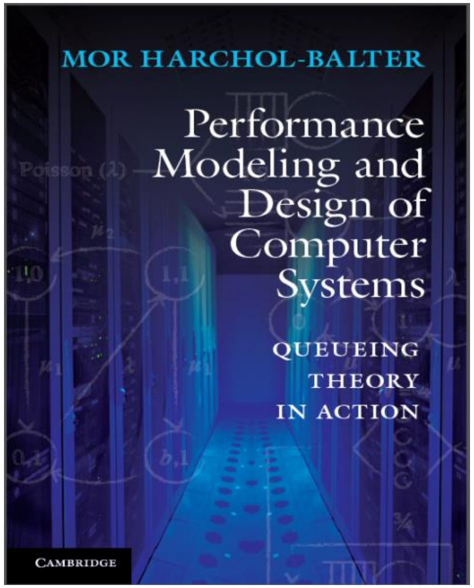
What Queueing Theory Teaches Us About Computer Systems Design

Mor Harchol-Balter
Computer Science Dept, CMU



Outline

I. Basic Vocabulary

- Avg arrival rate, λ
- Avg service rate, μ
- Avg load, ρ
- Avg throughput, X
- Open vs. closed systems
- Response time, T
- Waiting time, T_Q
- Exponential vs. Pareto/Heavy-tailed
- Squared coefficient of variation, C^2
- Poisson Process

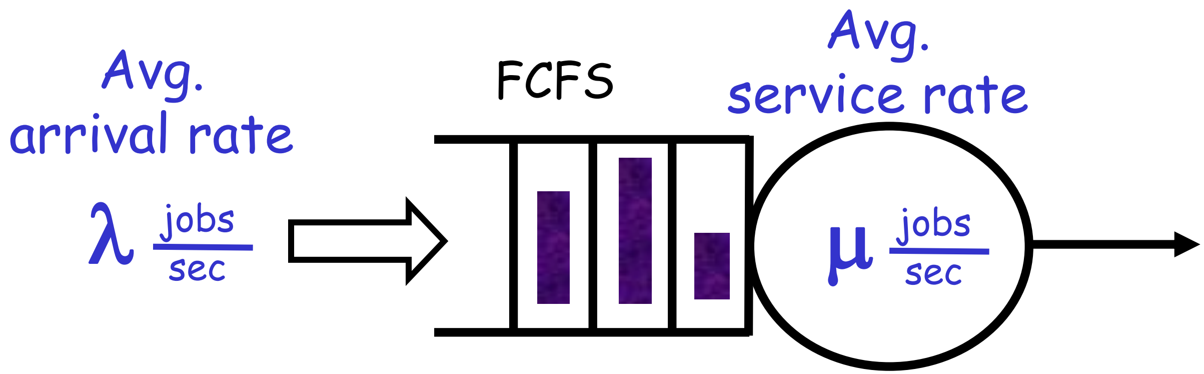
II. Single-server queues

- D/D/1, M/M/1, M/G/1
- Inspection Paradox
- Effect of job size variaibility
- Effect of load
- Provisioning bathrooms/scaling
- Scheduling: FCFS, PS, SJF, LAS, SRPT
- Web server scheduling implementation
- Open vs. closed systems: wait
- Open vs. closed systems: scheduling

III. Multi-server queues

- Static load balancing
- Throwing away servers
- M/M/k + Comparing architectures
- Many slow servers vs. 1 fast
- Capacity provisioning & scaling
- Square root staffing
- Dynamic power management
- Dynamic load balancing/FCFS servers
- Replication
- Dynamic load balancing/PS servers

Vocabulary



$\lambda < \mu$

throughout

S : job size (sec) = service requirement

$$E[S] = \frac{1}{\mu}$$

<u>Example</u>:

- On average, job needs 3×10^6 cycles
- Machine executes $9\text{X}10^6$ cycles/sec

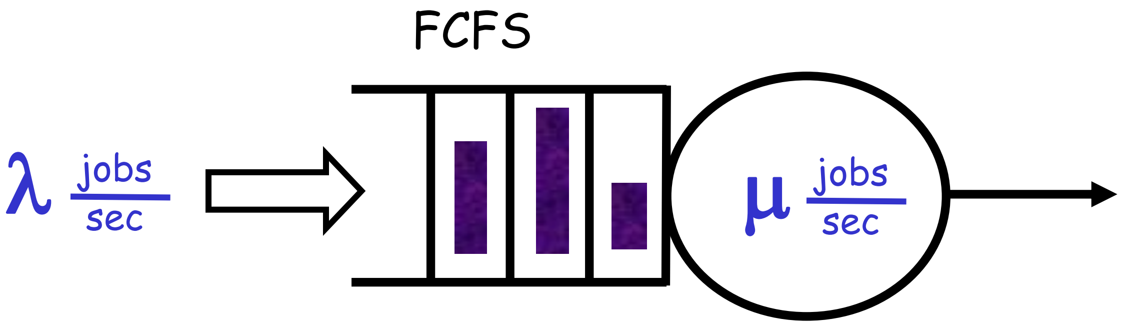
Avg service rate

$\mu = 3$ $\frac{\text{jobs}}{\text{sec}}$

Avg size of job on this server:

$E[S] = \frac{1}{3}$ sec.

Vocabulary



S : job size $\quad E[S] = \dfrac{1}{\mu}$

$$\rho = \text{Load (utilization)} = \text{Frac. time server busy} = \lambda E[S] = \frac{\lambda}{\mu}$$

Example:

- $\lambda = 2 \, \frac{\text{jobs}}{\text{sec}}$ arrive
- Each job requires $E[S] = \frac{1}{3}$ sec on avg

$\rho = \dfrac{2}{3}$

More Vocabulary

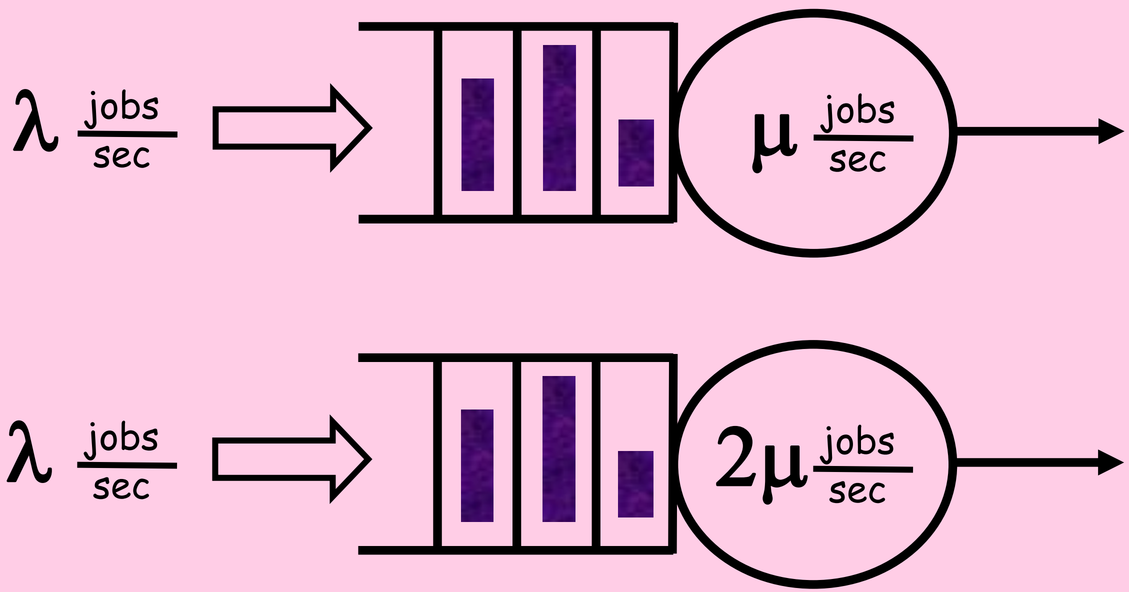
Defn: Throughput X denotes the average rate at which jobs complete (jobs/sec)

QUESTION:

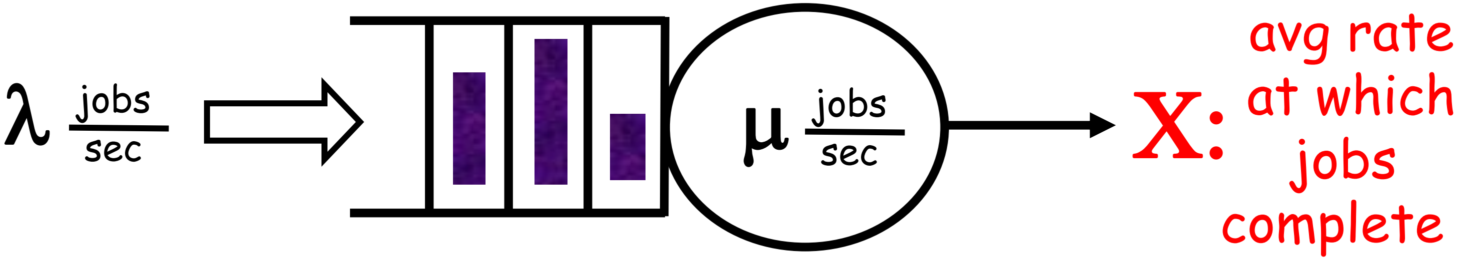
Which has higher throughput, X?

$\lambda < \mu$

throughout



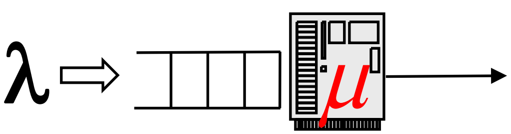
More Vocabulary



$$X = \lambda$$ (assuming no jobs dropped)

Open versus Closed Systems

Open

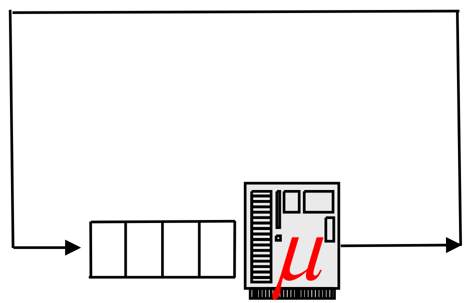


$$\rho = \lambda E[S] = \frac{\lambda}{\mu}$$

$$X = \lambda$$

Closed Batch

MPL N: fixed #jobs

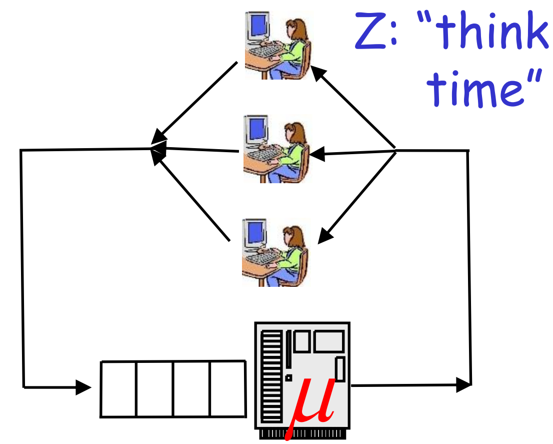


$$\rho = 1$$

$$X = \mu$$

Closed Interactive

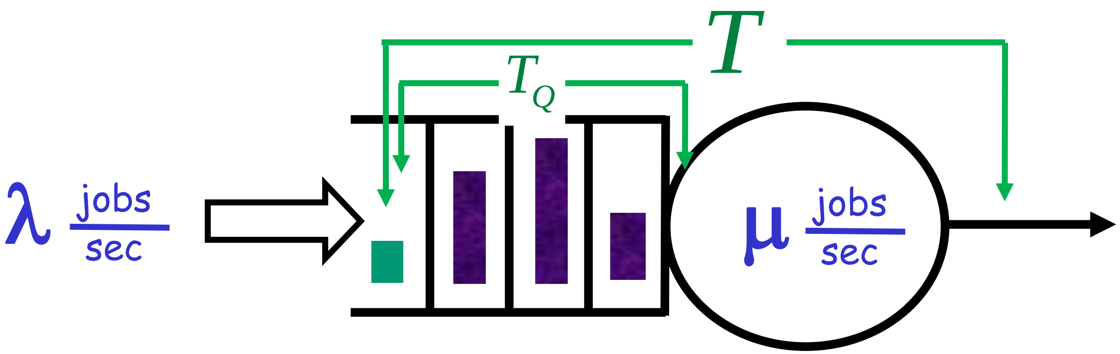
MPL N: fixed #users



$$\rho = 1 - \Pr\{\text{All thinking}\}$$

$$X = \rho\mu$$

More Vocabulary



S : job size

$$E[S] = \frac{1}{\mu}$$

$$\rho = \lambda E[S] = \frac{\lambda}{\mu}$$

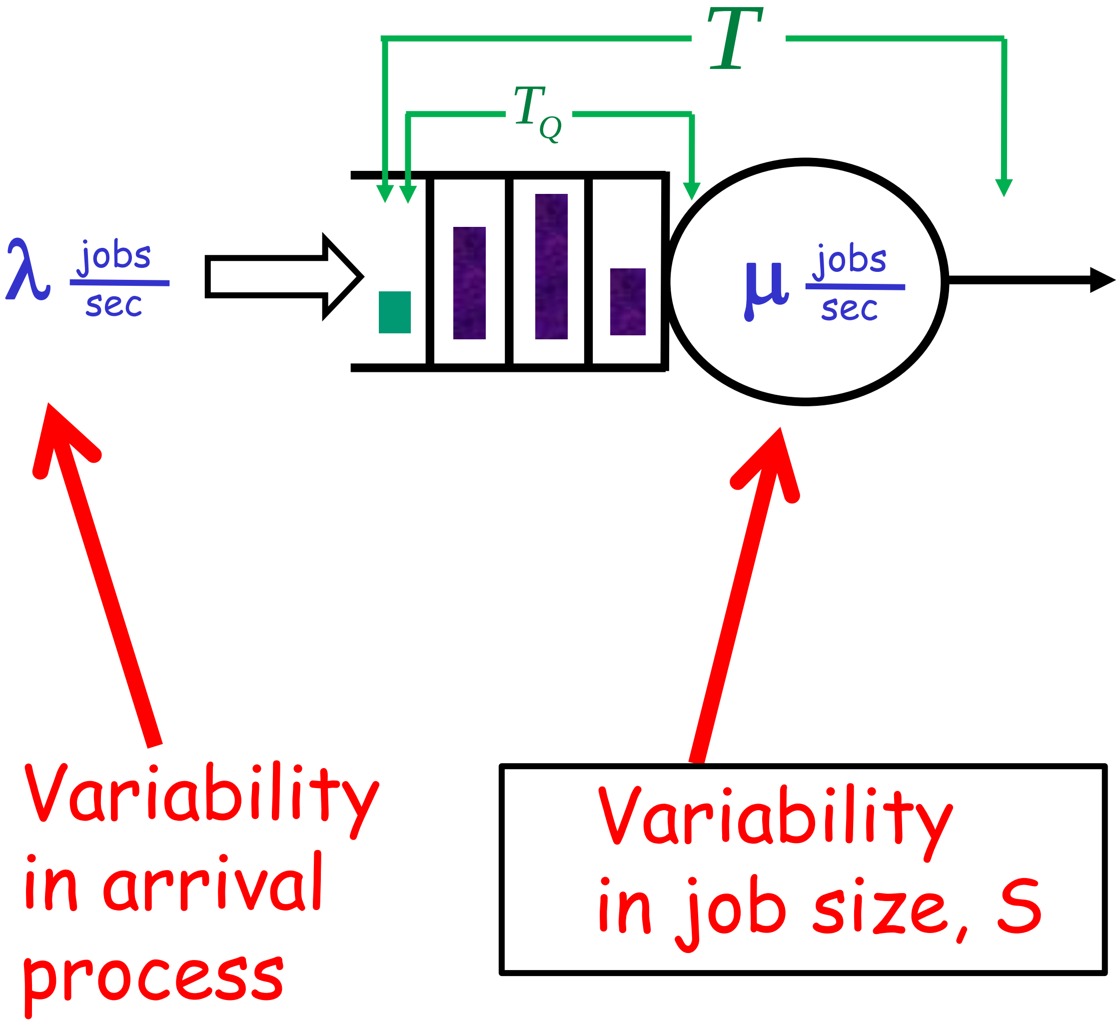
$T =$ response time

$T_Q =$ queueing time (waiting time)

Q: Given that $\lambda < \mu$, what causes wait?

A: Variability in the arrival process & service requirements

Variability



S : job size

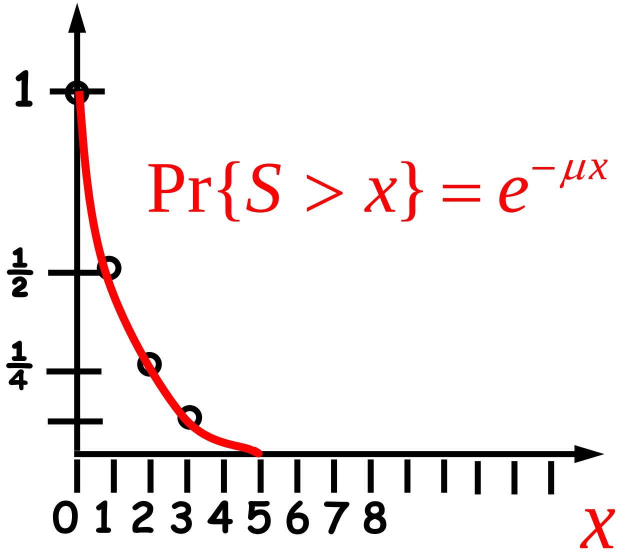
$$E[S] = \frac{1}{\mu}$$

$$\rho = \lambda E[S] = \frac{\lambda}{\mu}$$

Job Size Distributions

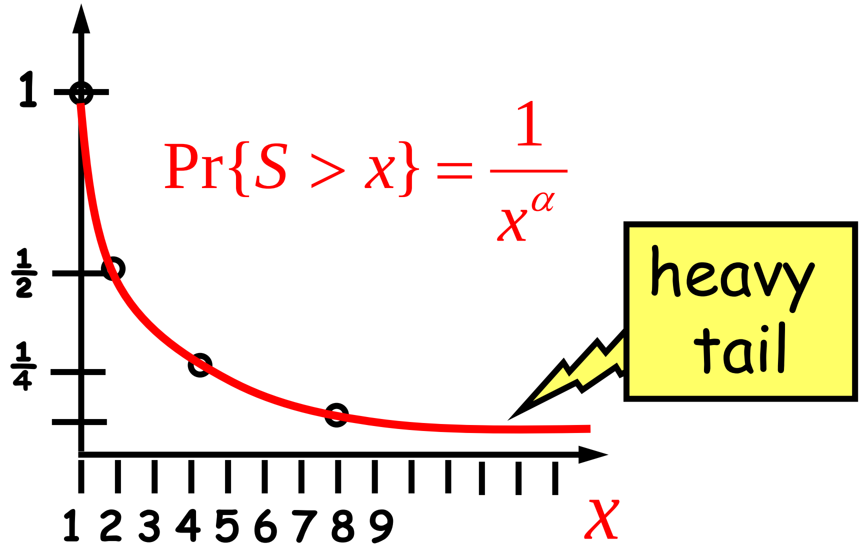
"Most jobs are small; few jobs are large"

$S \sim \text{Exp}(\mu)$



$$\Pr\{S > x\} = e^{-\mu x}$$

$S \sim \text{Pareto}(\alpha)$



$$\Pr\{S > x\} = \frac{1}{x^{\alpha}}$$

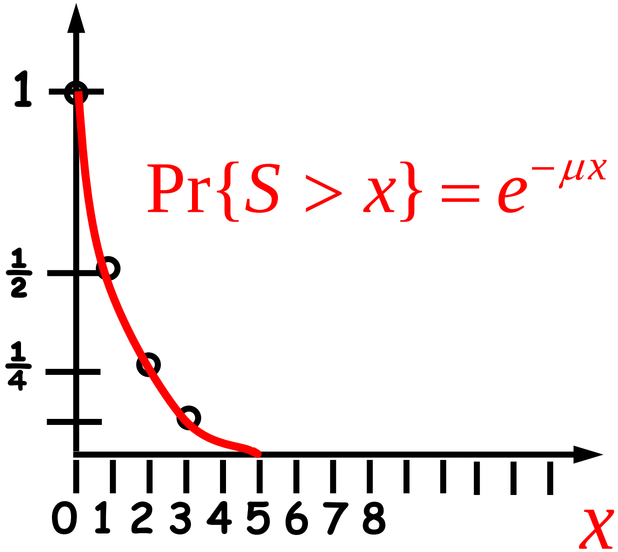
Job Size Distributions

QUESTION: Which best represents UNIX process lifetimes?

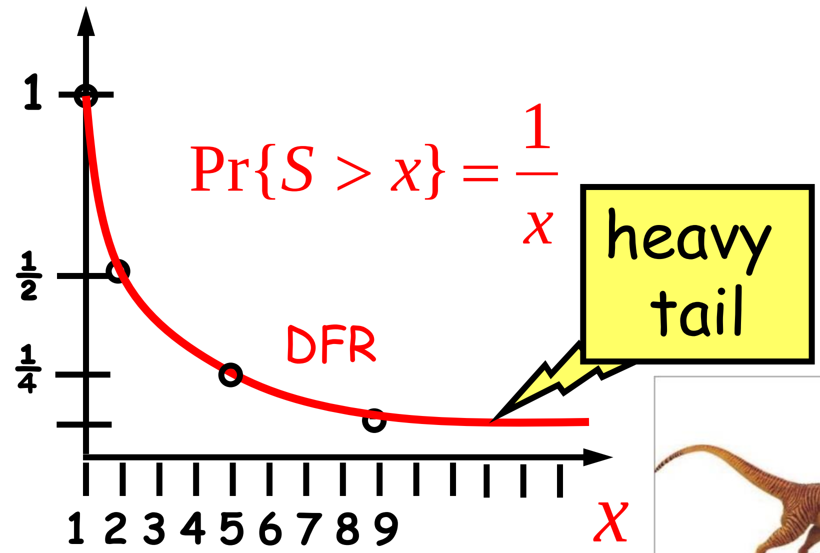
QUESTION: For which do top 1% of jobs comprise 50% of load?

QUESTION: Which distribution fits the saying, "the longer a job has run so far, the longer it is expected to continue to run."

$S \sim \text{Exp}(\mu)$



$S \sim \text{Pareto}(\alpha = 1)$



Pareto Job Size Distribution

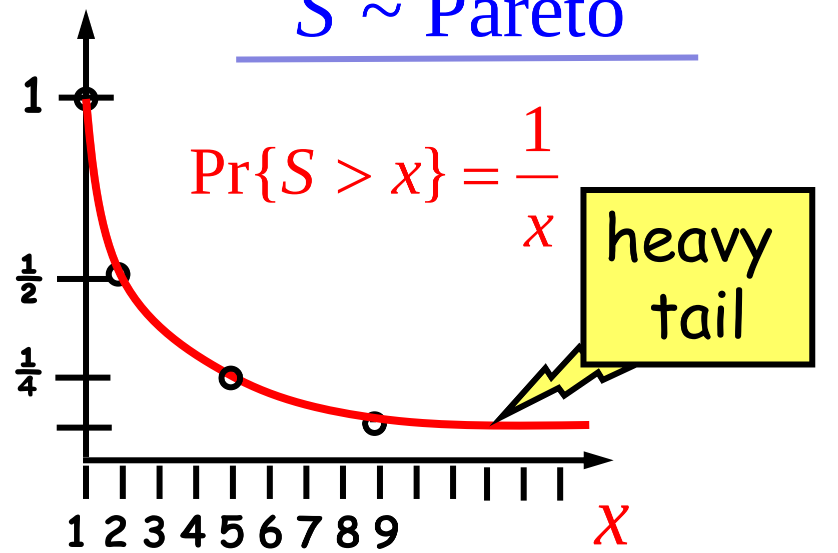
Pareto job sizes are ubiquitous in CS:

- CPU lifetimes of UNIX jobs [Harchol-Balter, Downey 96]
- Supercomputing job sizes [Schroeder, Harchol-Balter 00]
- Web file sizes [Crovella, Bestavros 98], [Barford, Crovella 98]
- IP flow durations [Shaikh, Rexford, Shin 99]
- Wireless call durations [Blinn, Henderson, Kotz 05]

Also ubiquitous in nature:

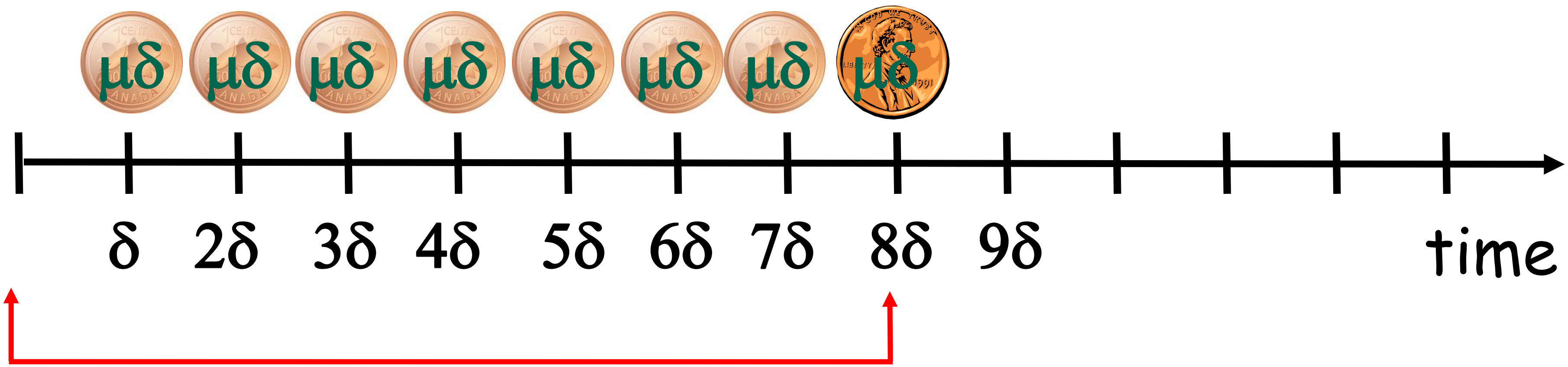
- Forest fire damage
- Earthquake damage
- Human wealth
  [Vilfredo Pareto '65]

$S \sim \text{Pareto}$



Exponential Job Size Distribution

$$S \sim \text{Exp}(\mu) \quad \Rightarrow \quad Pr\{S > x\} = e^{-\mu x}$$

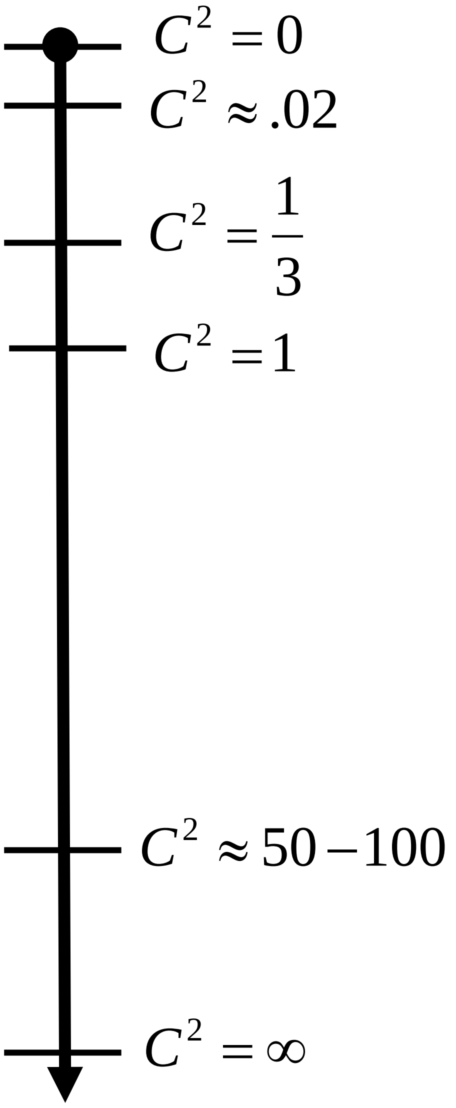


S is time until coin w/prob μδ comes up heads

$$E[S] = \frac{1}{\mu}$$

S is memoryless!

Variability in Job Sizes



Squared Coefficient of Variation $= C^2 = \dfrac{Var(S)}{E[S]^2}$

QUESTION:

Match these distributions to their C^2 values:

- Deterministic
- Exponential
- Uniform(0,b)
- Unix process lifetimes
- Human IQs
- Pareto distribution

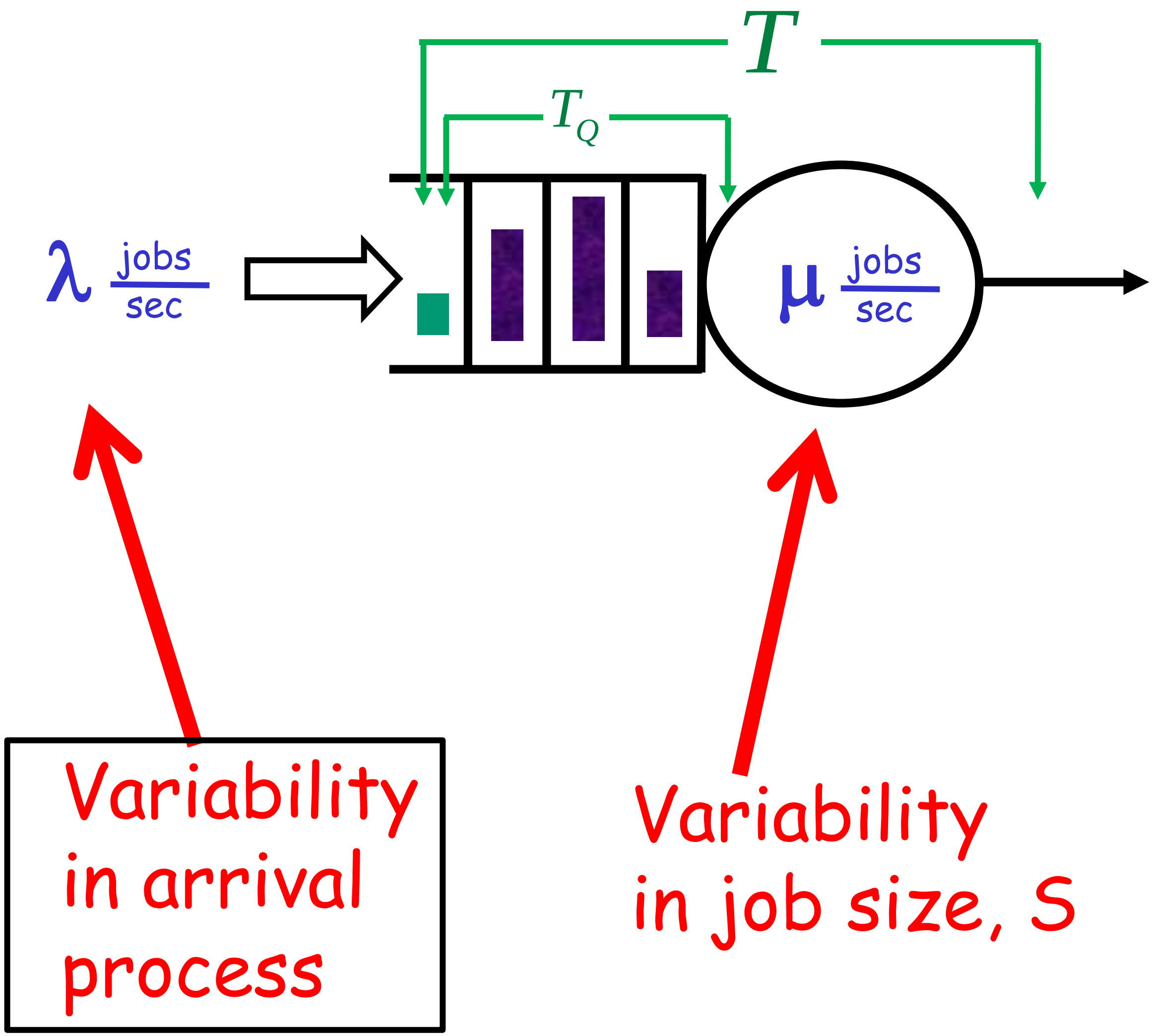
Variability in Job Sizes

$C^2 = 0$	Deterministic
$C^2 \approx .02$	Human IQs
$C^2 = \frac{1}{3}$	Uniform(0,b) - for any b
$C^2 = 1$	Exponential distribution
$C^2 = 50-100$	Unix process lifetimes
$C^2 = \infty$	Pareto distribution

Squared Coefficient of Variation

$$\mathrm{C}^2 = \frac{Var(S)}{E[S]^2}$$

Variability



S : job size

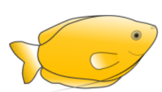
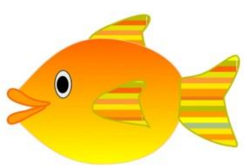
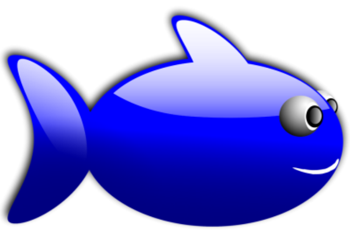
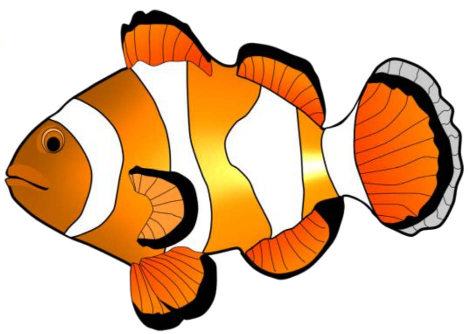
$$E[S] = \frac{1}{\mu}$$

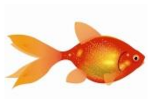
$$\rho = \lambda E[S] = \frac{\lambda}{\mu}$$

Poisson Process with rate λ

QUESTION: What's a Poisson process with rate λ?

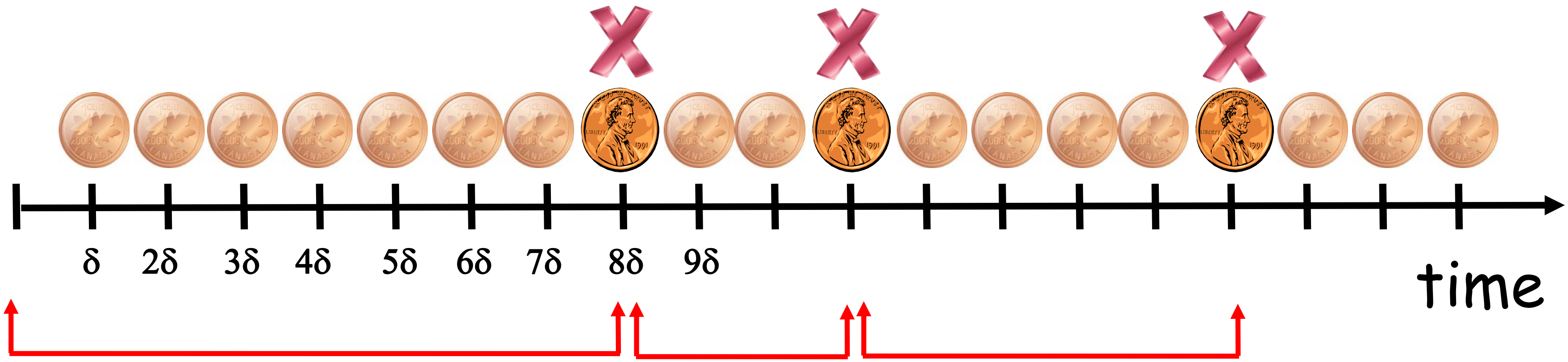
Hint: It's related to Exp(λ).



Poisson Process with rate λ

Poisson process models sequence of arrival times
(typically representing aggregation of many users)



$S \sim \text{Exp}(\lambda)$ $S \sim \text{Exp}(\lambda)$ $S \sim \text{Exp}(\lambda)$

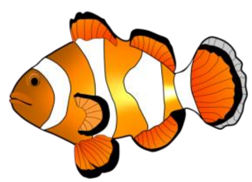
Summary Part I

I. Basic Vocabulary

- Avg arrival rate, λ
- Avg service rate, μ
- Avg load, ρ
- Avg throughput, X
- Open vs. closed systems
- Response time, T
- Waiting time, T_Q
- Exponential vs. Pareto/Heavy-tailed
- Squared coefficient of variation, C^2
- Poisson Process

Prize-winning messages ☺

Throughput is very different for open vs. closed systems



An Exponential distribution is the time to get a single "head."

A Poisson process is a sequence of "heads."

Heavy-tailed, Pareto distributions:
* represent real workloads
* very high variability & DFR
* top 1% comprise half the load

Variance in job sizes is key.
C^2: measure of variance.

Outline

I. Basic Vocabulary

- Avg arrival rate, λ
- Avg service rate, μ
- Avg load, ρ
- Avg throughput, X
- Open vs. closed systems
- Response time, T
- Waiting time, T_Q
- Exponential vs. Pareto/Heavy-tailed
- Squared coefficient of variation, C^2
- Poisson Process

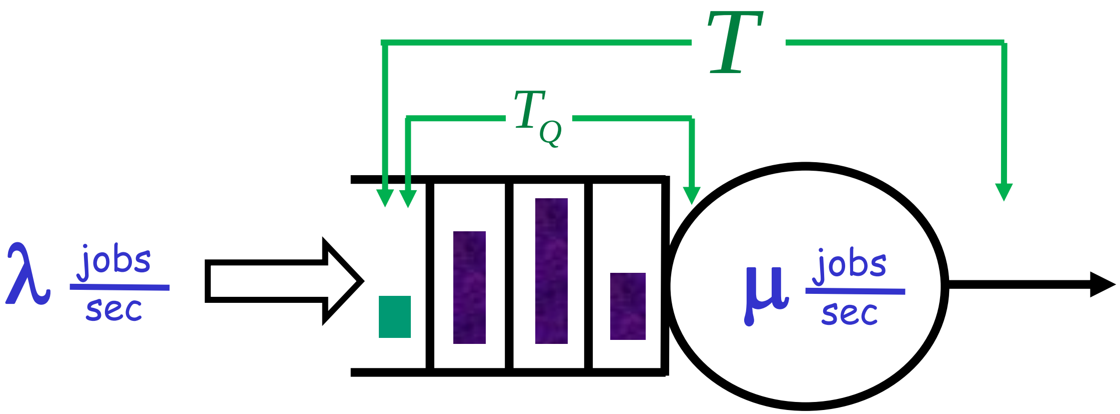
II. Single-server queues

- D/D/1, M/M/1, M/G/1
- Inspection Paradox
- Effect of job size variaibility
- Effect of load
- Provisioning bathrooms/scaling
- Scheduling: FCFS, PS, SJF, LAS, SRPT
- Web server scheduling implementation
- Open vs. closed systems: wait
- Open vs. closed systems: scheduling

III. Multi-server queues

- Static load balancing
- Throwing away servers
- M/M/k + Comparing architectures
- Many slow servers vs. 1 fast
- Capacity provisioning & scaling
- Square root staffing
- Dynamic power management
- Dynamic load balancing/FCFS servers
- Replication
- Dynamic load balancing/PS servers

Single-Server Queue



S : job size

$$E[S] = \frac{1}{\mu}$$

$$\rho = \lambda E[S] = \frac{\lambda}{\mu}$$

D/D/1

Deterministic service times

M/M/1

Exponential inter-arrival times

Exponential service times

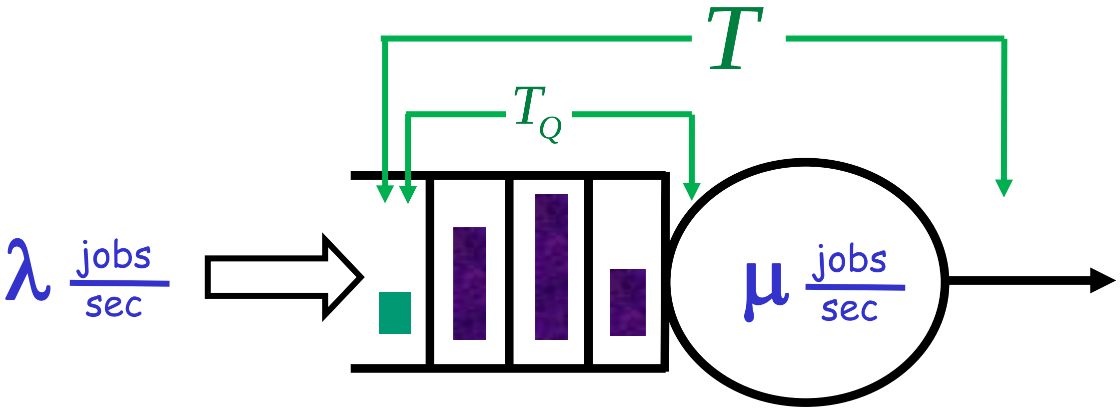
1 server

M/G/1

General service times

M="memoryless"="Markovian"

Single-Server Queue



S : job size

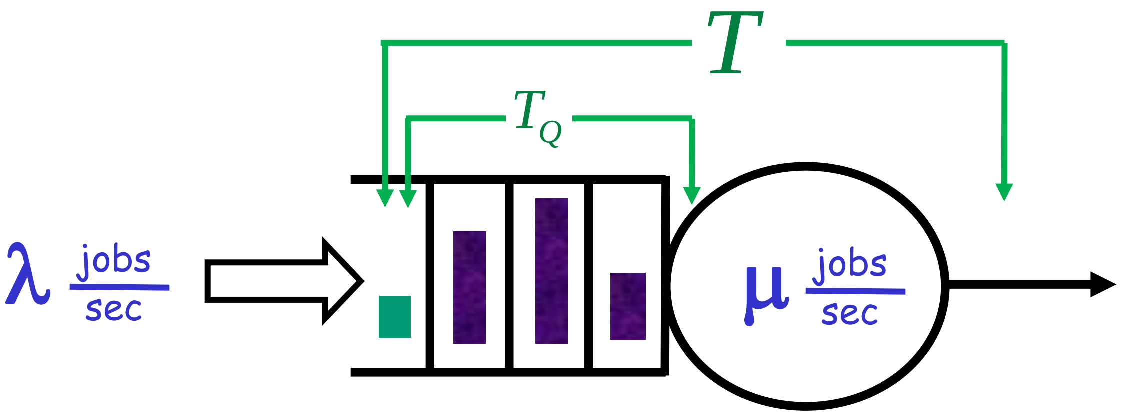
$$E[S] = \frac{1}{\mu}$$

$$\rho = \lambda E[S] = \frac{\lambda}{\mu}$$

D/D/1 **M/M/1** **M/G/1**

Q: Does low ρ ➔ low $E[T_Q]$?

Single-Server Queue



S : job size

$$E[S]=\frac{1}{\mu}$$

$$\rho=\lambda E[S]=\frac{\lambda}{\mu}$$

D/D/1

$$E[T_Q]=0$$

M/M/1

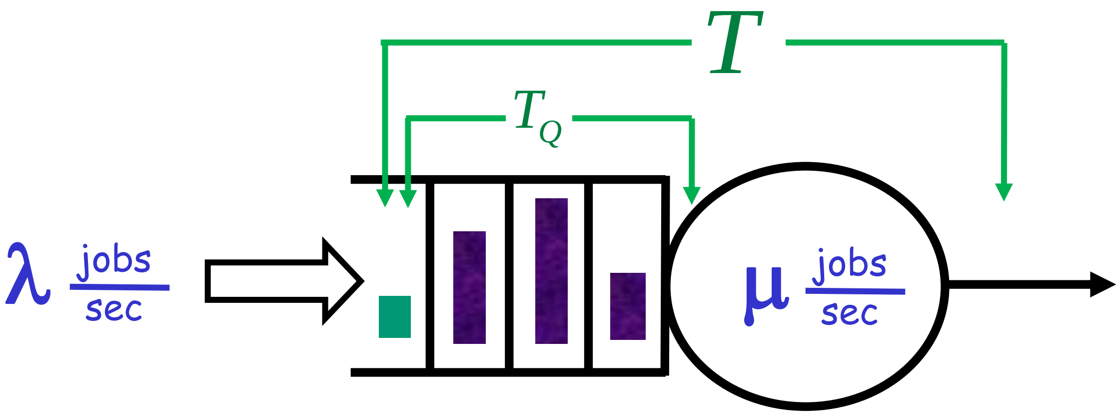
$$E[T_Q]=\frac{\rho}{1-\rho}\cdot E[S]$$

M/G/1

$$E[T_Q]=\frac{\rho}{1-\rho}\cdot\frac{E[S^2]}{2E[S]}$$

related to C^2: variability job size

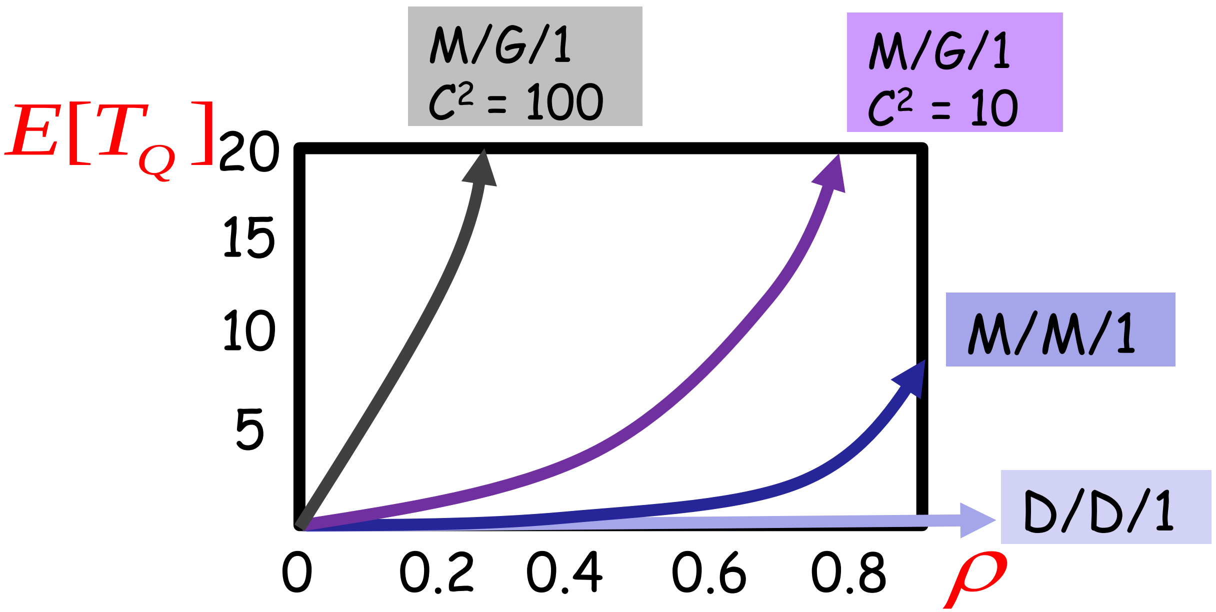
Single-Server Queue



S : job size

$$E[S] = \frac{1}{\mu}$$

$$\rho = \lambda E[S] = \frac{\lambda}{\mu}$$



low load
does NOT imply
low wait

M/G/1

$$E[T_Q] = \frac{\rho}{1-\rho} \cdot \frac{E[S^2]}{2E[S]}$$

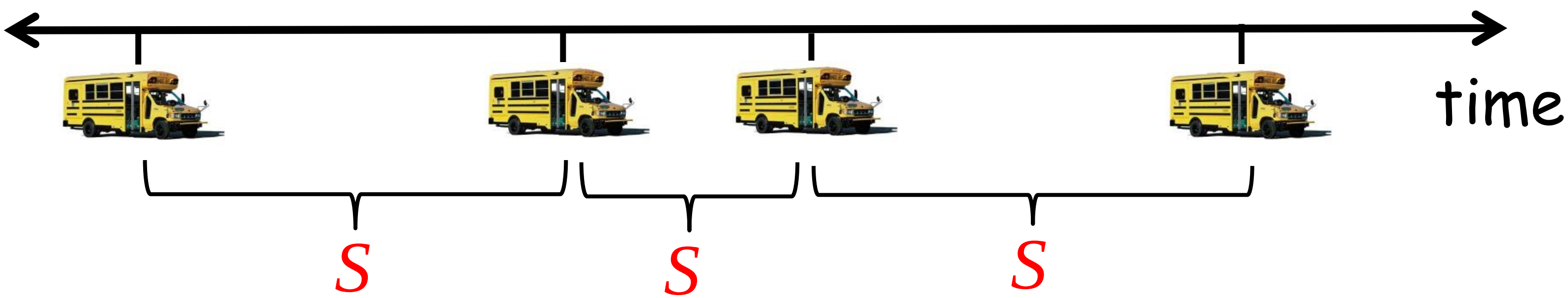
Where is this
coming from?

Waiting for the bus

Waiting for the bus

S: time between buses

$$E[S] = 10\,\text{min}$$



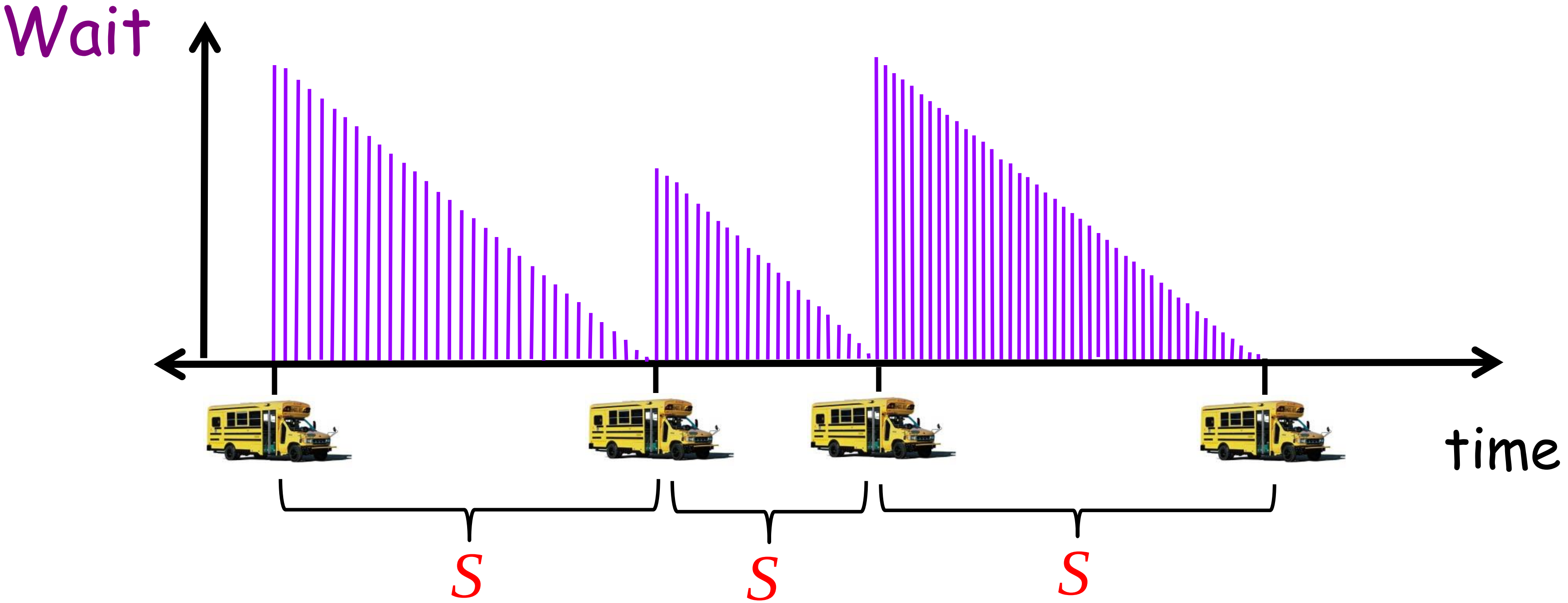
QUESTION:

On average, how long do I have to wait for a bus?

(a) < 5 min
(b) 5 min
(c) 10 min
(d) >10 min

Waiting for the bus

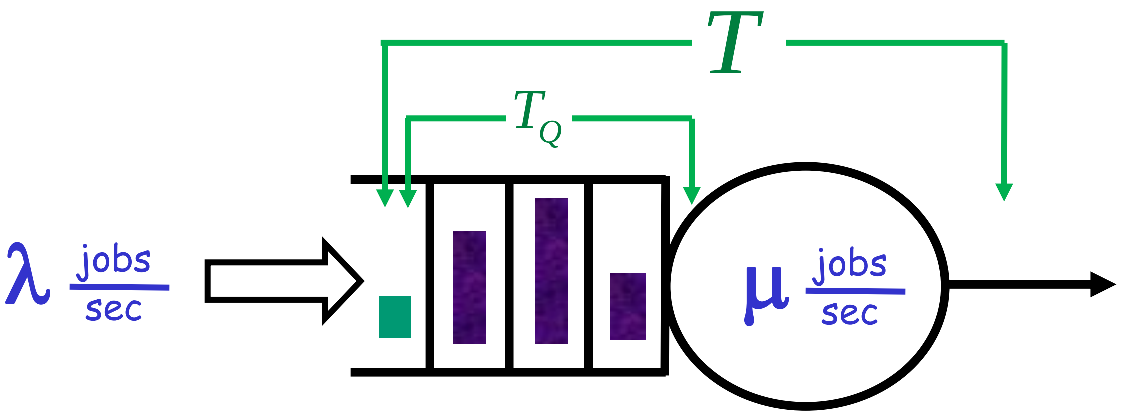
S: time between buses



$$E[\text{Wait}] = \frac{E[S^2]}{2E[S]} >> E[S]$$

"Inspection Paradox"

Back to Single-Server Queue



S : job size

$$E[S] = \frac{1}{\mu}$$

$$\rho = \lambda E[S] = \frac{\lambda}{\mu}$$

$$E[T_Q]^{M/G/1} = \frac{\rho}{1-\rho} \cdot \frac{E[S^2]}{2E[S]}$$

$$\text{Low } \rho \quad \not\Rightarrow \quad \text{Low } E[T_Q]$$

Waiting for the Loo

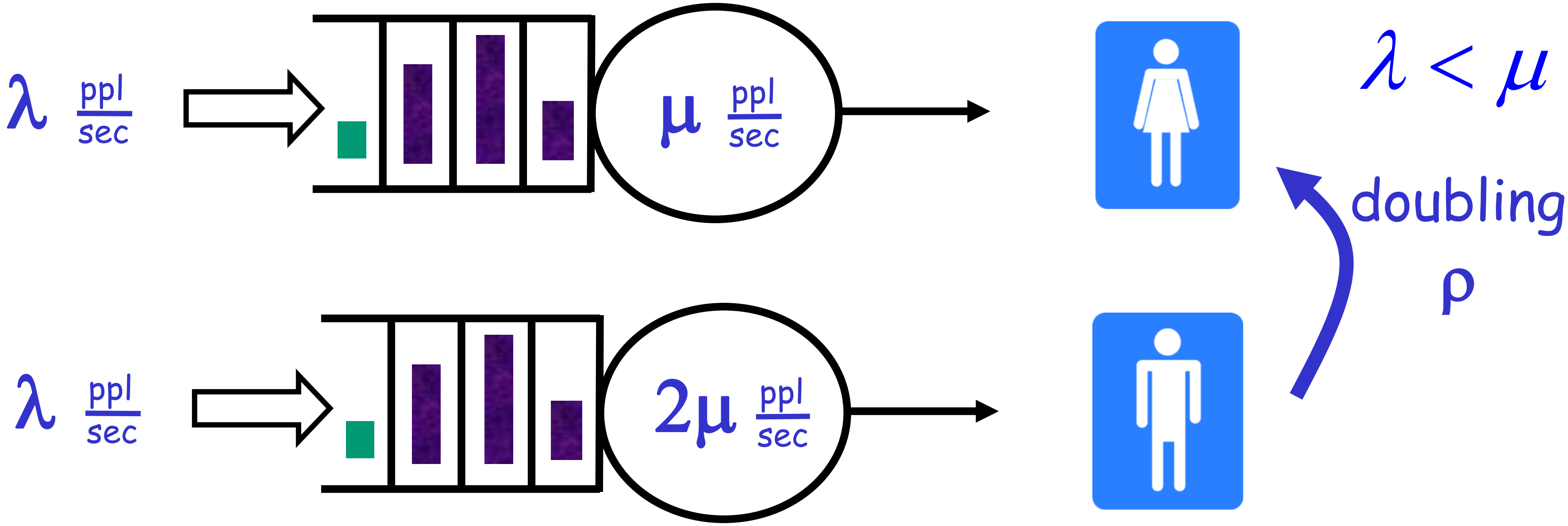
Check out the line for the men's room …

Waiting for the Loo

- On avg, Women spend 88 sec in loo.
- On avg, Men spend 40 sec in loo.

Waiting for the Loo

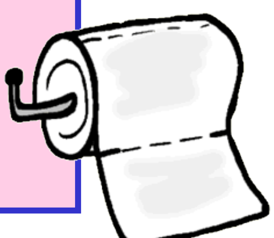
M/M/1 model



QUESTION:

Women take 2X as long. What's the difference in their wait?

(a) factor < 2
(b) factor 2
(c) factor 4
(d) factor > 4



Waiting for the Loo

M/M/1

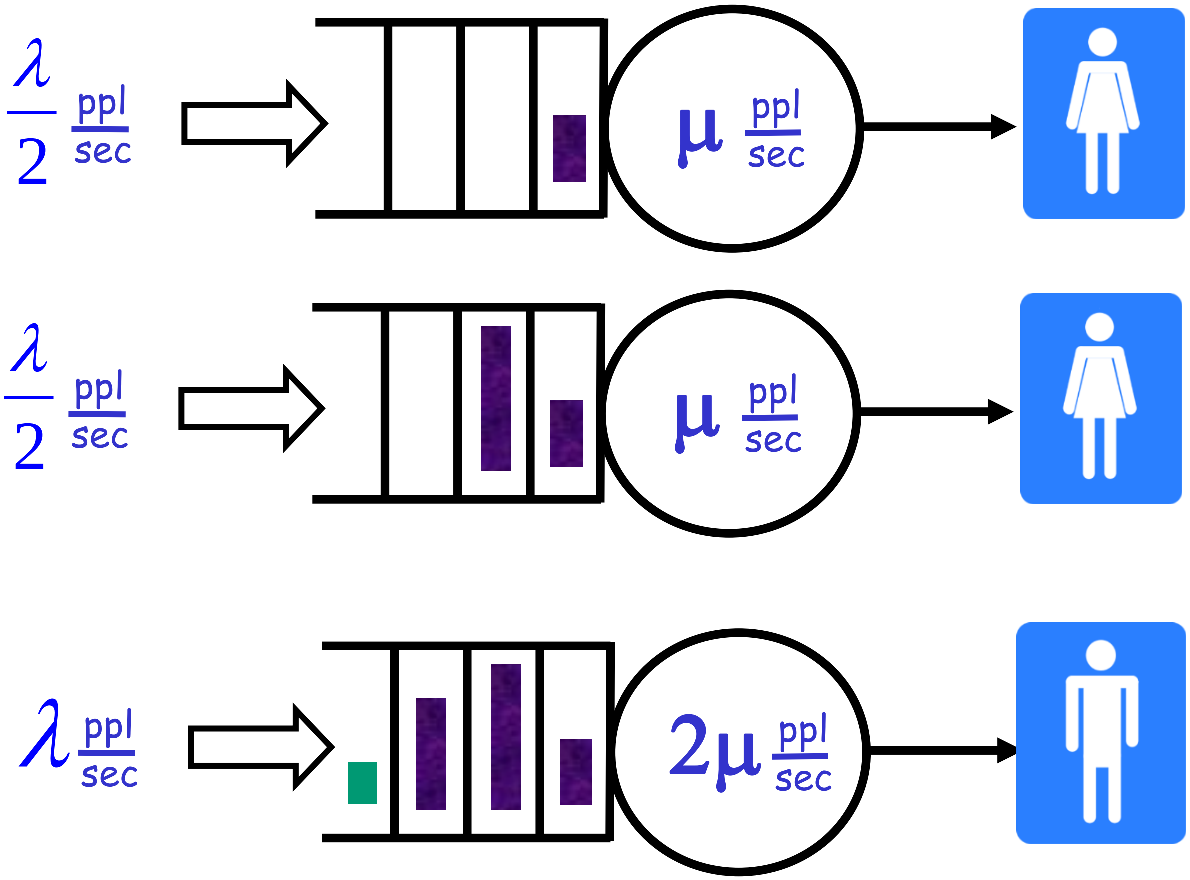
$$E[T_Q] = \frac{\rho}{1-\rho} \cdot E[S]$$

M/G/1

$$E[T_Q] = \frac{\rho}{1-\rho} \cdot \frac{E[S^2]}{2E[S]}$$

Doubling ρ can increase $\mathbf{E[T_Q]}$
by factor of 4 to ∞

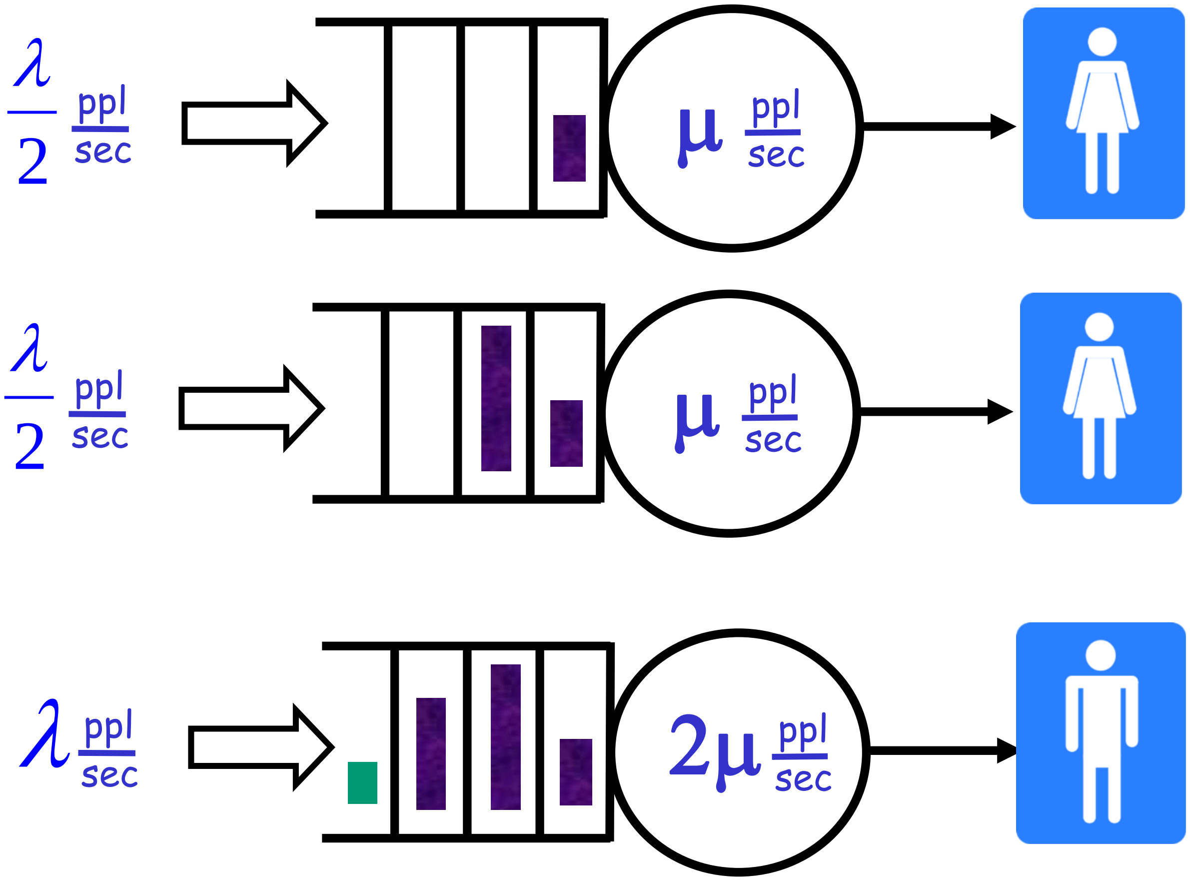
Equalizing the wait for men & women



2 Women's rooms for each Men's room.

QUESTION:
Is this (a) insufficient (b) overkill (c) just right

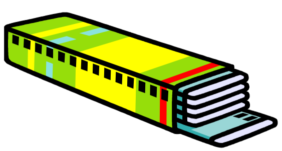
Equalizing the wait for men & women



Insufficient!

Waiting time for women is still factor of 2 higher.

$$E[T_Q] = \frac{\rho}{1-\rho} \cdot E[S]$$



Also true under M/G/1 model.

For what models is this not true?

M/G/1

$$E[T_Q] = \frac{\rho}{1-\rho} \cdot \frac{E[S^2]}{2E[S]}$$

High load leads to high wait

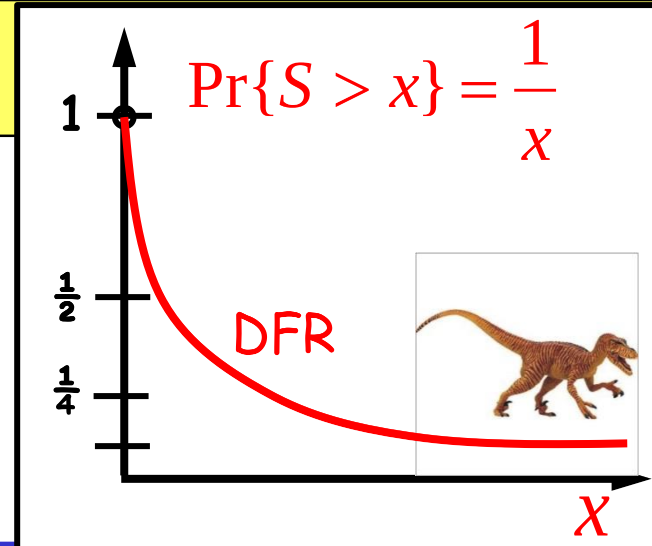
High job size variability leads to high wait

To drop load, we can increase server speed.

Q: What can we do to combat job size variability?

A: Smarter scheduling!

Scheduling in M/G/1



QUESTION:
Which scheduling policy is best for minimizing $\mathrm{E}[\mathrm{T}]$?

FCFS (First-Come-First-Served, non-preemptive)

PS (Processor-Sharing, preemptive)

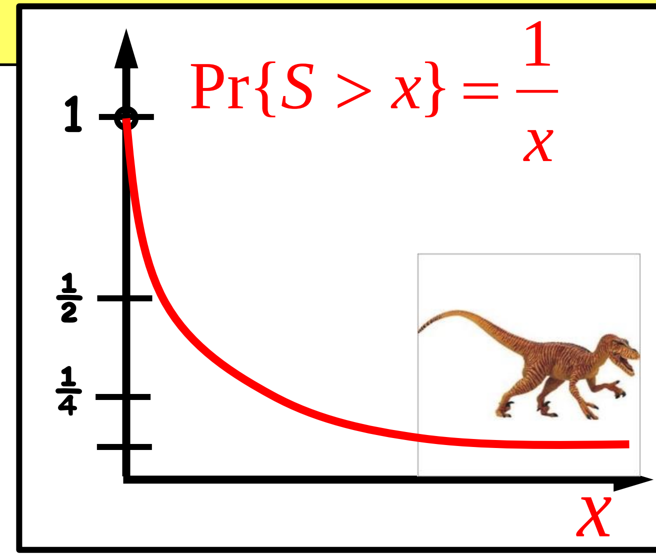
SJF (Shortest-Job-First, non-preemptive)

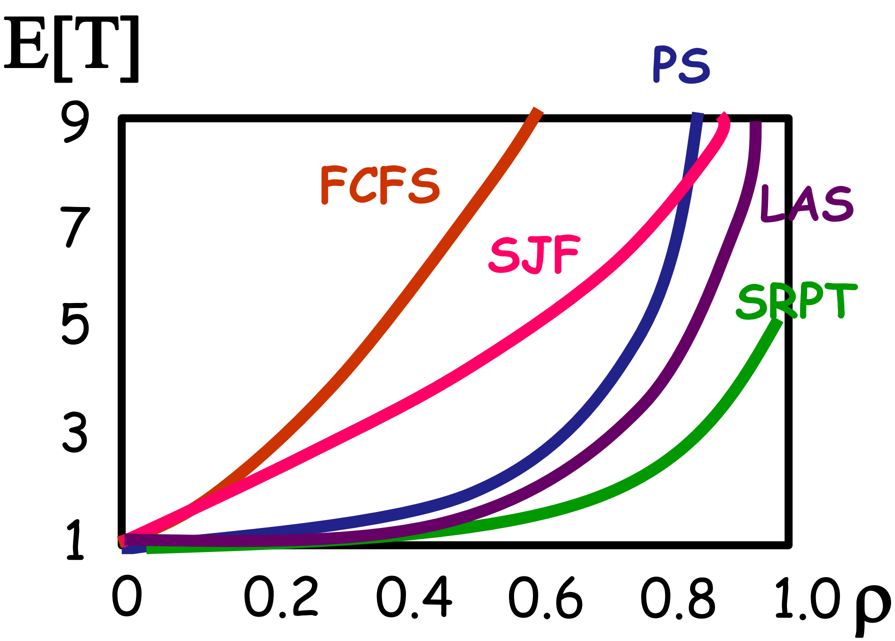
SRPT (Shortest-Remaining-Processing-Time, preemptive)

LAS (Least-Attained-Service First, preemptive)

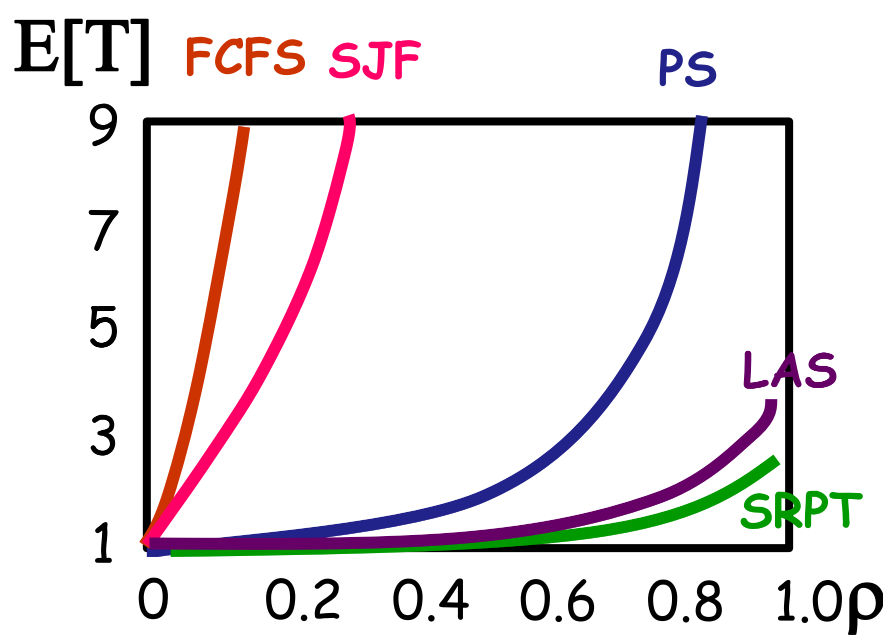
[Harchol-Balter EORMS 2011]

Scheduling in M/G/1



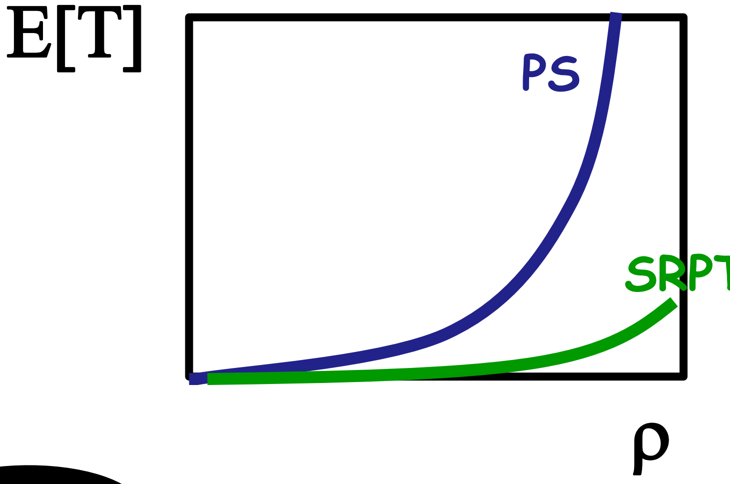


$C^2=10$



$C^2=100$

Scheduling in M/G/1

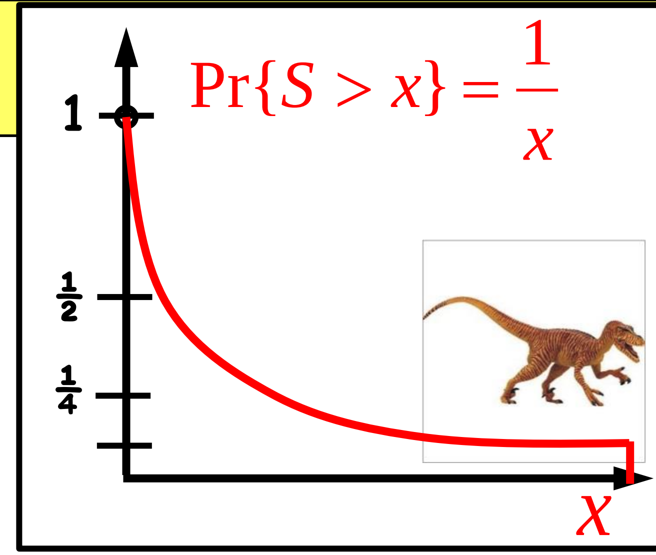


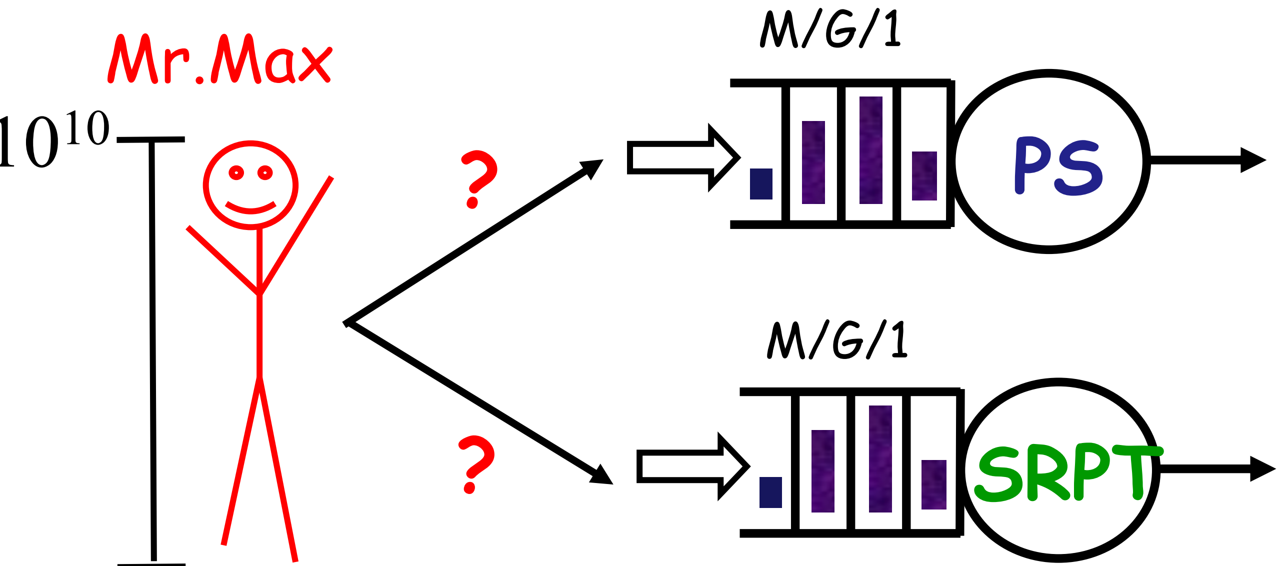
We saw: $E[T]^{\text{SRPT}} \ll E[T]^{\text{PS}}$

Unfairness Question

Let $S \sim$ Bounded Pareto with $\max = 10^{10}$

Let $\rho = 0.9$

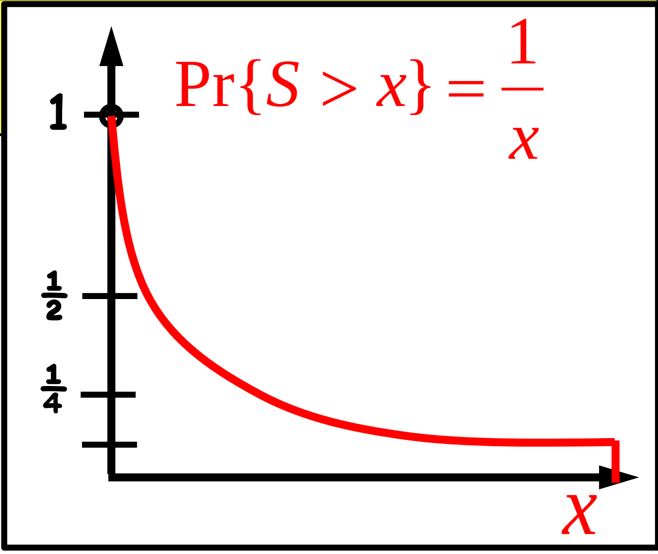




QUESTION:

Which queue does Mr. Max prefer?

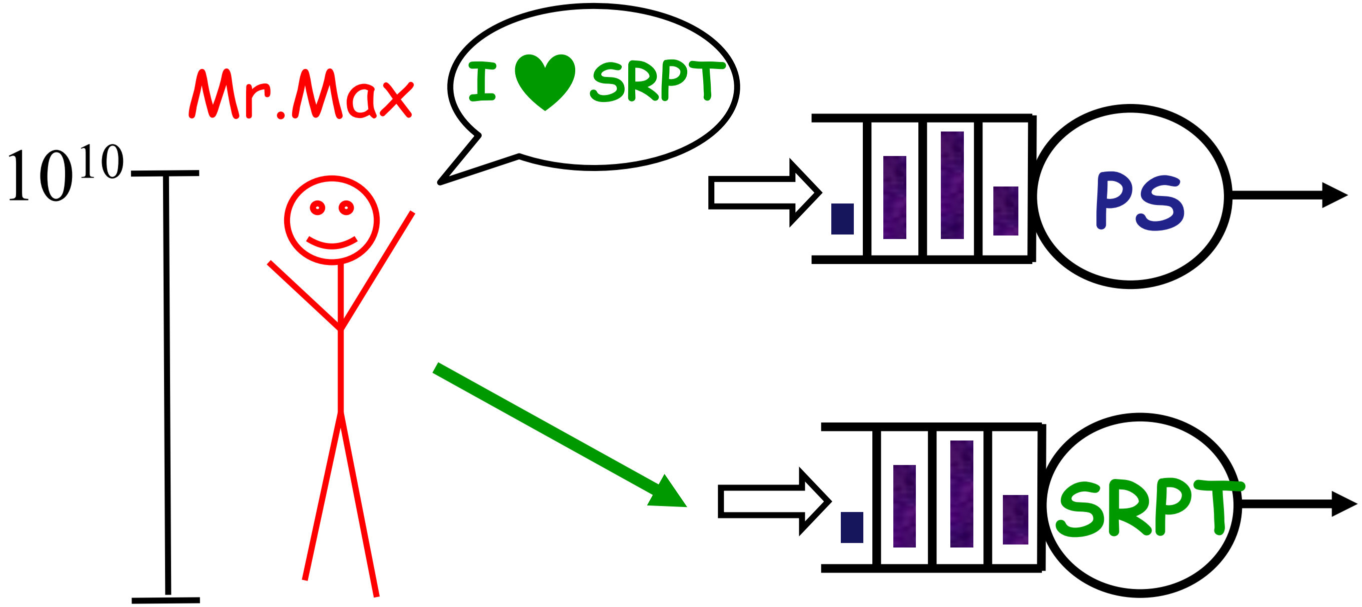
Unfairness Question



Let S ~ Bounded Pareto ($\alpha = 1.1$)

with $\max = 10^{10}$

Let $\rho = 0.9$



Unfairness Question

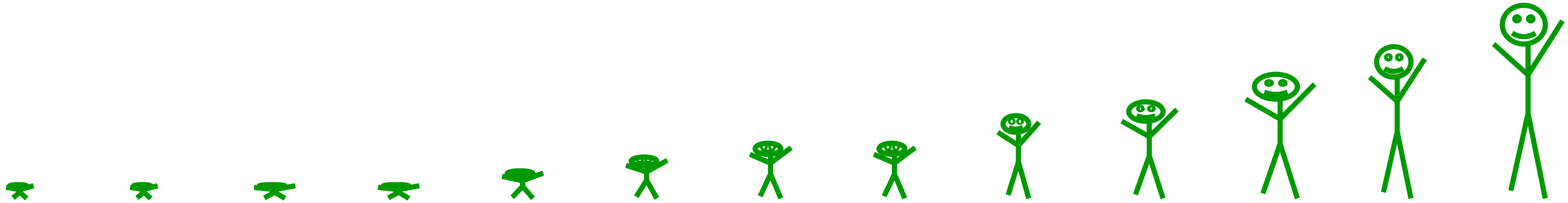
Defies Kleinrock's Conservation Law

__All-can-win-theorem__:
[Bansal, Harchol-Balter, Sigmetrics 2001]

Under M/G/1, for all job size distributions, if $\rho < 0.5$,

$$E[T(x)]^{\text{SRPT}} < E[T(x)]^{\text{PS}} \quad \text{for all job size } x.$$

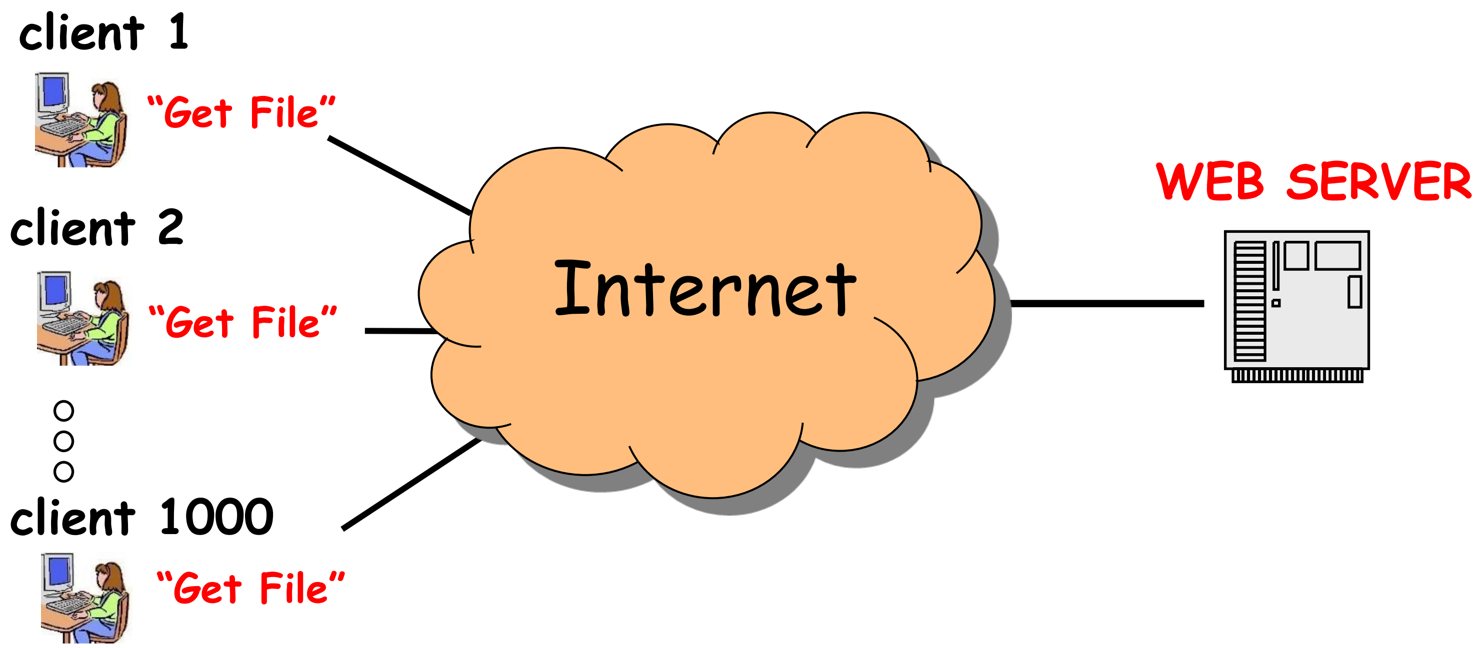
For heavy-tailed distributions, holds for $\rho < 0.95$.



Scheduling in the Real World

Traditional web servers use **PS (Fair)** scheduling.

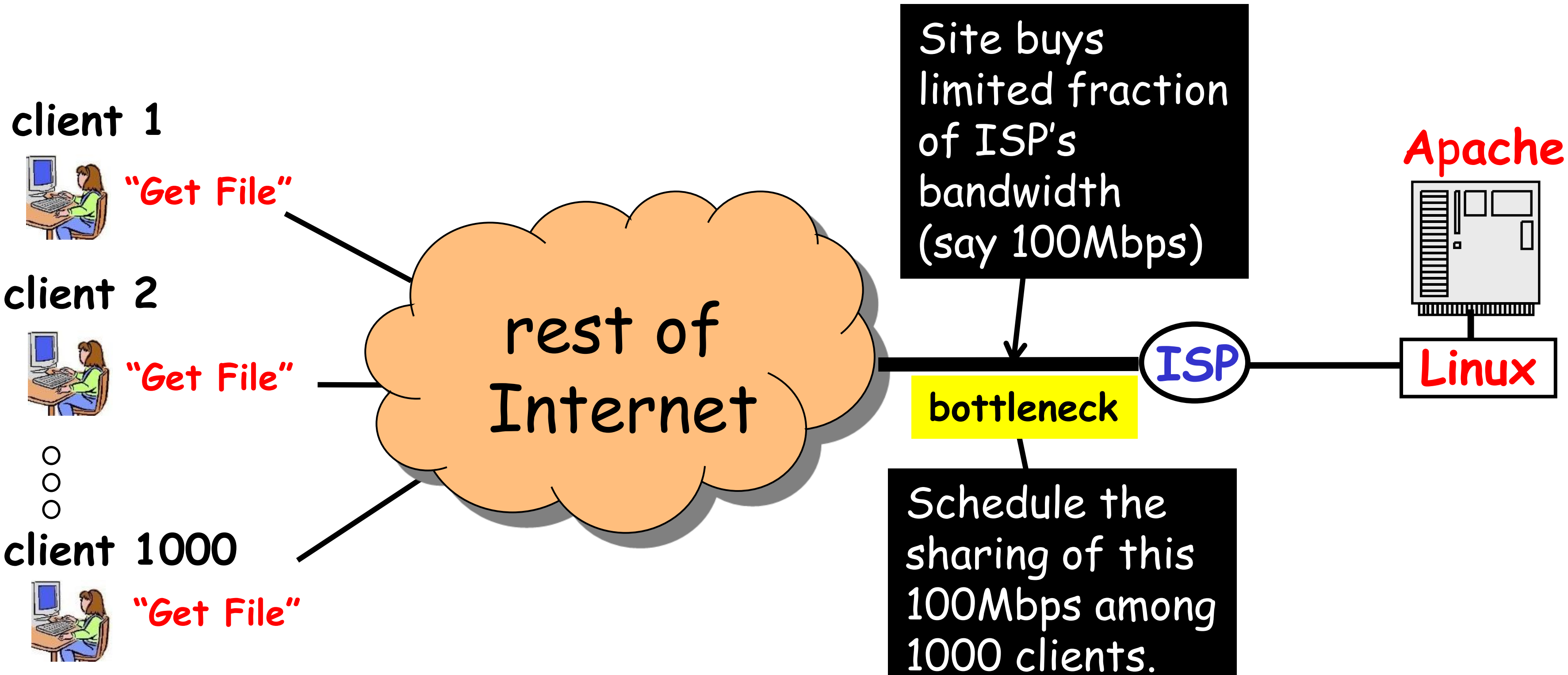
Let's do **SRPT** scheduling instead! [Harchol-Balter et al. TOCS 2003]



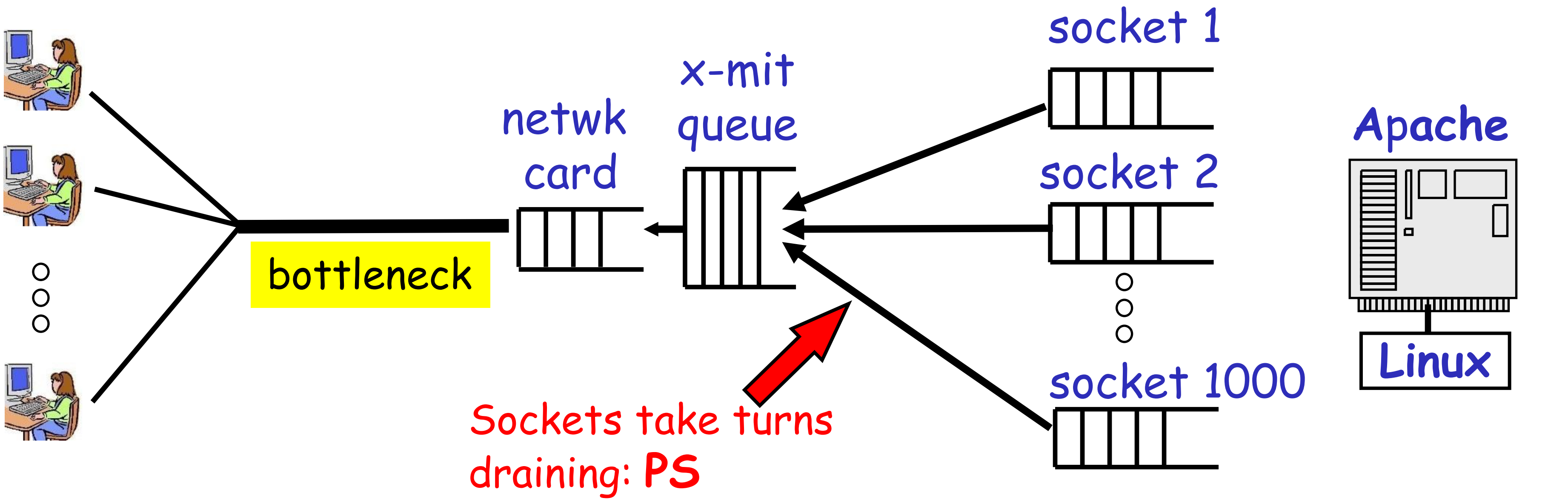
Q: What is being scheduled?
Q: How is size used?

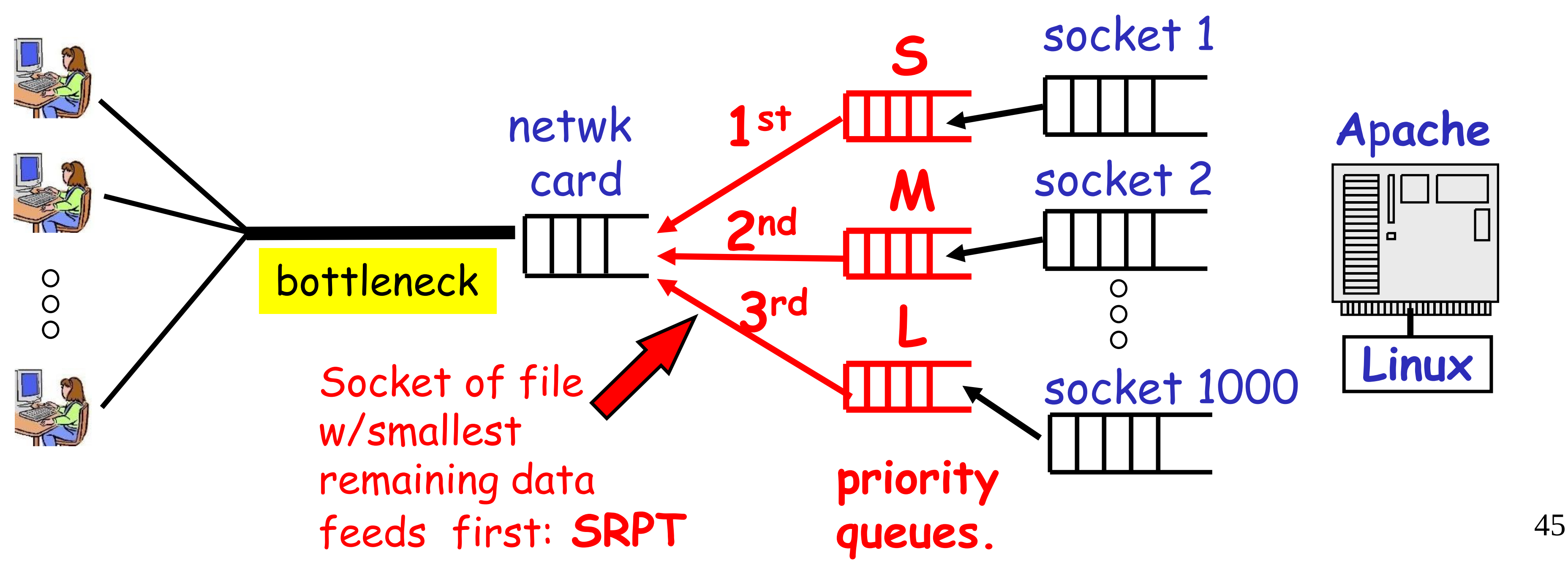
SRPT Scheduling for Web Servers

Q: What is being scheduled?
A: Bottleneck device is limited ISP bandwidth.
Q: How is size being used?
A: S = Size of requests = Size of file ~ Pareto(α = 1)

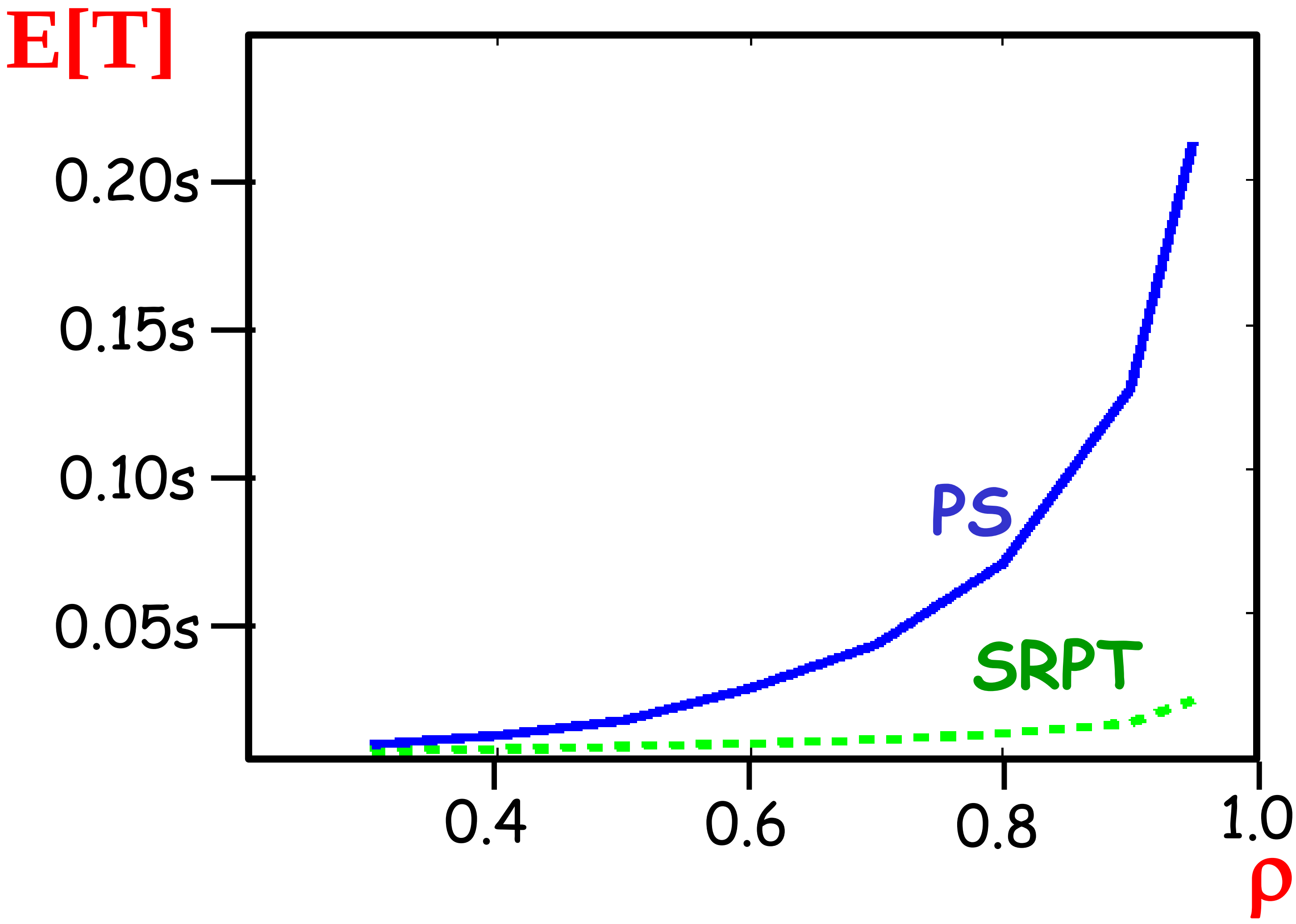


Linux Implementation

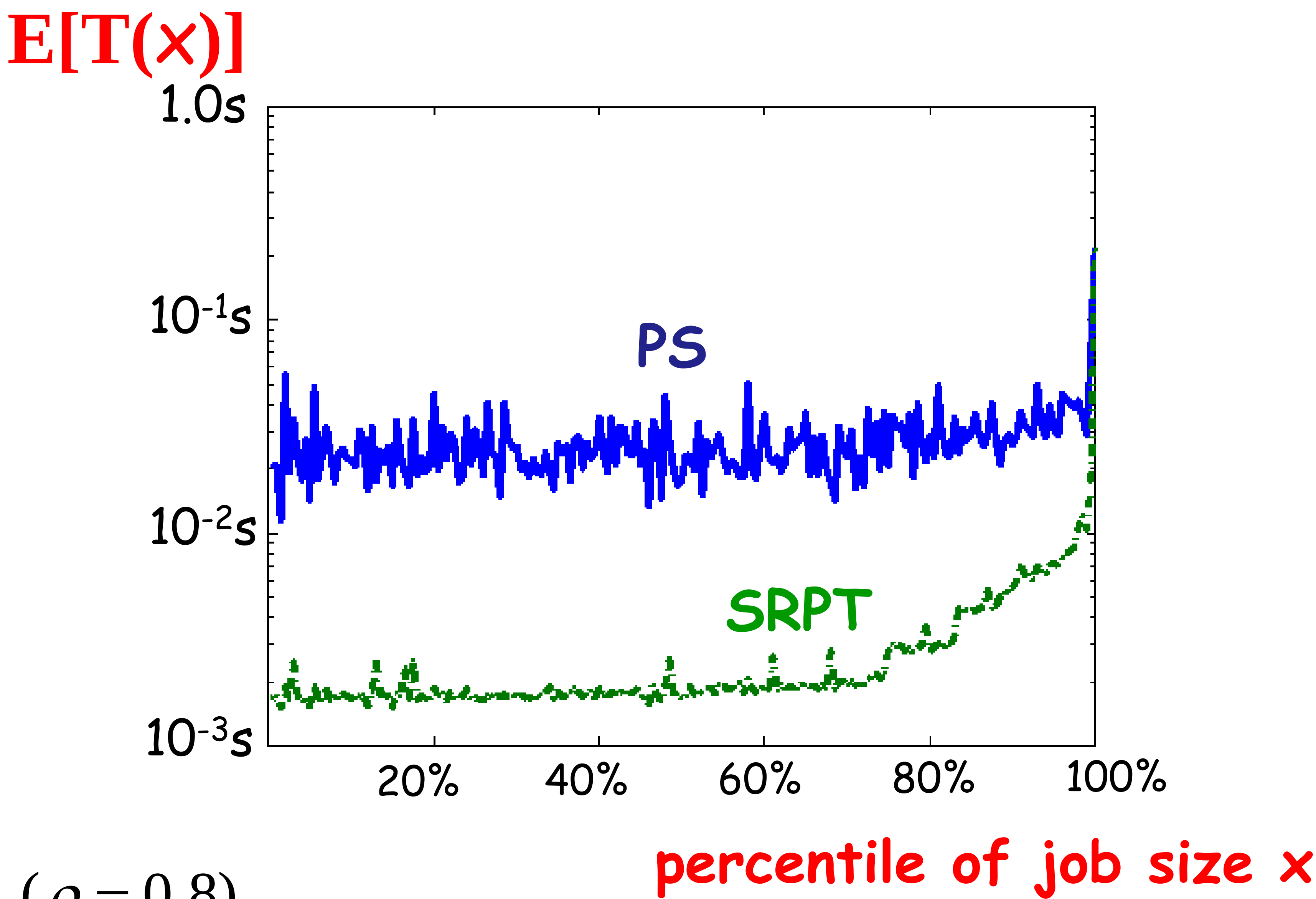




Mean response time results



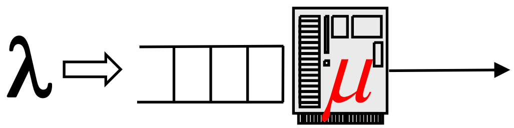
Response time as fcn of Size



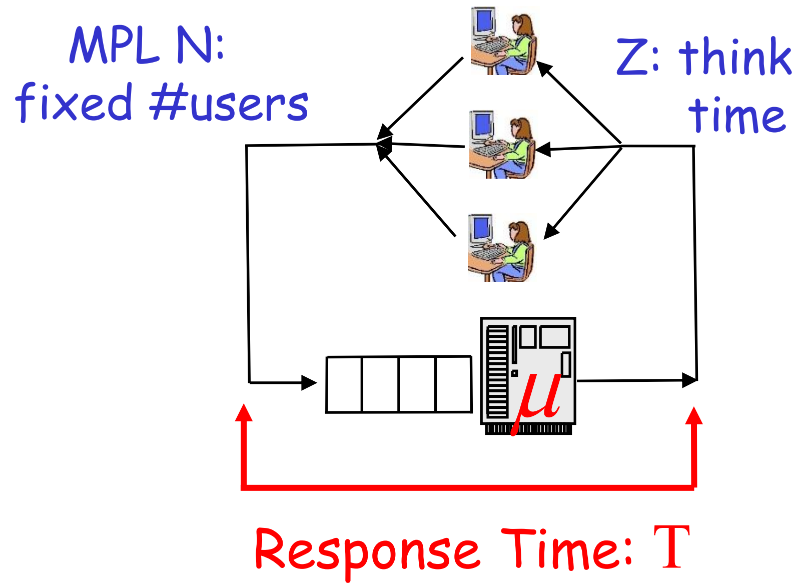
$(\rho = 0.8)$

Caution: Open versus Closed

Open



Closed

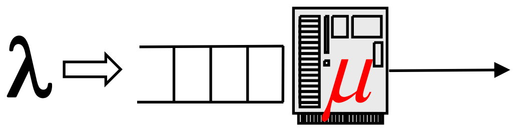


QUESTION: When run with same load ρ, which has higher $E[T]$?

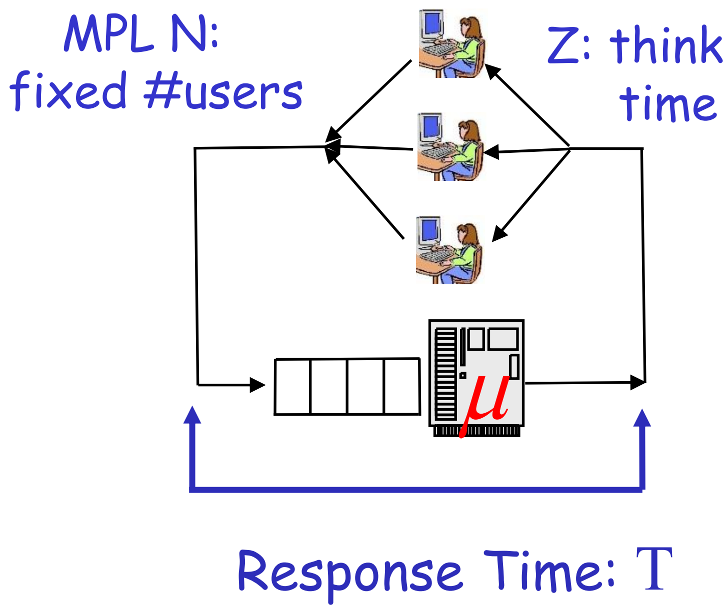
(a) Open
(b) Closed
(c) Same

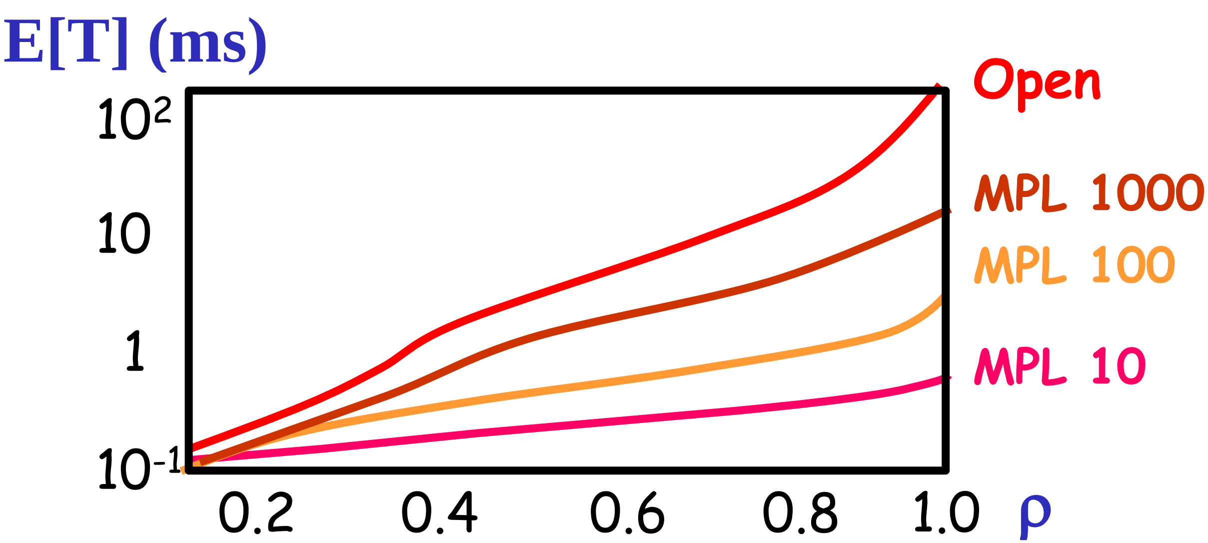
Caution: Open versus Closed

Open



Closed



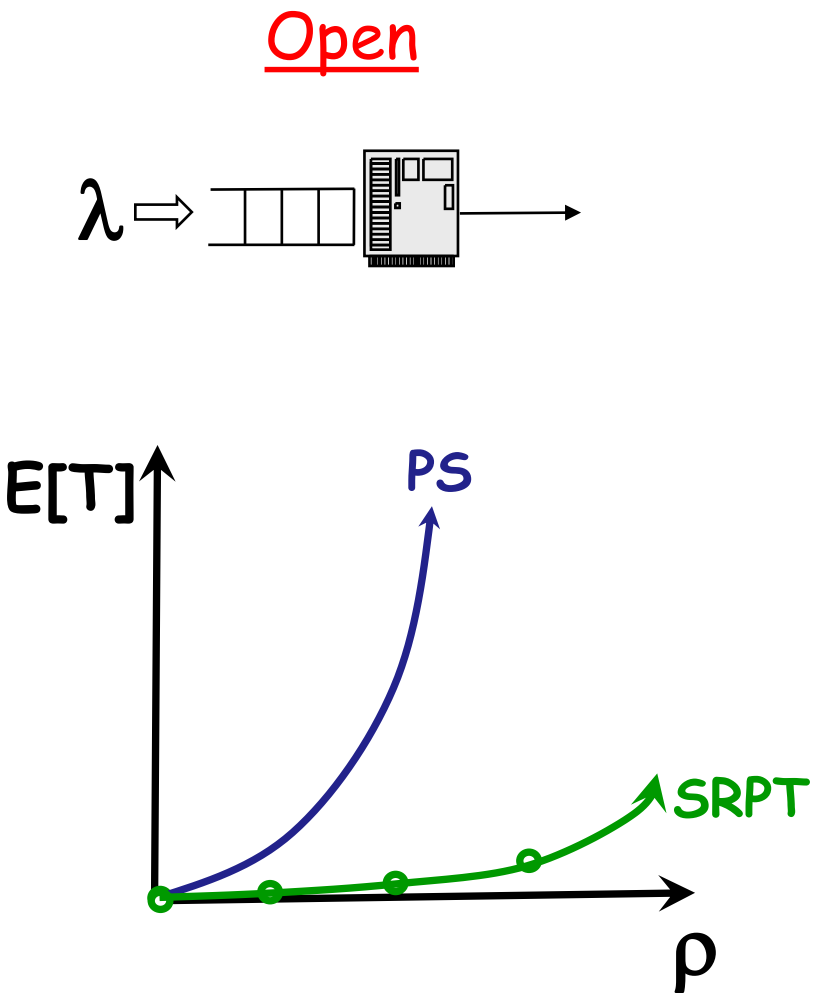


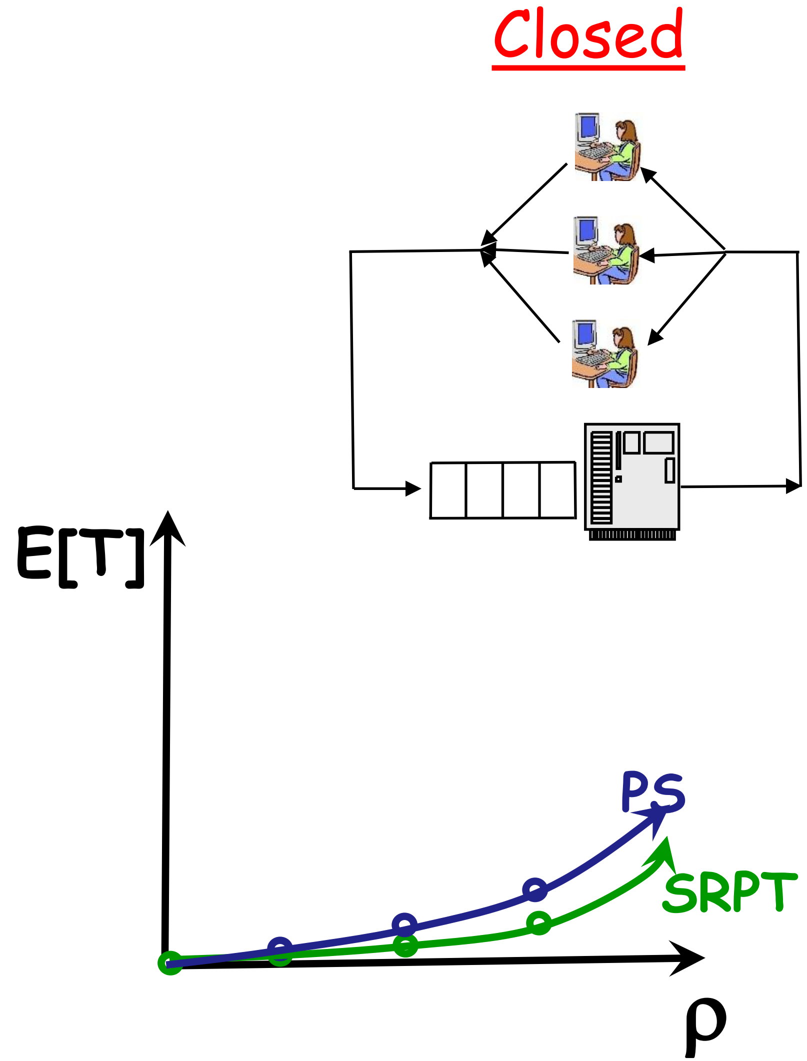
Performance of Auction Site

[Schroeder, Wierman, Harchol-Balter NSDI 2006]

E[T] much lower for closed system w/ same ρ

Caution: Open versus Closed





Closed & open systems run w/ same job size distribution and same load.

[Schroeder, Wierman, Harchol-Balter, NSDI 06]

Summary Part II

II. Single-server queues

- D/D/1, M/M/1, M/G/1
- Inspection Paradox
- Effect of job size variaibility
- Effect of load
- Provisioning bathrooms/scaling
- Scheduling: FCFS, PS, SJF, LAS, SRPT
- Web server scheduling implementation
- Open vs. closed systems: wait
- Open vs. closed systems: scheduling

Prize-winning messages ☺

M/G/1:
Low load does NOT always imply low waiting time.

Waiting time has non-linear relationship to load.

Policies that seem unfair may not be.

"Inspection paradox"
Waiting time is affected by variability in job size.

Smart scheduling can combat job size variability.

Closed systems behave very differently from open.

Outline

I. Basic Vocabulary

- Avg arrival rate, λ
- Avg service rate, μ
- Avg load, ρ
- Avg throughput, X
- Open vs. closed systems
- Response time, T
- Waiting time, T_Q
- Exponential vs. Pareto/Heavy-tailed
- Squared coefficient of variation, C^2
- Poisson Process

II. Single-server queues

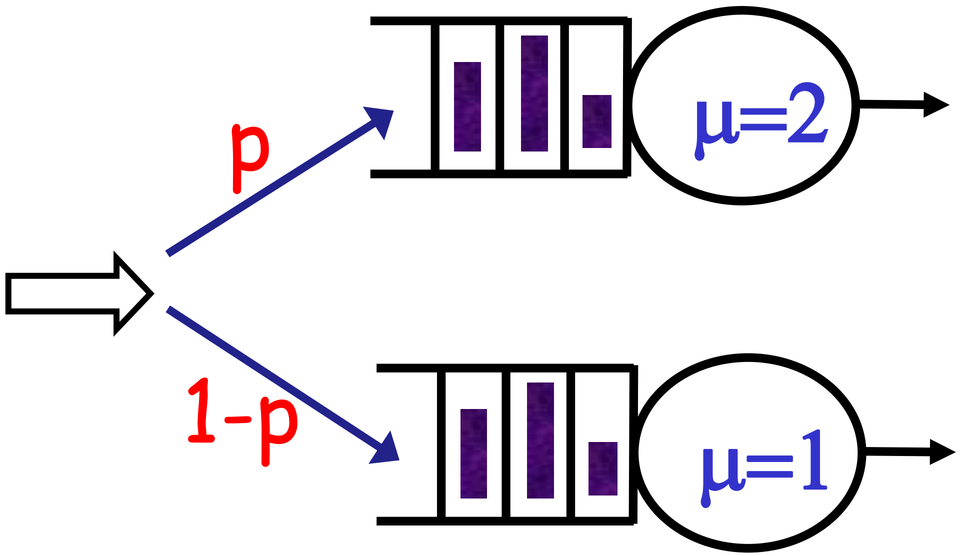
- D/D/1, M/M/1, M/G/1
- Inspection Paradox
- Effect of job size variaibility
- Effect of load
- Provisioning bathrooms/scaling
- Scheduling: FCFS, PS, SJF, LAS, SRPT
- Web server scheduling implementation
- Open vs. closed systems: wait
- Open vs. closed systems: scheduling

III. Multi-server queues

- Static load balancing
- Throwing away servers
- M/M/k + Comparing architectures
- Many slow servers vs. 1 fast
- Capacity provisioning & scaling
- Square root staffing
- Dynamic power management
- Dynamic load balancing/FCFS servers
- Replication
- Dynamic load balancing/PS servers

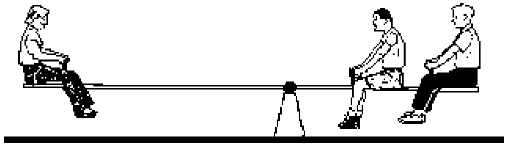
Load Balancing

Poisson process with rate $\lambda \frac{\text{jobs}}{\text{sec}}$

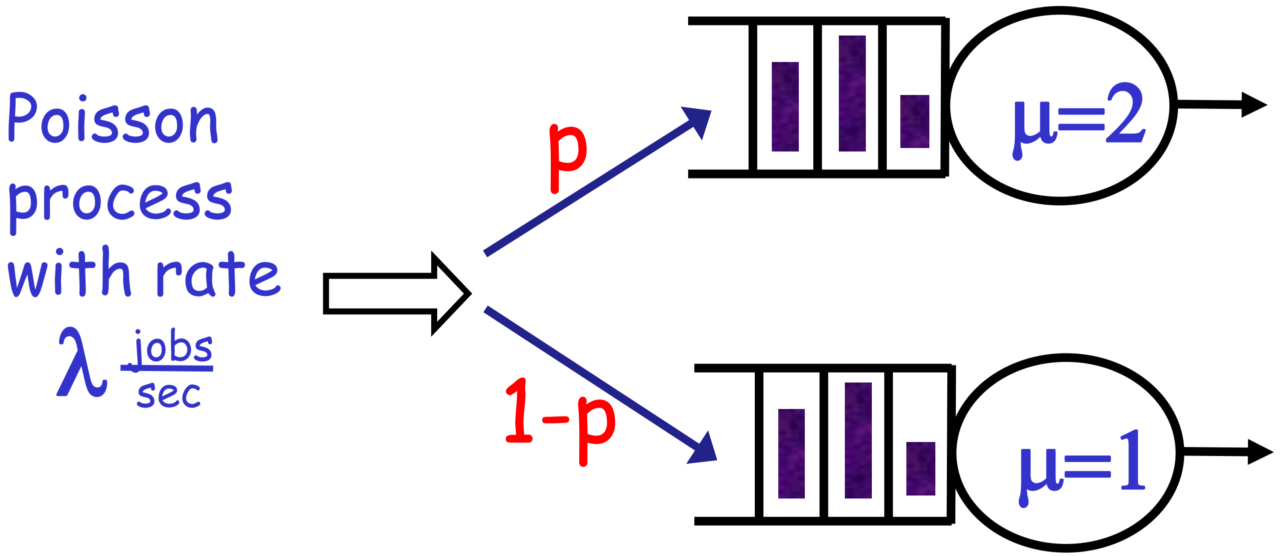


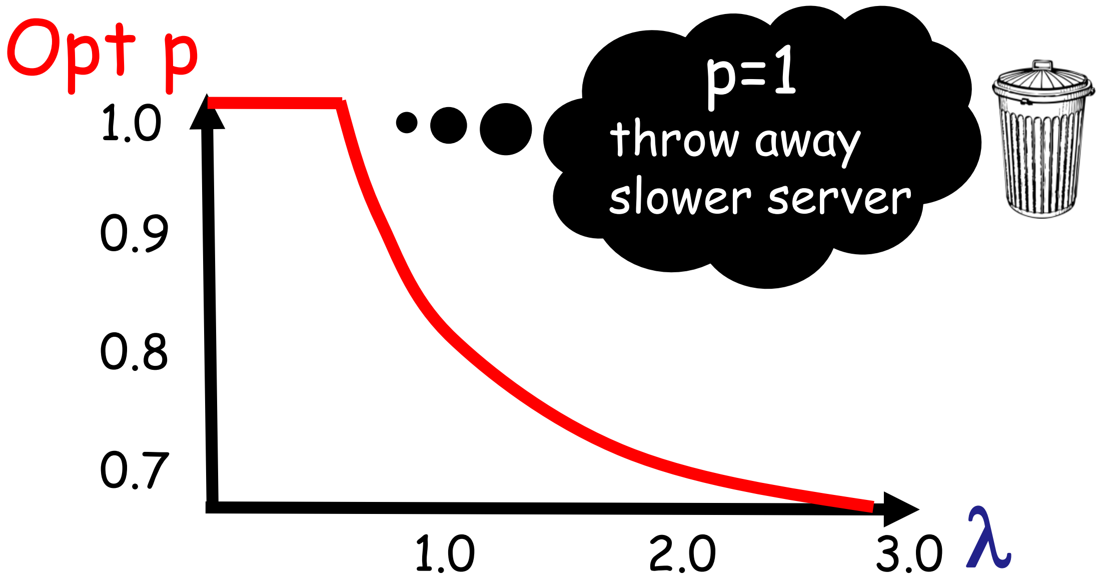
QUESTION: What is the optimal p to minimize $\mathbf{E[T]}$?

(a) $p = 2/3$ (b) $p > 2/3$ (c) $p < 2/3$

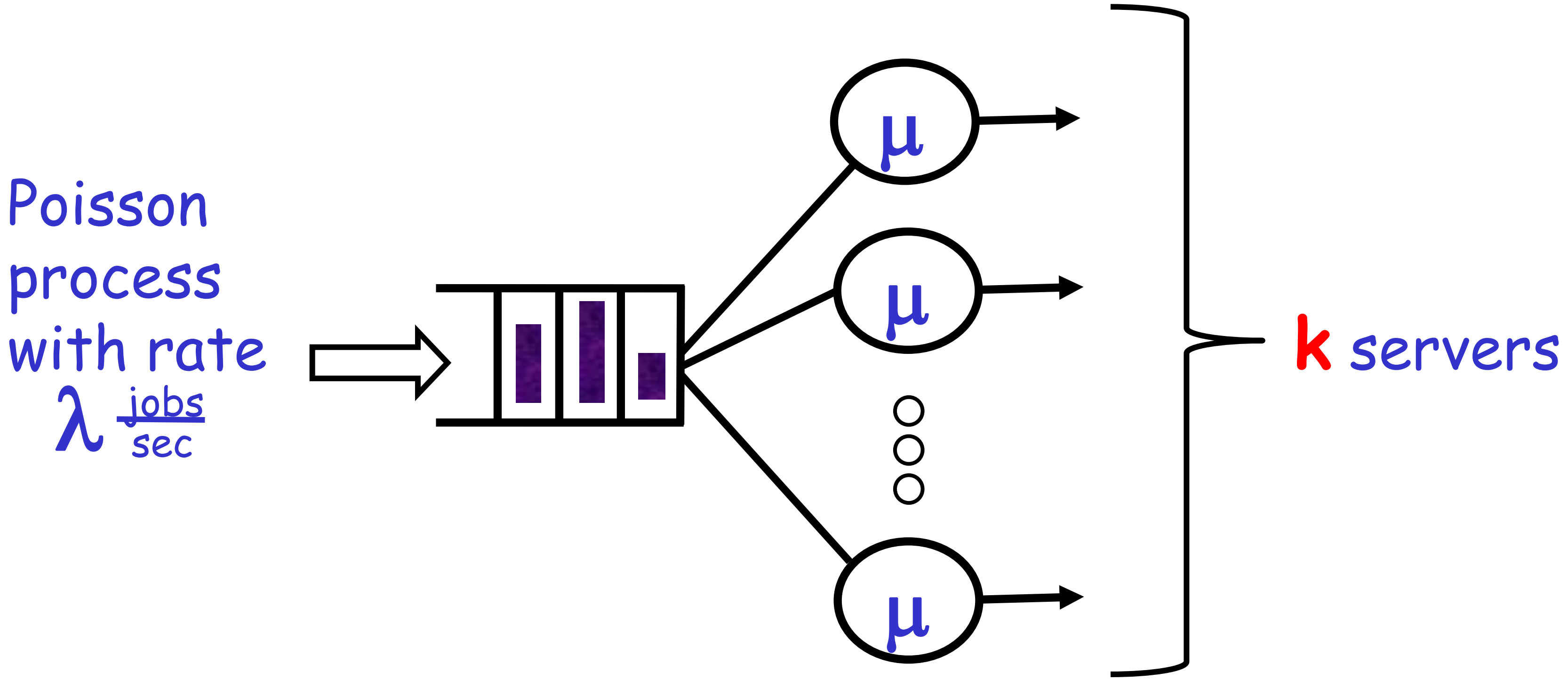


Load Balancing





M/M/k



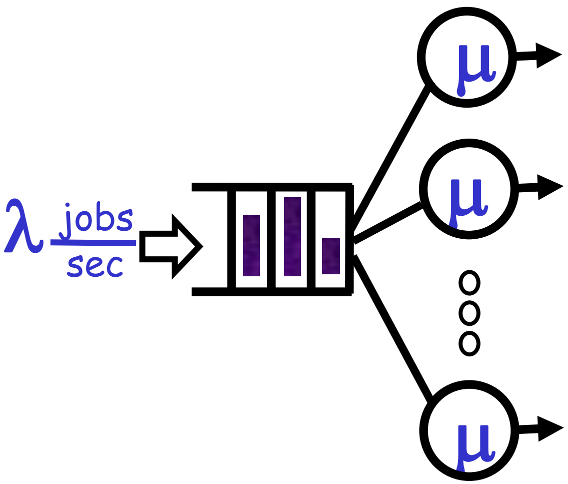
Central queue. Server takes job when free.

Job size $S \sim \mathrm{Exp}(\mu)$

$$\rho \equiv \text{System Load} \equiv \frac{\lambda}{k\mu}$$

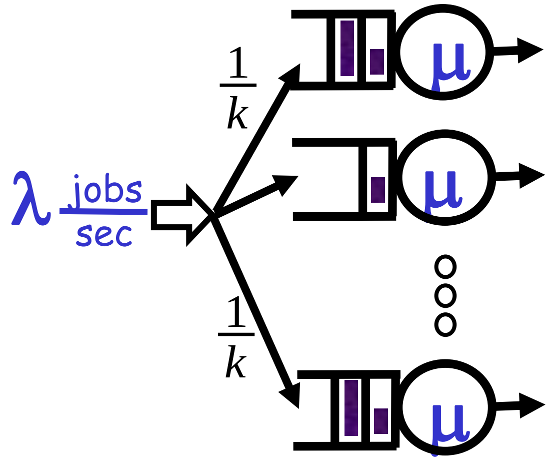
3 Architectures

M/M/k



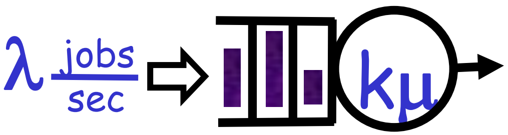
$$\rho = \frac{\lambda}{k\mu}$$

Splitting



$$\rho = \frac{\lambda}{k\mu}$$

M/M/1fast

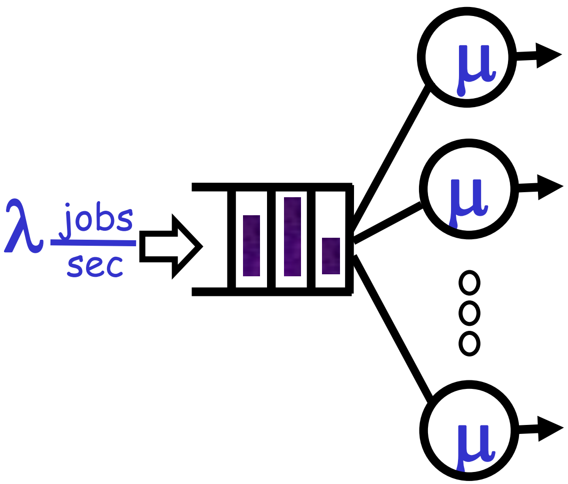


$$\rho = \frac{\lambda}{k\mu}$$

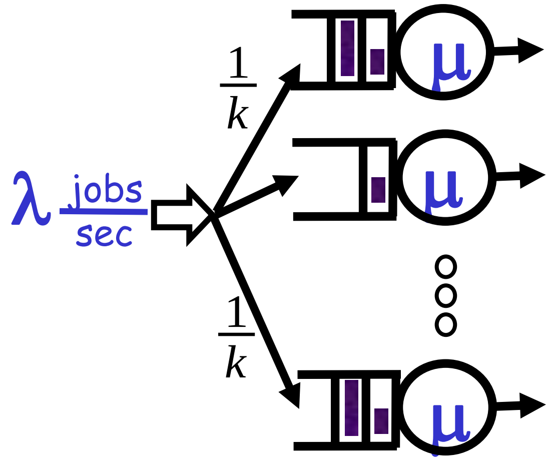
Q: Which is best for minimizing E[T]?

3 Architectures

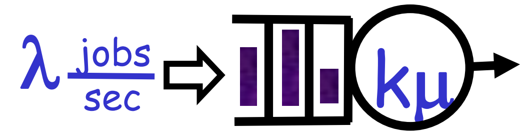
M/M/k

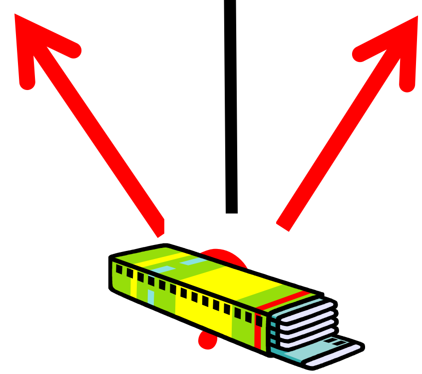


Splitting



M/M/1fast

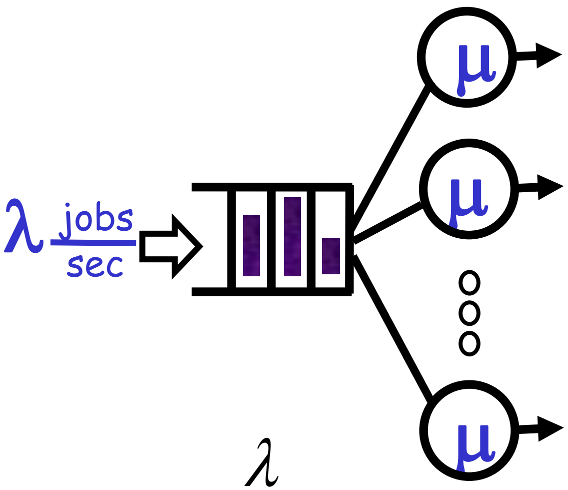




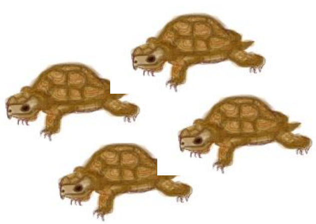
$$E[T_Q]^{M/M/1} = \frac{\rho}{1-\rho} \cdot E[S]$$

Many slow or 1 fast?

M/M/k

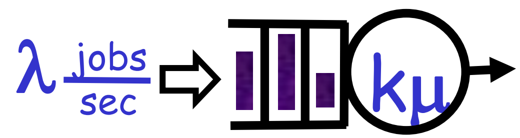


$$\rho = \frac{\lambda}{k\mu}$$



vs.

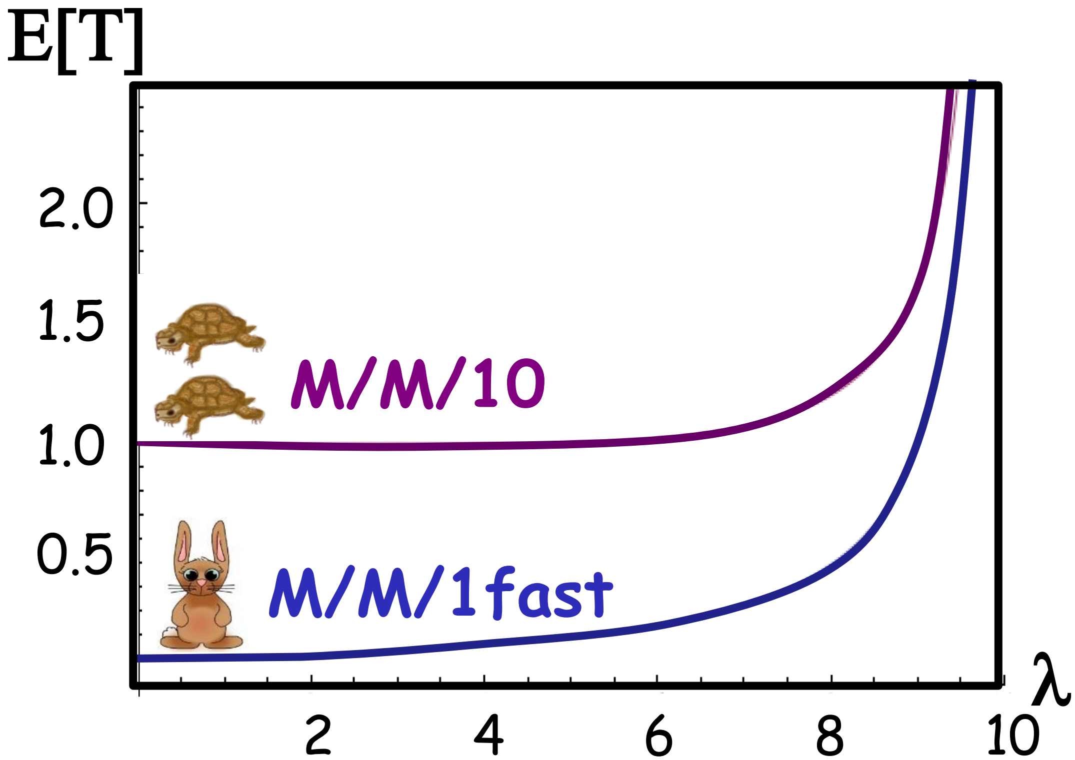
M/M/1fast

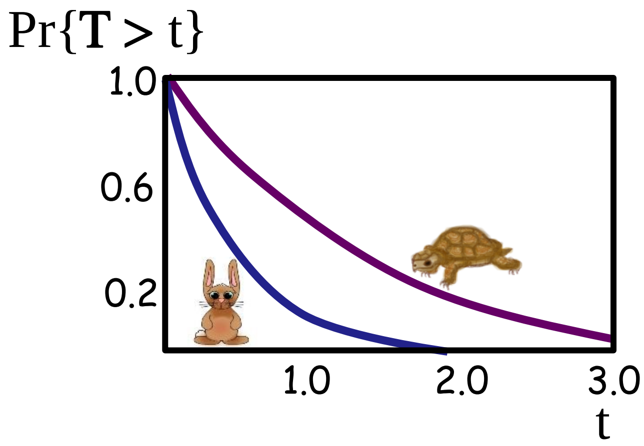


$$\rho = \frac{\lambda}{k\mu}$$

QUESTION: Which is best for minimizing E[T]?

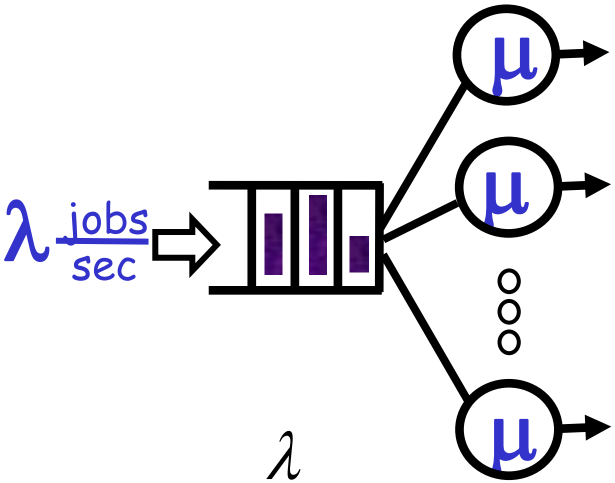
Many slow or 1 fast?



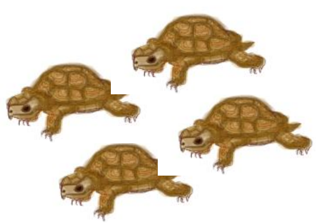


Many slow or 1 fast: Revisited

M/G/k

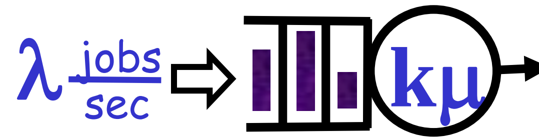


$$\rho = \frac{\lambda}{k\mu}$$



vs.

M/G/1fast

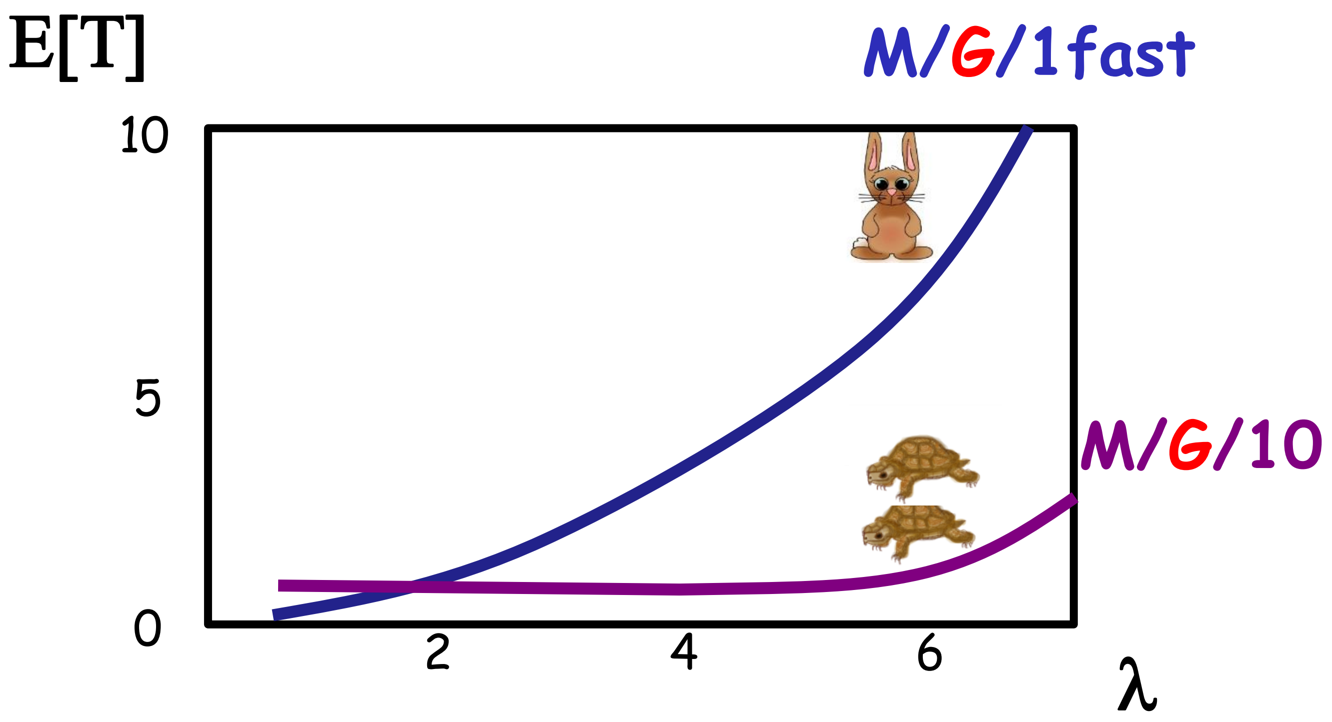


$$\rho = \frac{\lambda}{k\mu}$$

QUESTION: Which is best for minimizing E[T]?

$$C_G^2 = 100$$

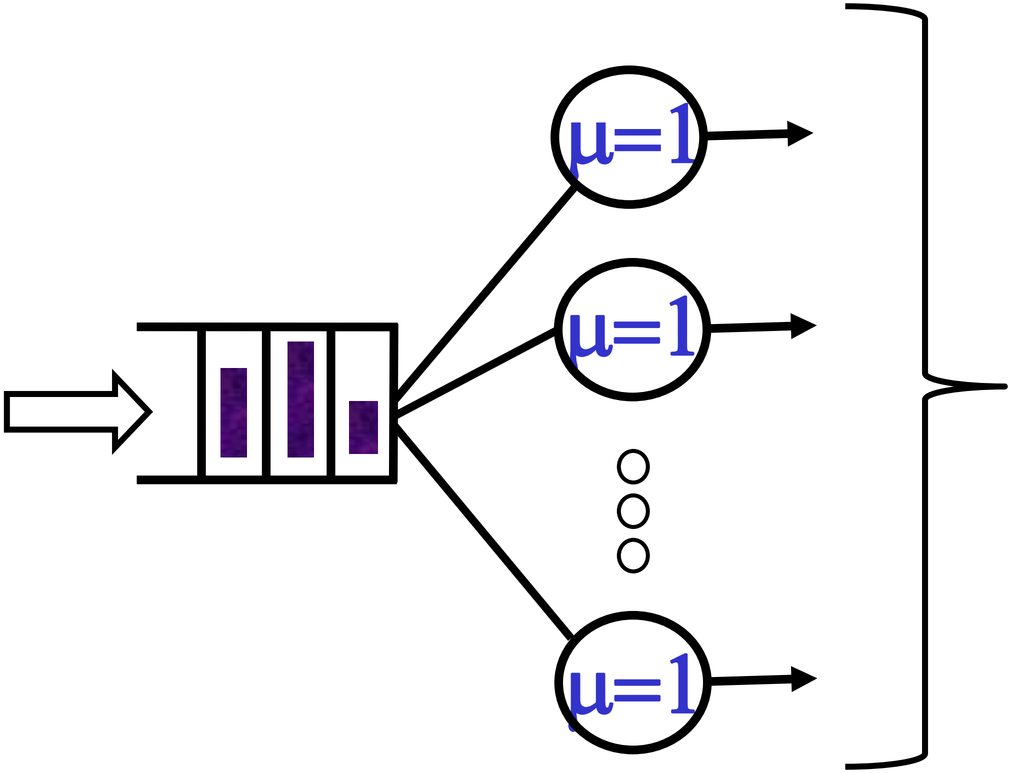
Many slow or 1 fast: Revisited



Capacity Provisioning & Scaling

Consider the following example:

Poisson process with rate $\lambda = 9 \frac{\text{jobs}}{\text{sec}}$

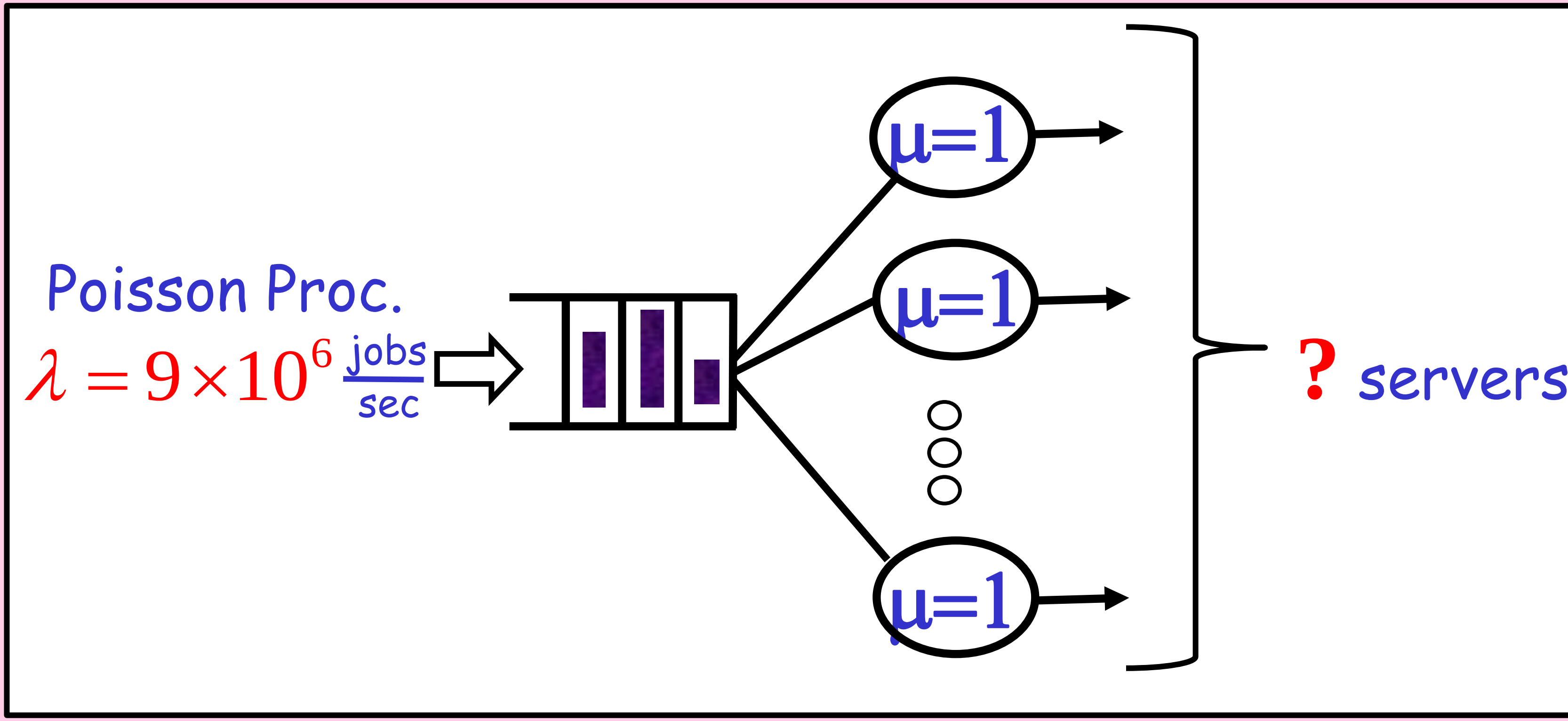


12 servers

$P_Q = \begin{array}{c}\text{Probability an arrival}\\ \text{has to queue}\end{array} = 20\%$

Capacity Provisioning & Scaling

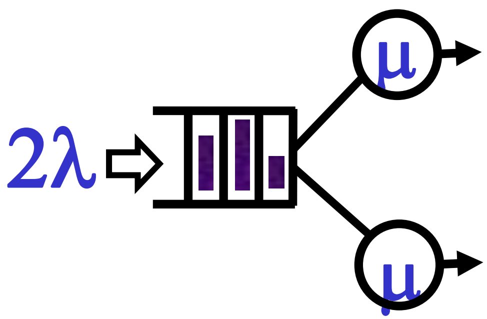
QUESTION: If arrival rate becomes 10^6 times higher, how many servers do we need to keep P_Q the same?



(a) 9.1×10^6
(b) 10×10^6
(c) 11×10^6
(d) 12×10^6
(e) 13×10^6
(f) none

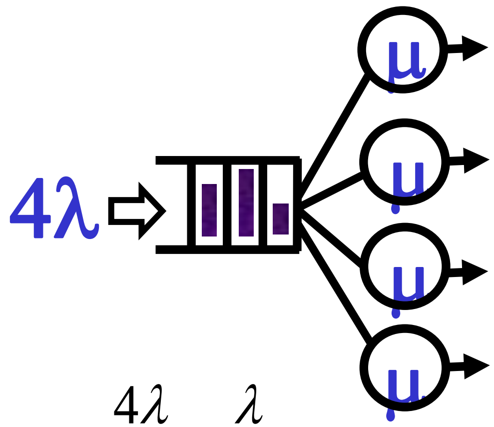
Proportional Scaling is Overkill

M/M/2



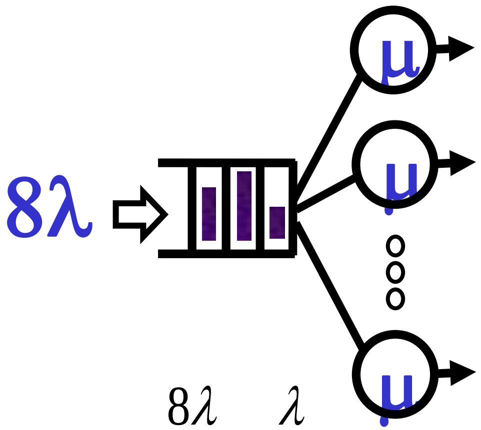
$$\rho = \frac{2\lambda}{2\mu} = \frac{\lambda}{\mu}$$

M/M/4

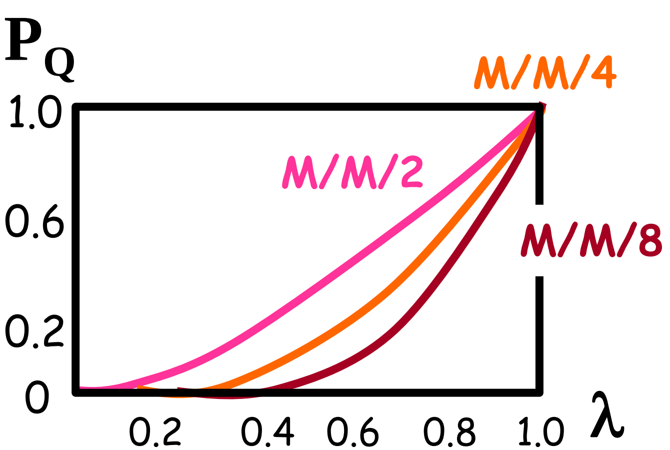


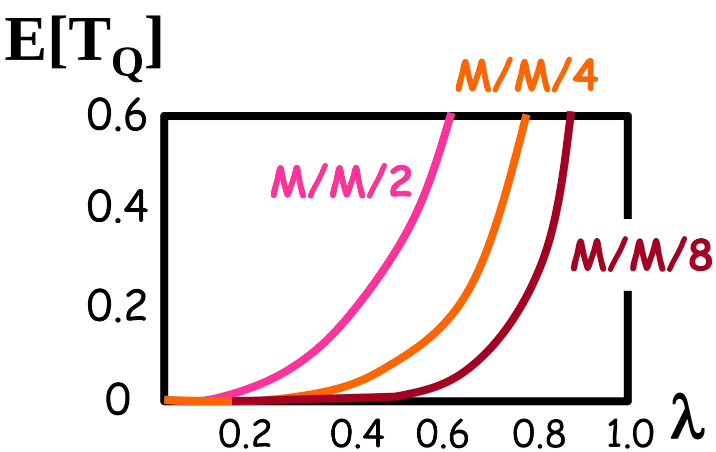
$$\rho = \frac{4\lambda}{4\mu} = \frac{\lambda}{\mu}$$

M/M/8



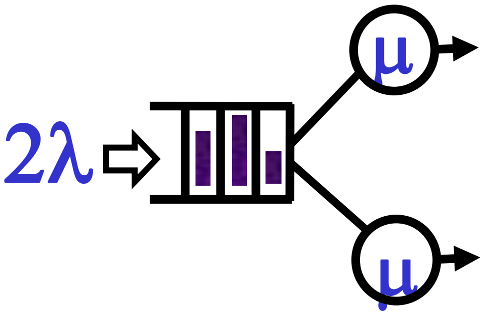
$$\rho = \frac{8\lambda}{8\mu} = \frac{\lambda}{\mu}$$





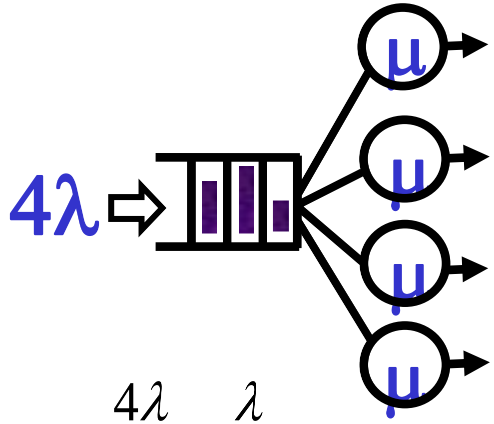
Proportional Scaling is Overkill

M/M/2



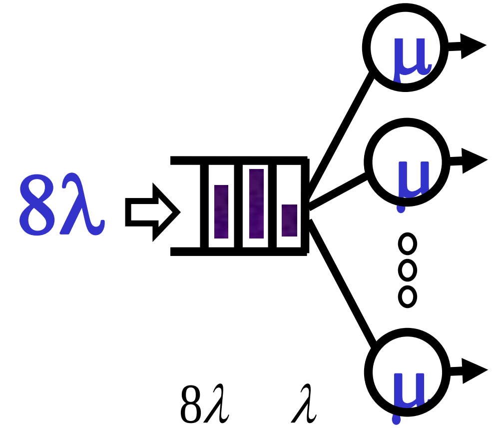
$$\rho = \frac{2\lambda}{2\mu} = \frac{\lambda}{\mu}$$

M/M/4



$$\rho = \frac{4\lambda}{4\mu} = \frac{\lambda}{\mu}$$

M/M/8

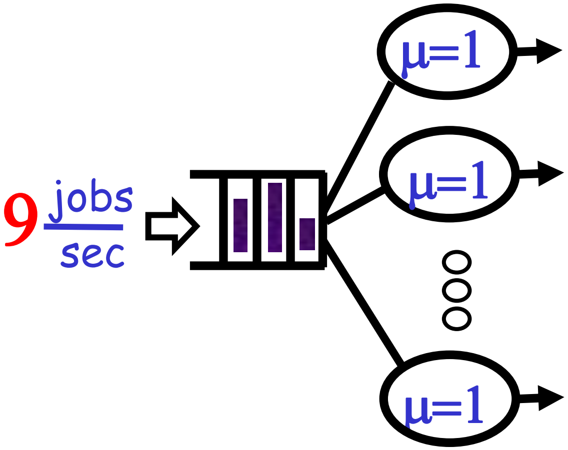


$$\rho = \frac{8\lambda}{8\mu} = \frac{\lambda}{\mu}$$

More servers at same system load $\rightarrow$ lower P_Q $\rightarrow$ lower $\mathbf{E}[T_Q]$

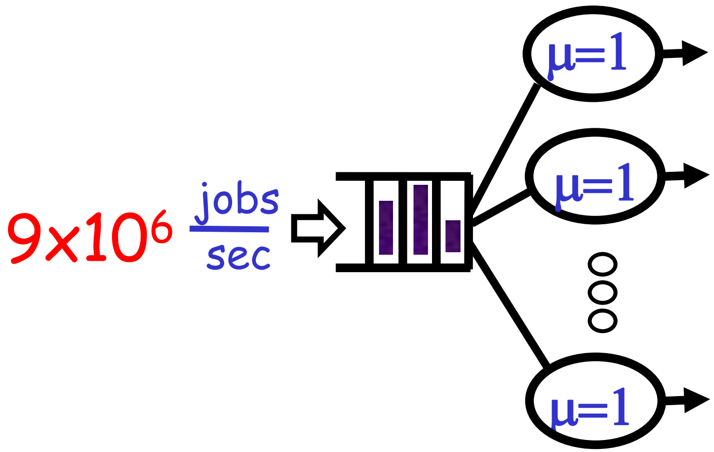
high ρ $\not\rightarrow$ high $\mathbf{E}[T_Q]$, given enough servers

Back to Capacity Provisioning



12 servers

$$P_Q \equiv$$



Q: How many servers?

A: 9.003×10^6 servers

"Square root staffing"
[Halfin, Whitt OR 1981]

Let R be the minimum #servers for stability.

Then $R + \sqrt{R}$ servers yields $P_Q = 20\%$.

Lesson: SAVE MONEY: Don't scale proportionately!

Dynamic Power Management

- Annual U.S. data center energy consumption: 100B kWh
- Unfortunately most is wasted…
- Servers are only busy 5-30% time on average, but they're left ON, wasting power. *[Gartner Report] [NYTimes]*

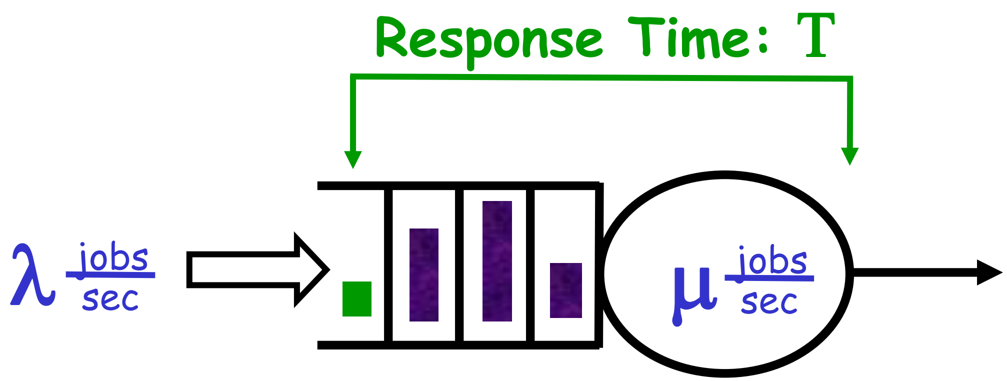
Setup time 260s 200W

- ❑ BUSY server: 200 Watts
- ❑ IDLE server: 140 Watts
- ❑ OFF server: 0 Watts

Intel Xeon E5520
2 quad-core 2.27 GHz
16 GB memory

Q: Given setup time, does dynamic power mgmt work?

M/M/1/setup model



When server is idle, immediately shuts off. Requires setup time to get it back on.

Thm: *[Welch '64]*

$$E[T^{M/M/1/Setup}] = E[T^{M/M/1}] + E[Setup]$$

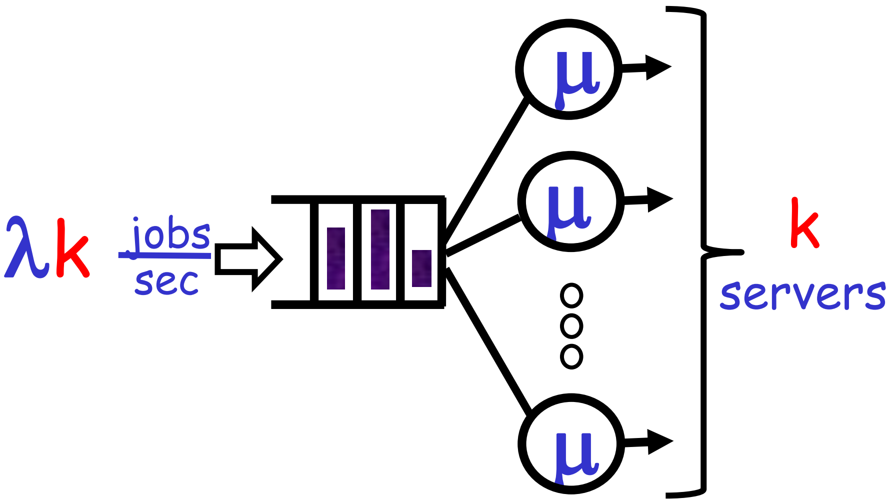
This adds 260s to response time!

QUESTION: Does setup have same effect for larger (M/M/k) systems?

Effect of setup in larger systems

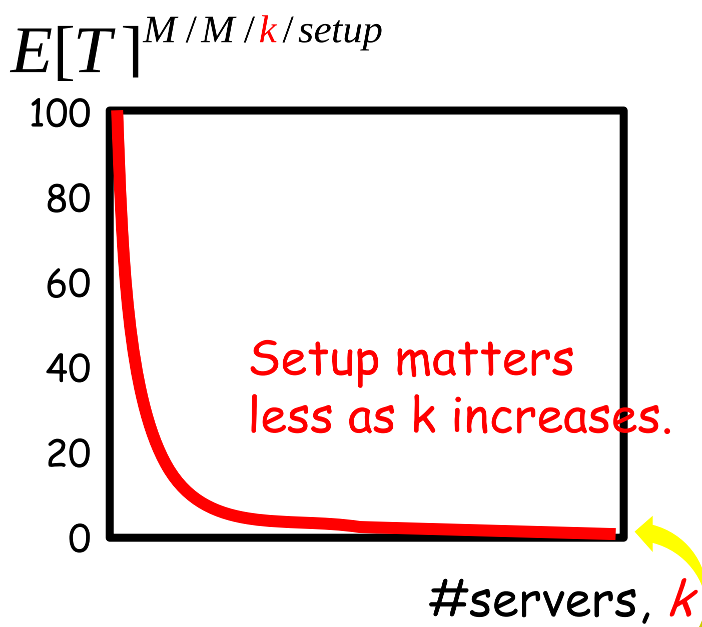
We will scale up system size, while keep load fixed.

M/M/k/setup



$$\rho = \frac{\lambda k}{\mu k} = \frac{\lambda}{\mu} = 30\%: \text{ indpt of } k$$

$$E[S]=1 \qquad E[Setup]=100$$

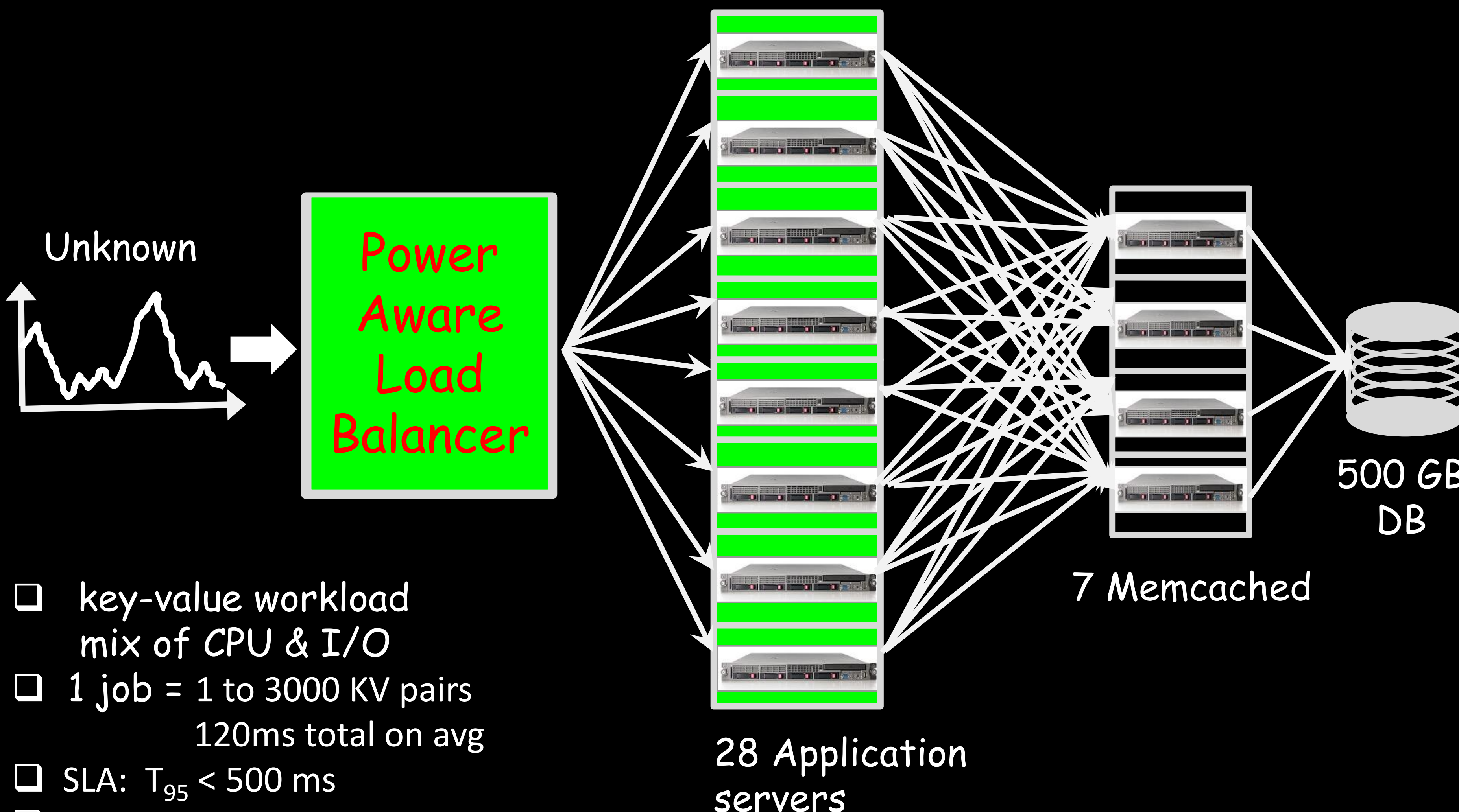


This is why dynamic power mgmt works!

[Gandhi, Doroudi, Harchol-Balter, Scheller-Wolf Sigmetrics 2013]

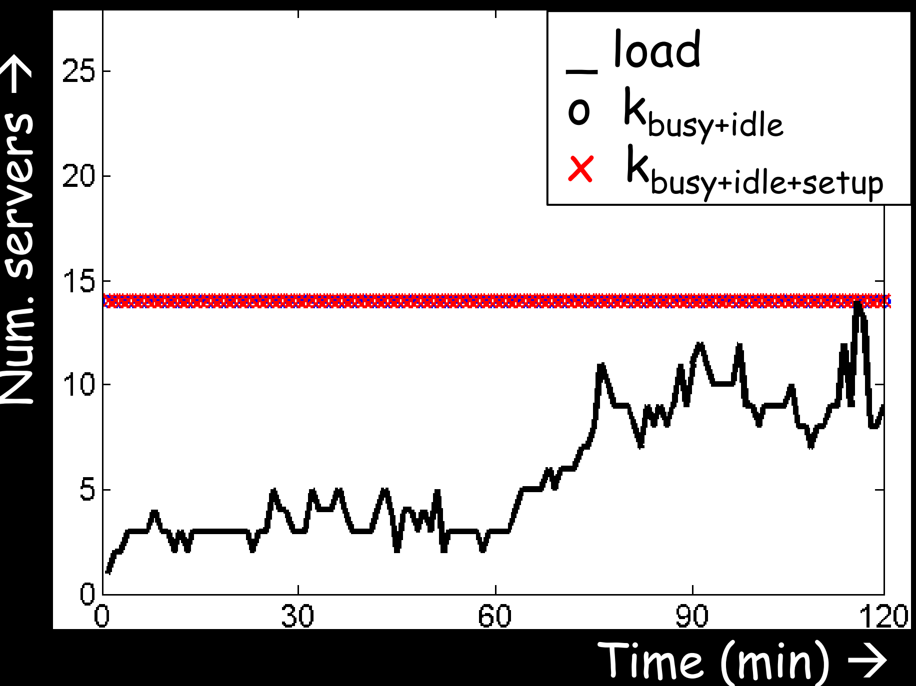
Dynamic Power Mgmt Implementation

[Gandhi, Harchol-Balter, Raghunathan, Kozuch TOCS 2012]



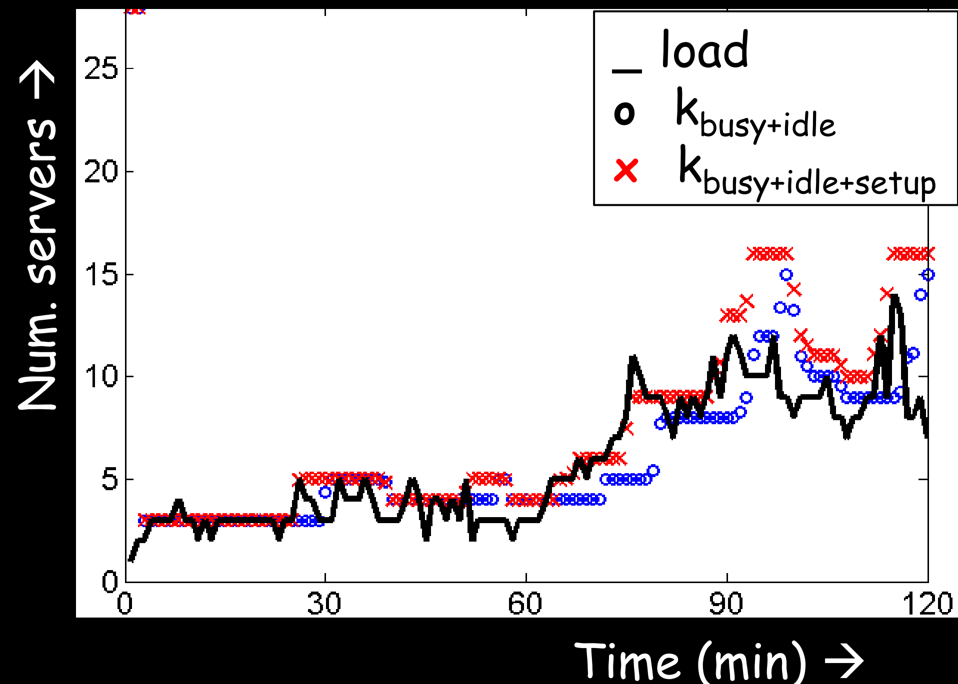
- key-value workload mix of CPU & I/O
- 1 job = 1 to 3000 KV pairs
  120ms total on avg
- SLA: T_{95} < 500 ms
- Setup time: 260 s

AlwaysOn



T_{95}=291ms, P_{avg}=2,323W

AutoScale



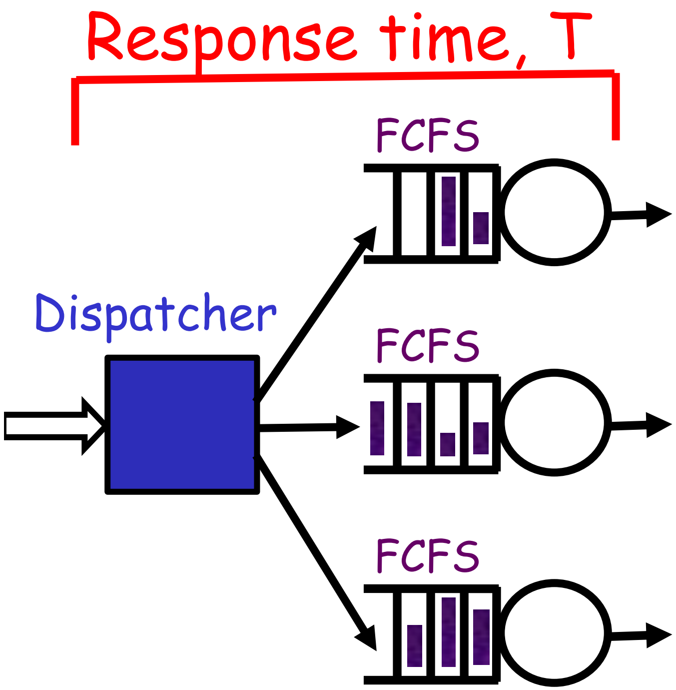
T_{95}=491ms, P_{avg}=1,297W

Within 30% of OPT
power on all our traces!

Facebook has adopted AutoScale

Dynamic Load Balancing

- F5 Big-IP
- Microsoft SharePoint
- Cisco Local Director
- Coyote Point Equalizer
- IBM Network Dispatcher
- etc.



QUESTION:
What is a good dispatching policy for minimizing E[T]?

- All hosts identical.
- Jobs i.i.d. with highly variable size distrib.

Dynamic Load Balancing

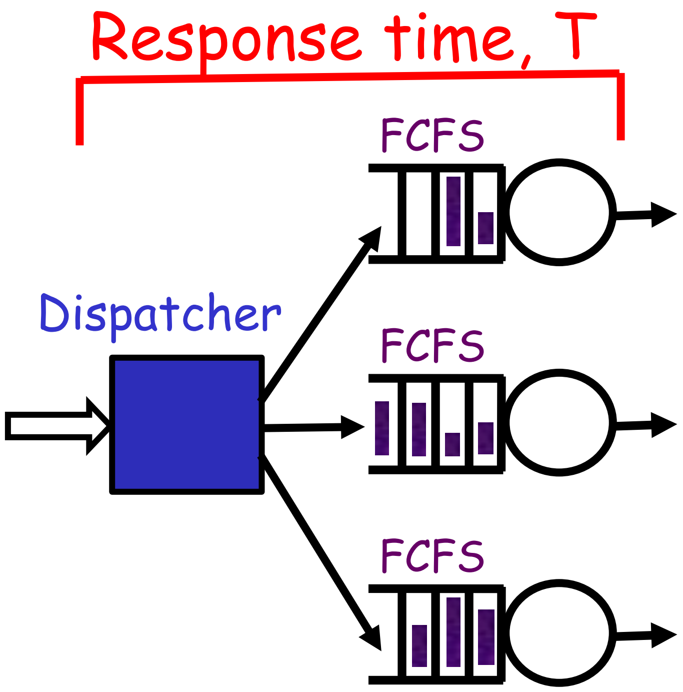
1. **Round-Robin**

2. **Join-Shortest-Queue**
   Go to host w/ fewest # jobs.

3. **Least-Work-Left**
   Go to host with least total work.

4. **Central-Queue (M/G/k)**
   Host grabs next job when free.

5. **Size-Interval Splitting**
   Jobs are split up by size among hosts.



- *All hosts identical.*
- *Jobs i.i.d. with highly variable size distrib.*

[Harchol-Balter, Crovella, Murta JPDC 99], [Harchol-Balter JACM 02], [Harchol-Balter,Scheller-Wolf,Young SIGMETRICS 09]

Dynamic Load Balancing

High E[T]

generally

Low E[T]

1. **Round-Robin**

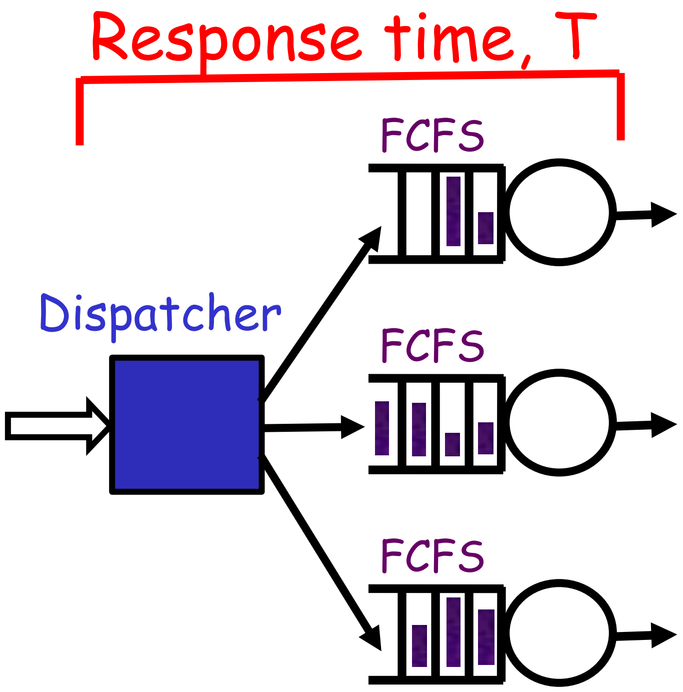
2. **Join-Shortest-Queue**
   Go to host w/ fewest # jobs.

3. **Least-Work-Left**
   Go to host with least total work.

||

4. **Central-Queue (M/G/k)**
   Host grabs next job when free.

5. **Size-Interval Splitting**
   Jobs are split up by size among hosts.



- All hosts identical.
- Jobs i.i.d. with highly variable size distrib.

[Harchol-Balter, Crovella, Murta JPDC 99], [Harchol-Balter JACM 02], [Harchol-Balter,Scheller-Wolf,Young SIGMETRICS 09]

Dynamic Load Balancing

High E[T]

generally

Low E[T]

1. **Round-Robin**

2. **Join-Shortest-Queue**
   Go to host w/ fewest # jobs.

3. **Least-Work-Left**
   Go to host with least total work.

||

4. **Central-Queue (M/G/k)**
   Host grabs next job when free.

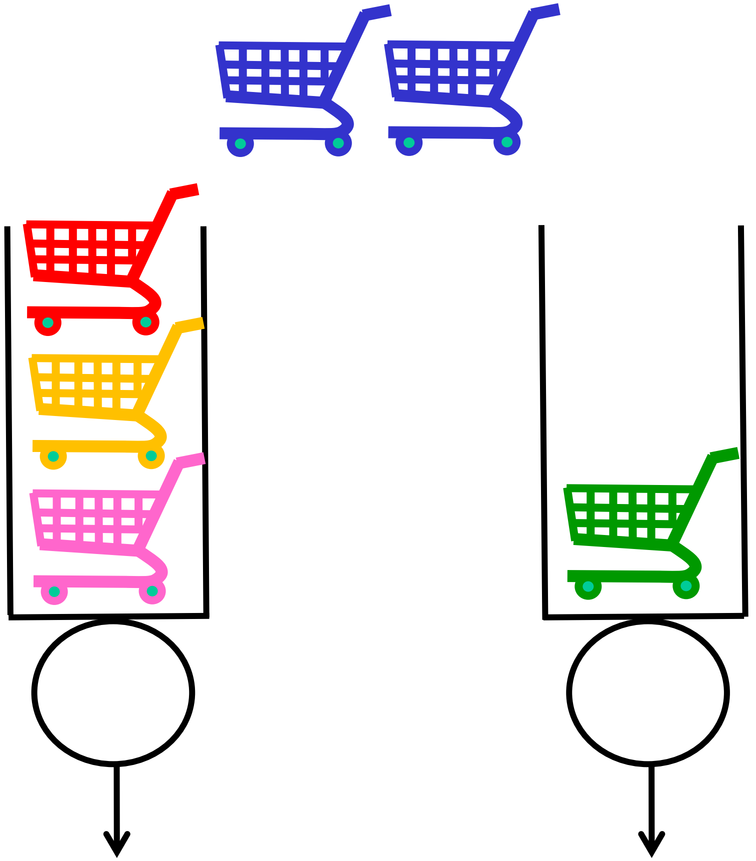
5. **Size-Interval Splitting**
   Jobs are split up by size among hosts.

Central-Queue:
+ Good utilization of servers.
+ Some isolation for smalls

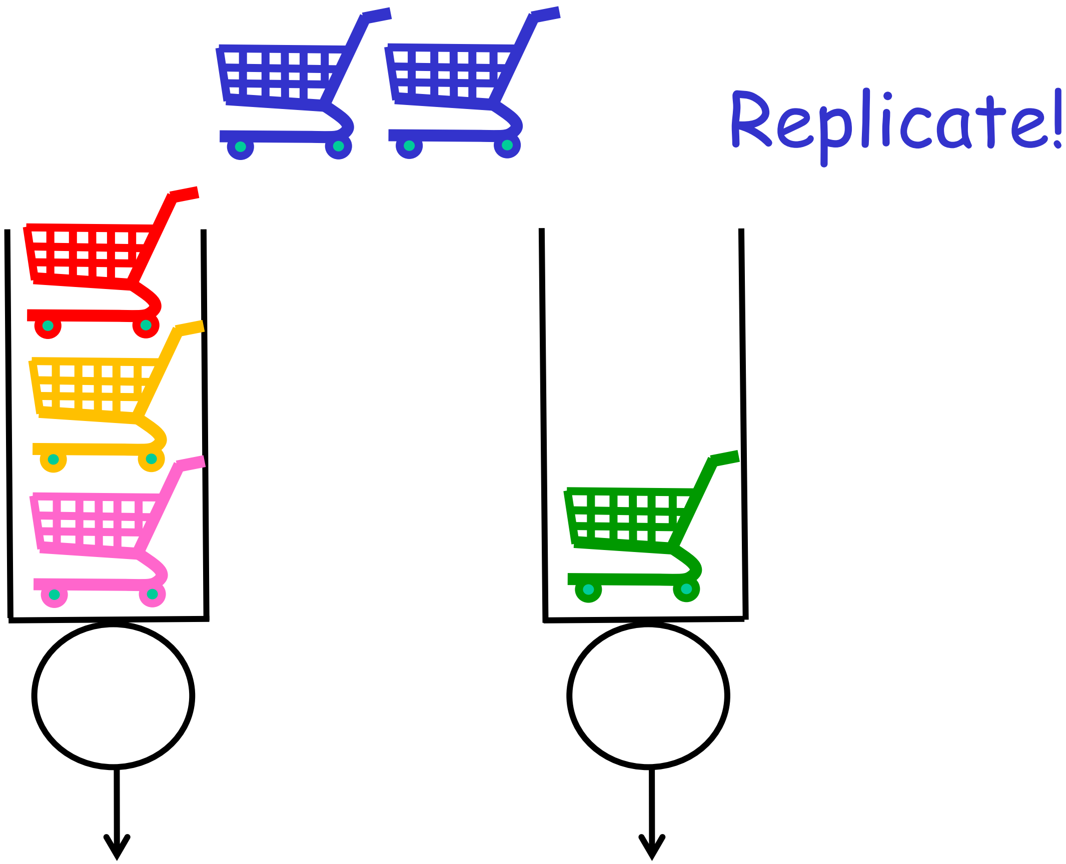
Size-Interval Task Assignment
- Worse utilization of servers.
+ Great isolation for smalls!

[Harchol-Balter, Crovella, Murta JPDC 99], [Harchol-Balter JACM 02], [Harchol-Balter, Scheller-Wolf, Young SIGMETRICS 09]

Newest work: Don't Decide. Send to all!



Newest work: Don't Decide. Send to all!

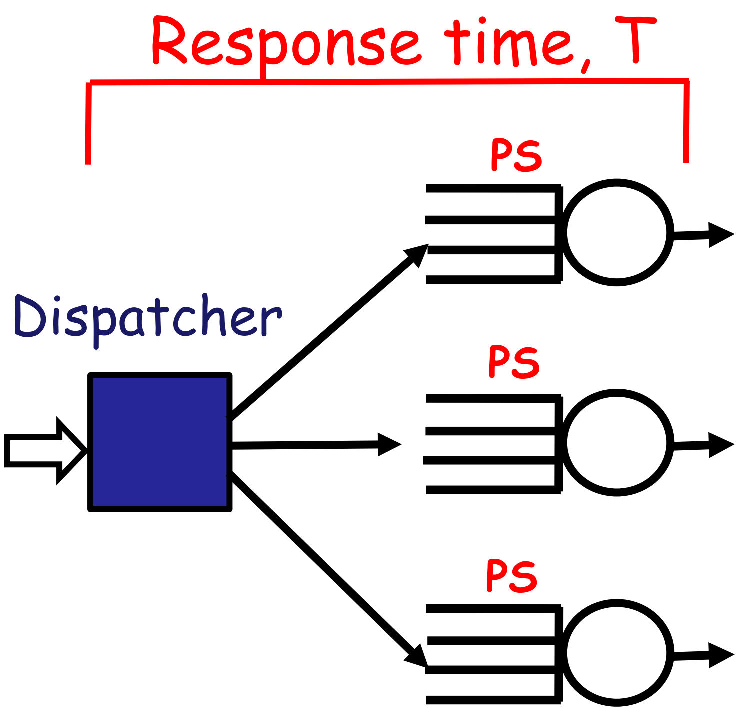


- Microsoft/Berkeley Dolly System 2012 [Ananthanarayanan, Ghodsi, Shenker, Stoica]
- Google "Tail at Scale" 2013 [Dean, Barroso]
- Berkeley Sparrow paper 2013 [Ousterhout et al.]
- DNS and Database query systems 2013 [Vulimiri et al.]
- CMU first exact analysis of replication SIGMETRICS 2015 [Gardner et al.]

Dynamic Load Balancing 2

HTTP Web requests:
- ➔ immediately dispatched to server

Commodity servers used:
- ➔ do Processor-Sharing



QUESTION:
What is a good dispatching policy for minimizing E[T]?

- All hosts identical.
- Jobs i.i.d. with highly variable size distrib.

Dynamic Load Balancing 2

High $E[T]^{FCFS}$

1. **Round-Robin**

2. **Join-Shortest-Queue**

   Go to host w/ fewest # jobs.

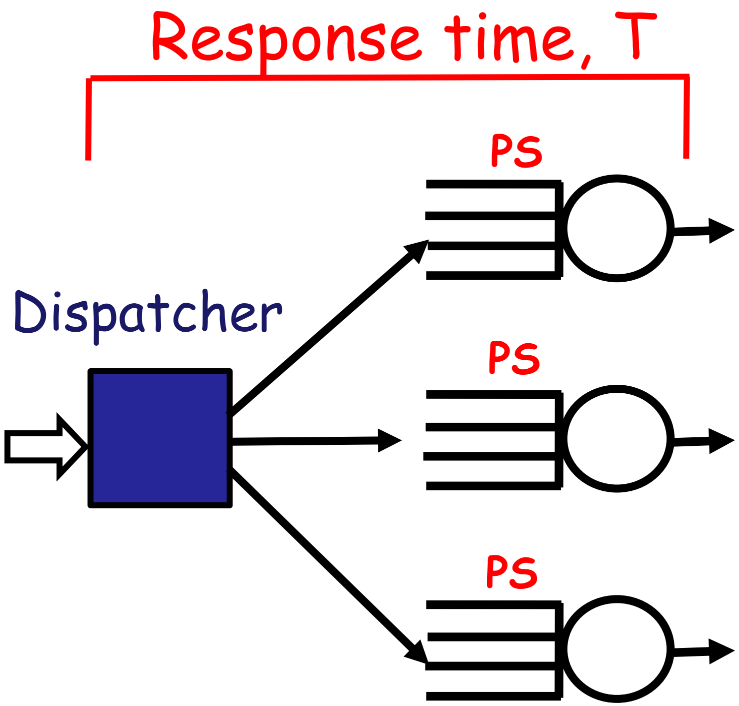
3. **Least-Work-Left**

   Go to host with least total work.

4. **Size-Interval Splitting**

   Jobs are split up by size among hosts.

Low $E[T]^{FCFS}$



- All hosts identical.
- Jobs i.i.d. with highly variable size distrib.

Dynamic Load Balancing 2

High
$E[T]^{FCFS}$

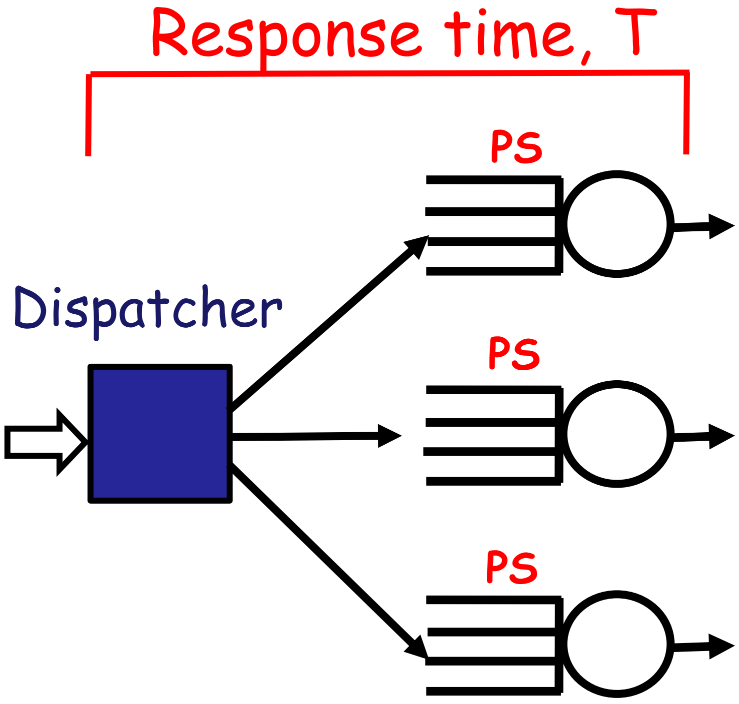
1. **Round-Robin**

2. **Join-Shortest-Queue**
   Go to host w/ fewest # jobs.

3. **Least-Work-Left**
   Go to host with least total work.

4. **Size-Interval Splitting**
   Jobs are split up by size among hosts.

Low
$E[T]^{FCFS}$



QUESTION:
What is the best of these for PS server farms?

Dynamic Load Balancing 2

High $E[T]^{FCFS}$

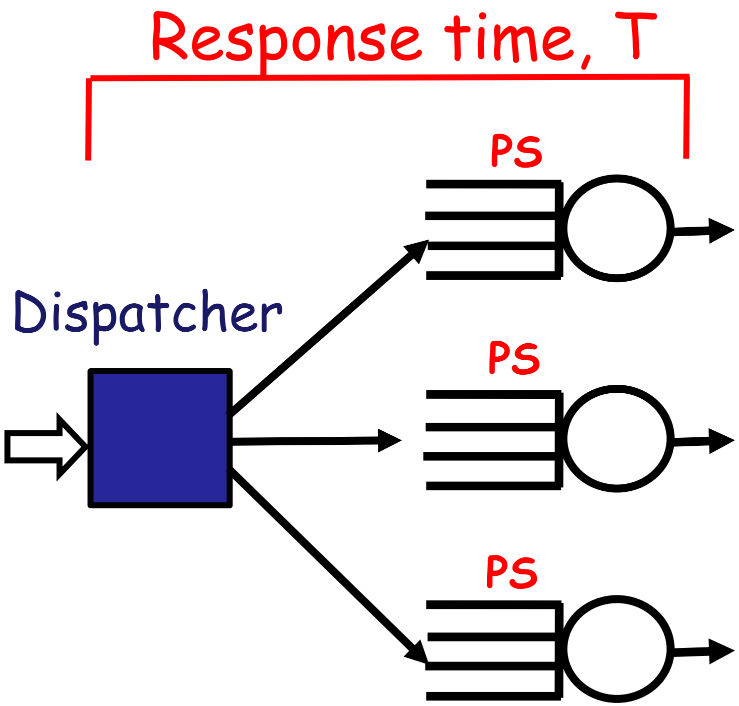
1. **Round-Robin**

2. **Join-Shortest-Queue**
   Go to host w/ fewest # jobs.

3. **Least-Work-Left**
   Go to host with least total work.

4. **Size-Interval Splitting**
   Jobs are split up by size among hosts.

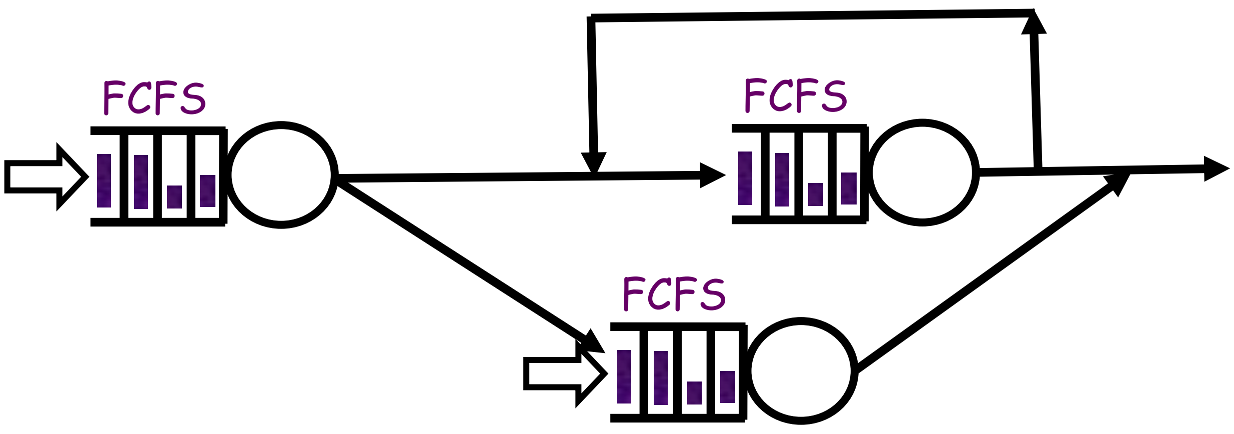
Low $E[T]^{FCFS}$



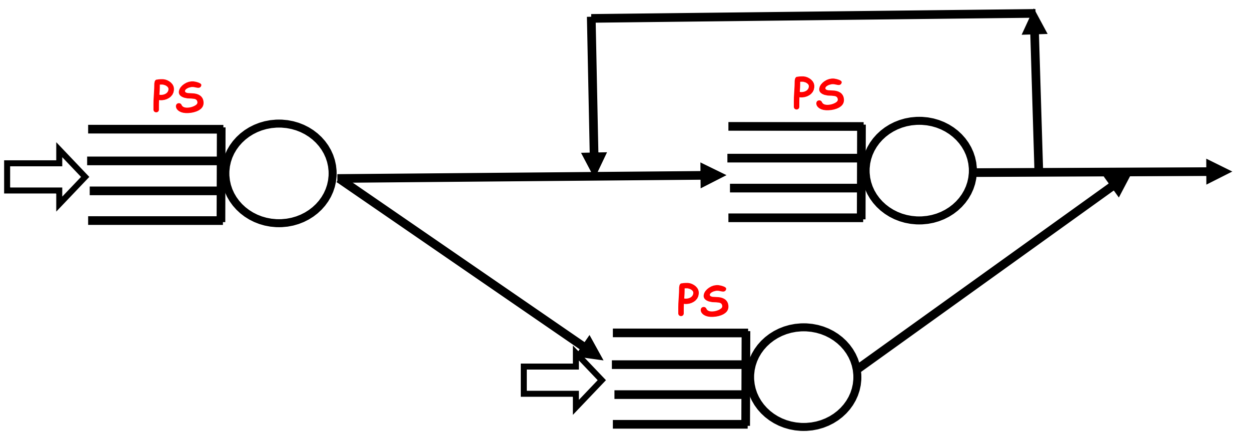
QUESTION:
What is the best of these for PS server farms?

[Gupta,Harchol-Balter,Sigman,Whitt Performance 07]

Not covering: Networks of Queues



- \+ Closed-form analysis exists
- \- Requires Poisson arrivals & indpt Exponential service times
- \+ Routes can depend on packet's "class."



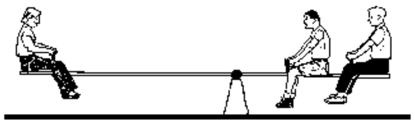
- \+ Closed-form analysis exists
- \- Requires Poisson arrivals.
- \+ General service times!
- \+ Routes and service rates can depend on packet's class.

Summary Part III

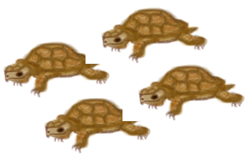
III. Multi-server queues

- Static load balancing
- Throwing away servers
- M/M/k + Comparing architectures
- Many slow servers vs. 1 fast
- Capacity provisioning & scaling
- Square root staffing
- Dynamic power management
- Dynamic load balancing/FCFS servers
- Replication
- Dynamic load balancing/PS servers

Prize-winning messages ☺



UNbalancing load is best! Throw away slow servers.



Best choice depends on job size variability.

Dynamic power mgmt works because setup time (and high load) hurt less in large systems.

Proportional scaling is overkill! Square-root staffing.

Best dispatching policies aim to mitigate effect of job size variability.

THANK YOU!

www.cs.cmu.edu/~harchol/