathsf{A} = \mathsf{A}^{\mathsf{A}} \end{array}$$

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

CS147 - Introduction and Terminology - Poisson Distributions - Kendall's Notation

Kendall's Notation
A SimBKC delines a (single) queueing system compo A Denotes annual distribution, as follows: M Exponential (Memorytess) E, Ertang with parameter k D Deterministic G Correlately general (very han analyze)
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

Common Variables

- τ Interarrival time. Usually varies per customer, e.g., τ_1, τ_2, \dots
- λ Mean arrival rate: $1/\overline{ au}$
- 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 V	ariables
	Interarrival time. Usually varies per customer, τ_1, τ_2, \ldots
د ا	Mean arrival rate: 1/7
	Service time for job i, sometimes called x
	Mean service rate per server, 1/3
	Traffic intensity or system load = $\lambda/m\mu$. This is most important parameter in most queueing sy
	Waiting time, or time in queue: interval betwee arrival and beginning of service
	Response time = w + a

Fundamental Results

A system is stable iff λ < mµ = ρ < 1
 Otherwise, system can't keep up and queue grows to ∞
 Exception: in D/D/m, ρ = 1 is OK

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

Stability

Fundamental Results Little's Law

Little's Law

- Let n = Number of jobs in system
- Then $n = \lambda \overline{r}$
- ▶ Likewise, if n_q = Number of jobs in queue, then $n_q = \lambda \overline{W}$
- 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 pkt/s \times 2s = 200 pkts



Little's Law

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







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: Mean-tasks_t = spent(t)/t
- Above three equations give us:

 $Mean-tasks_t = spent(t)/t$ = $T_t arr(t)/t$ = $\lambda_t T_t$



Deriving Little's Law (continued)



iving Little's Law (continued

We've shown that Mean-tasks = $\lambda_1 T_1$

Assuming limits of λ_1 and T_2 exist, limit of mean-tasks, also exists and gives Little's result:

Mean tasks in system = arrival rate × mean time in system

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

Mean tasks in system = arrival rate \times 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

 $\bar{r} = (1/\mu)/(1-\rho) = 1/\mu(1-\rho) = \bar{s}/(1-\rho)$

CS147 └──M/M/* └──M/M/1 └──The M/M/1 Queue

M/M/1 Queue

Remember this one if you don't remember anything else Assumption are accombined as salidates, constellines not • Near infinite austionaries or capacity • Service Inness and 't sky' Reason • Innovanity limits more likely to be Polacen Sill provides arryinghing good analysis of the Assumption MMN's characteristics are ulso to many other quouse Primary nasular (in equilibrium); • Mann trains in system $\pi = (1/n)(-1) = 1/n(1-n) = \pi(1-n)$

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.

I/M/* M/M/1

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 $\overline{r} \ln[100/(100 q)]$
 - ▶ 90th percentile is $2.3\overline{r}$
- 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 µ/(1 − µ)²; so standard deviation is √7/(1 − • q-parcentile of time in system is 7 ln[100/(100 − q)] • 30ⁿ parcentile at 2.3ⁿ
 Mean waiting time is w = 1/2 from provide the second second
$\max\left(0, \frac{\overline{w}}{\rho} \ln[100\rho/(100-q)]\right)$
 Mean jobs served in a busy period: 1/(1 - p)
-11

Probability of > n jobs in system;

M/M/1 Example

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



Web server gets 100 requests/hour w/ Poisson am/web Tripical request tables 116 server e _ = 1200 3000 - 2013 - 2013 - 2013 Maar request is an value - 320 302 - 2 5 Maar request is an value - 320 302 - 5 5 O "O" performance requests in a - 3.5

M/M/1 Example

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



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
- 90th percentile response time = 82.8s

CS147 └──M/M/* └──M/M/1 └──M/M/1 Example (cont'd)

This slide has animations.

M/M/1 Example (cont'd)

 $\label{eq:sphere:sphe$

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 = 82.8s
- And don't even think about 4000 requests/hr



M/M/1 Example (cont'd)

Suppose Slashdot raises request rate to 3500hr New # = 3500/3600 = 0.972 Mean requests in service = 0.972/(1 = 0.972) = 34.7 # = 1/0.028 = 35.7 seconds 90% percentile response time = 82.8s And don't even think about 4000 requests/hr

M/M/m

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

$$p_0 = rac{1}{rac{(m
ho)^m}{m!(1-
ho)} + \sum_{k=0}^{m-1}rac{(m
ho)^k}{k!}}$$

Probability of queueing:

$$arrho = {\it P}(\geq m \, {
m jobs}) = rac{(m
ho)^m}{m!(1-
ho)} {\it p}_0$$

S147 ↓ M/M/* ↓ M/M/m ↓ M/M/m

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



M/M/*

M/M/m (cont'd)

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

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

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

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

CS147 2015-06-15 -M/M/* Mean jobs in system: $\overline{n} = m_{\theta} + e_{\theta}/(1 - e)$ Mean time in system: └─M/M/m Mean waiting time: $\overline{w} = \rho/[m\mu(1 - \rho)]$ -M/M/m (cont'd) g-percentile of waiting time;

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

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

$m \times M/M/1$ 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

$$\overline{r}_{m \times M/M/1} = \frac{1}{\mu} \left(\frac{1}{1-\rho} \right)^{2} \overline{r}_{M/M/m} = \frac{1}{\mu} \left(1 + \frac{\varrho}{m(1-\rho)} \right)^{2}$$

$$1^{2} - \rho + \frac{\varrho}{m}$$

$$\rho^{2} p_{0} \frac{(m\rho)^{m}}{m!m(1-\rho)}$$

$$1^{2} p_{0} \frac{(m\rho)^{m-1}}{m!(1-\rho)}$$

$$1^{2} \left(\frac{1}{\frac{(m\rho)^{m}}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^{k}}{k!}} \right) \frac{(m\rho)^{m-1}}{m!(1-\rho)}$$



$m \times M/M/1$ 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

$$\overline{r}_{m \times M/M/1} = \frac{1}{\mu} \left(\frac{1}{1-\rho} \right)^{2} \overline{r}_{M/M/m} = \frac{1}{\mu} \left(1 + \frac{\varrho}{m(1-\rho)} \right)^{2}$$

$$1 \stackrel{?}{>} 1 - \rho + \frac{\varrho}{m}$$

$$\rho \stackrel{?}{>} \rho_{0} \frac{(m\rho)^{m}}{m!(1-\rho)}$$

$$1 \stackrel{?}{>} \rho_{0} \frac{(m\rho)^{m-1}}{m!(1-\rho)}$$

$$1 \stackrel{?}{>} \left(\frac{1}{\frac{(m\rho)^{m}}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^{k}}{k!}} \right) \frac{(m\rho)^{m-1}}{m!(1-\rho)}$$

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

 $\begin{array}{c} \text{CS147} \\ \hline \text{M/M/*} \\ \hline \text{M/M/m} \\ \hline \text{M/M/m} \\ \hline \text{M/M/m} \\ \hline \text{M/M/m} \\ \hline \text{M/M/l vs. M/M/m} \end{array}$

M/M/1 vs. M/N

• $1 \stackrel{?}{>} 1 - \rho + \frac{\sigma}{m}$ • $\rho \stackrel{?}{>} \rho_0 \frac{(m_0)^m}{m(m_1 - \sigma)}$

1 2 ps⁽ⁿ⁾ⁿ⁻¹

of $\lambda_{M/M/1} = \lambda/m$

• But ρ is unchanged • $T_{m:M/M/1} = \frac{1}{n} \left(\frac{1}{1-\sigma}\right)^2 T_{M/M/m} = \frac{1}{n} \left(1 + \frac{\sigma}{m(1-\sigma)}\right)$

▶ $1 \ge \left(\frac{1}{\frac{2mp^{m-1}}{2m(1-p)}} \right) \frac{(mp)^{m-1}}{2m(1-p)}$

• $\frac{(m_p)^n}{4m(1-p)} + \sum_{k=0}^{m-1} \frac{(m_p)^k}{k!} > \frac{(m_p)^{m-1}}{4m(1-p)}$

For m separate M/M1 queues, each queue sees arrival rate

$m \times M/M/1$ 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

$$\overline{r}_{m \times M/M/1} = \frac{1}{\mu} \left(\frac{1}{1-\rho} \right)^{2} \overline{r}_{M/M/m} = \frac{1}{\mu} \left(1 + \frac{\varrho}{m(1-\rho)} \right)^{2}$$

$$1 \stackrel{?}{>} 1 - \rho + \frac{\varrho}{m}$$

$$\rho \stackrel{?}{>} p_{0} \frac{(m\rho)^{m}}{m!m(1-\rho)}$$

$$1 \stackrel{?}{>} p_{0} \frac{(m\rho)^{m-1}}{m!(1-\rho)}$$

$$1 \stackrel{?}{>} \left(\frac{1}{\frac{(m\rho)^{m}}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^{k}}{k!}} \right) \frac{(m\rho)^{m-1}}{m!(1-\rho)}$$

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

$m \times M/M/1$ 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

$$\overline{r}_{m \times M/M/1} = \frac{1}{\mu} \left(\frac{1}{1-\rho} \right)^{\frac{2}{2}} \overline{r}_{M/M/m} = \frac{1}{\mu} \left(1 + \frac{\varrho}{m(1-\rho)} \right)^{\frac{2}{2}}$$

$$1 \stackrel{?}{>} 1 - \rho + \frac{\varrho}{m}$$

$$\rho \stackrel{?}{>} p_0 \frac{(m\rho)^m}{m!m(1-\rho)}$$

$$1 \stackrel{?}{>} p_0 \frac{(m\rho)^{m-1}}{m!(1-\rho)}$$

$$1 \stackrel{?}{>} \left(\frac{1}{\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}} \right) \frac{(m\rho)^{m-1}}{m!(1-\rho)}$$

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

 $\begin{array}{c} \text{CS147} \\ \leftarrow \text{M/M/*} \\ \leftarrow \text{M/M/m} \\ \leftarrow \text{M/M/m} \\ \leftarrow m \times \text{M/M/1 vs. M/M/m} \end{array}$

M/M/1 vs. M/N

• $1 \stackrel{?}{>} 1 - \rho + \frac{\sigma}{m}$ • $\rho \stackrel{?}{>} \rho_0 \frac{(m_0)^m}{m(m_1 - \sigma)}$

1 2 ps(n)=1

of $\lambda_{M/M/1} = \lambda/m$

• But ρ is unchanged • $T_{m:M/M/1} = \frac{1}{n} \left(\frac{1}{1-\sigma}\right)^2 T_{M/M/m} = \frac{1}{n} \left(1 + \frac{\sigma}{m(1-\sigma)}\right)$

▶ $1 \ge \left(\frac{1}{\frac{2mp^{m-1}}{2m(1-p)}}\right) \frac{(mp)^{m-1}}{2m(1-p)}$

• $\frac{(m_p)^m}{4m(1-p)} + \sum_{k=0}^{m-1} \frac{(m_p)^k}{k!} > \frac{(m_p)^{m-1}}{4m(1-p)}$

For m separate M/M1 queues, each queue sees arrival rate

Running Some Numbers

- ▶ Assume 5 servers, $\rho = 0.5, \mu = 1$
- Then $\overline{r}_{m \times M/M/1} = 1/(1-\rho) = 2$ • $\rho = \frac{(m\rho)^m}{m!(1-\rho)} p_0 = \frac{(2.5)^5}{5!(0.5)} p_0 = \frac{97.7}{60} p_0 = 1.63 p_0$ $\blacktriangleright p_0 = \frac{1}{\frac{(m\rho)^m}{m!(1-\rho)} + \sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}} = \frac{1}{1.63 + 1 + \frac{2.51}{1} + \frac{2.52}{2} + \frac{2.53}{2!} + \frac{2.54}{4!}}$ $p_0 = \frac{1}{1.63 + 1 + 2.5 + 3.13 + 2.60 + 1.63} = \frac{1}{12.49} = 0.08$ **So** $\rho = 1.63(0.08) = 0.13$ • And $\overline{r}_{m/M/m} = 1 + \frac{\varrho}{m(1-\varrho)} = 1 + \frac{0.13}{5(1-0.5)} = 1 + \frac{0.13}{2.5} = 1.05$ In terms of previous slide's inequality, $\frac{97.7}{60}$ + 1 + 2.5 + 3.13 + 2.60 + 1.63 = 12.49 > $\frac{2.5^4}{51(0.5)}$ = $\frac{39.1}{60}$ = 0.65

unning Some Numbers

 $\begin{array}{l} \text{Assume } p_{i} = 0.5 \ \text{scales} p_{i} = 0.5$

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



- (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
- 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 popcern?

- 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



M/M/m/B

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

- 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)



ty equates to (mathemat
ve been analyzed (e.g., E
ies
tractable)