What Queueing Theory Teaches Us About Computer Systems Design

Mor Harchol-Balter
Computer Science Dept, CMU
Outline

I. Basic Vocabulary
  o Avg arrival rate, $\lambda$
  o Avg service rate, $\mu$
  o Avg load, $\rho$
  o Avg throughput, $X$
  o Open vs. closed systems
  o Response time, $T$
  o Waiting time, $T_Q$
  o Exponential vs. Pareto/Heavy-tailed
  o Squared coefficient of variation, $C^2$
  o Poisson Process

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

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

Avg. arrival rate: \( \lambda \) \( \frac{\text{jobs}}{\text{sec}} \)

Avg. service rate: \( \mu \) \( \frac{\text{jobs}}{\text{sec}} \)

Avg. size of job on this server: \( E[S] = \frac{1}{\mu} \)

\( S \): job size (sec) = service requirement

Example:

- On average, job needs \( 3 \times 10^6 \) cycles
- Machine executes \( 9 \times 10^6 \) 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.

\( \lambda < \mu \) throughout
Vocabulary

$\lambda$ jobs/sec $\rightarrow$ FCFS $\rightarrow$ $\mu$ jobs/sec

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

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

Example:

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

$\rho = \frac{2}{3}$
Defn: Throughput $\lambda$ denotes the average rate at which jobs complete (jobs/sec)

QUESTION:
Which has higher throughput, $\lambda$?

$\lambda < \mu$ throughout

$\lambda$ (jobs/sec) $\rightarrow$ $\mu$ (jobs/sec)

$\lambda$ (jobs/sec) $\rightarrow$ $2\mu$ (jobs/sec)
More Vocabulary

\[ X = \lambda \]  (assuming no jobs dropped)
Open versus Closed Systems

**Open**

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

\[ X = \lambda \]

\[ \lambda \rightarrow \mu \]

**Closed Batch**

\[ MPL N: \text{ fixed #jobs} \]

\[ \rho = 1 \]

\[ X = \mu \]

**Closed Interactive**

\[ MPL N: \text{ fixed #users} \]

\[ Z: \text{“think time”} \]

\[ \rho = 1 - \Pr\{\text{All thinking}\} \]

\[ X = \rho \mu \]
More Vocabulary

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

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

$T = \text{response time}$

$T_Q = \text{queueing time (waiting time)}$

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

A: Variability in the arrival process & service requirements
Variability in job size, $S$

$S : \text{job size}$

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

$\rho = \lambda E[S] = \frac{\lambda}{\mu}$
"Most jobs are small; few jobs are large"

\[ S \sim \text{Exp}(\mu) \]

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

\[ S \sim \text{Pareto}(\alpha) \]

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

heavy tail
**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.”

\[
S \sim \text{Exp}(\mu)
\]

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

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

\[
\Pr\{S > x\} = \frac{1}{x}
\]

---

**heavy tail**
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]

\[ S \sim \text{Pareto} \]

\[ \Pr\{S > x\} = \frac{1}{x} \]

heavy tail
Exponential Job Size Distribution

\[ S \sim \text{Exp}(\mu) \implies Pr\{S > x\} = e^{-\mu x} \]

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

\( S \) is time until coin w/prob \( \mu \delta \) comes up heads

\( S \) is memoryless!
Variability in Job Sizes

Squared Coefficient of Variation\

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

\[ C^2 = 0 \]
\[ C^2 \approx 0.02 \]
\[ C^2 = \frac{1}{3} \]
\[ C^2 = 1 \]
\[ C^2 \approx 50 - 100 \]
\[ C^2 = \infty \]

**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

\[ C^2 = 0 \quad \text{Deterministic} \]
\[ C^2 \approx 0.02 \quad \text{Human IQs} \]
\[ C^2 = \frac{1}{3} \quad \text{Uniform}(0,b) - \text{for any } b \]
\[ C^2 = 1 \quad \text{Exponential distribution} \]

Squared Coefficient of Variation

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

\[ C^2 = 50 - 100 \quad \text{Unix process lifetimes} \]

\[ C^2 = \infty \quad \text{Pareto distribution} \]
Variability

\[ \lambda \text{ jobs/sec} \rightarrow T \rightarrow \mu \text{ jobs/sec} \]

Variability in arrival process

Variability in job size, \( S \)

\[ S: \text{ job size} \]

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

\[ \rho = \lambda E[S] = \frac{\lambda}{\mu} \]
QUESTION: What’s a Poisson process with rate $\lambda$?

Hint: It’s related to $\text{Exp}(\lambda)$. 
Poisson Process with rate $\lambda$

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

$S \sim \text{Exp}(\lambda)$  $S \sim \text{Exp}(\lambda)$  $S \sim \text{Exp}(\lambda)$
I. Basic Vocabulary

- Avg arrival rate, $\lambda$
- Avg service rate, $\mu$
- Avg load, $\rho$
- 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, $\lambda$
   - Avg service rate, $\mu$
   - Avg load, $\rho$
   - Avg throughput, $X$
   - Open vs. closed systems

II. Single-server queues
   - D/D/1, M/M/1, M/G/1
   - Inspection Paradox
   - Effect of job size variability
   - 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

\[ \lambda \quad \text{jobs/second} \quad \rightarrow \quad T \]

\[ T_Q \]

\[ \mu \quad \text{jobs/second} \quad \rightarrow \quad S : \text{job size} \]

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

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

\[ \text{D/D/1} \]

Deterministic service times

\[ \text{M/M/1} \]

Exponential inter-arrival times
Exponential service times

1 server

\[ \text{M/G/1} \]

General service times

M=“memoryless”=“Markovian”
Single-Server Queue

$\lambda \text{ jobs/sec} \rightarrow \mu \text{ jobs/sec}$

$T$

$T_Q$

$S : \text{ job size}$

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

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

D/D/1

M/M/1

M/G/1

Q: Does low $\rho \Rightarrow$ low $E[T_Q]$?
Single-Server Queue

\[ \lambda \] jobs/sec \[ \rightarrow \] \[ \mu \] jobs/sec

\[ T \]

\[ T_Q \]

\[ S: \text{job size} \]

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

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

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

\[ \lambda \text{ jobs/sec} \rightarrow \mu \text{ jobs/sec} \]

\[ T \]

\[ T_Q \]

\[ S : \text{job size} \]

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

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

\[ E[T_Q] \]

\[ 0 \quad 0.2 \quad 0.4 \quad 0.6 \quad 0.8 \]

\[ 0 \quad 5 \quad 10 \quad 15 \quad 20 \]

\[ M/G/1 \quad C^2 = 100 \]

\[ M/G/1 \quad C^2 = 10 \]

\[ M/M/1 \]

\[ D/D/1 \]

low load does NOT imply low wait
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

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

“Inspection Paradox”
$S$: job size

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

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

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

Low $\rho \nRightarrow$ 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.
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

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

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

Doubling \( \rho \) can increase \( E[T_Q] \) by factor of 4 to \( \infty \)
Equalizing the wait for men & women

\[ \frac{\lambda}{2} \text{ ppl/sec} \rightarrow \mu \text{ ppl/sec} \rightarrow \text{ppl} \]

\[ \frac{\lambda}{2} \text{ ppl/sec} \rightarrow \mu \text{ ppl/sec} \rightarrow \text{ppl} \]

\[ \lambda \text{ ppl/sec} \rightarrow 2\mu \text{ ppl/sec} \rightarrow \text{ppl} \]

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

\[ \frac{\lambda}{2} \text{ ppl/sec} \rightarrow \mu \text{ ppl/sec} \rightarrow \text{ waiting time for women} \]

Waiting time for women is still a factor of 2 higher.

Also true under M/G/1 model.

For what models is this not true?
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!
QUESTION:
Which scheduling policy is best for minimizing $E[T]$?

**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

\[ \Pr\{S > x\} = \frac{1}{x} \]

\[ \begin{align*}
\lambda & \text{ jobs/sec} \\
\mu & \text{ jobs/sec}
\end{align*} \]

E[T]

\[ \begin{align*}
C^2 &= 10 \\
C^2 &= 100
\end{align*} \]
We saw: \( E[T]_{\text{SRPT}} \ll E[T]_{\text{PS}} \)

But isn’t SRPT unfair to large jobs, when compared to PS?
Unfairness Question

Let $S \sim$ Bounded Pareto with $\max = 10^{10}$
Let $\rho = 0.9$

QUESTION:
Which queue does Mr. Max prefer?

\[ \Pr\{S > x\} = \frac{1}{x} \]
Let $S \sim \text{Bounded Pareto} \ (\alpha = 1.1)$ with $\max = 10^{10}$

Let $\rho = 0.9$

$$\Pr\{S > x\} = \frac{1}{x}$$
Unfairness Question

**All-can-win-theorem:**
[Bansal, Harchol-Balter, Sigmetrics 2001]

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

$$E[T(x)]_{SRPT} < E[T(x)]_{PS}$$

for all job size $x$.

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

Defies Kleinrock’s Conservation Law
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?
**Q:** What is being scheduled?
**A:** Bottleneck device is limited ISP bandwidth.

**Q:** How is size being used?
**A:** $S = \text{Size of requests} = \text{Size of file} \sim \text{Pareto}(\alpha = 1)$

---

Site buys limited fraction of ISP’s bandwidth (say 100Mbps)

Schedule the sharing of this 100Mbps among 1000 clients.
Linux Implementation

Sockets take turns draining: PS

Socket of file w/smallest remaining data feeds first: SRPT

1st

2nd

3rd

priority queues.
Mean response time results

$E[T]$
Response time as \textit{fcn} of Size

\[ E[T(x)] \]

\( (\rho = 0.8) \)
**Caution: Open versus Closed**

**Open**

\[ \lambda \rightarrow \mu \]

**Closed**

MPL N: fixed #users

Z: think time

Response Time: \( T \)

**QUESTION:** When run with same load \( \rho \), which has higher \( E[T] \)?

(a) Open  
(b) Closed  
(c) Same
Caution: Open versus Closed

Performance of Auction Site
[Schroeder, Wierman, Harchol-Balter NSDI 2006]
Closed & open systems run w/ same job size distribution and same load.

[Schroeder, Wierman, Harchol-Balter, NSDI 06]
II. Single-server queues

- D/D/1, M/M/1, M/G/1
- Inspection Paradox
- Effect of job size variability
- 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.

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

Smart scheduling can combat job size variability.

Policies that seem unfair may not be.

Closed systems behave very differently from open.
I. Basic Vocabulary
- Avg arrival rate, $\lambda$
- Avg service rate, $\mu$
- Avg load, $\rho$
- 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 variability
- 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 $\lambda$ jobs/sec

QUESTION: What is the optimal $p$ to minimize $E[T]$?

(a) $p = \frac{2}{3}$  (b) $p > \frac{2}{3}$  (c) $p < \frac{2}{3}$
Load Balancing

Poisson process with rate $\lambda$ jobs/sec

Opt $p$

$p=1$ throw away slower server
Poisson process with rate \( \lambda \frac{\text{jobs}}{\text{sec}} \)

Central queue. Server takes job when free.

Job size \( S \sim \text{Exp}(\mu) \)

\[
\rho \equiv \text{System Load} \equiv \frac{\lambda}{k \mu}
\]
Q: Which is best for minimizing $E[T]$?
3 Architectures

M/M/k

\[
\lambda \frac{\text{jobs}}{\text{sec}} \rightarrow \mu \rightarrow \mu \rightarrow \mu
\]

Splitting

\[
\frac{1}{k} \quad \frac{1}{k}
\]

M/M/1 fast

\[
\lambda \frac{\text{jobs}}{\text{sec}} \rightarrow k \mu
\]

\[
E[T_Q]_{M/M/1} = \frac{\rho}{1 - \rho} \cdot E[S]
\]
Many slow or 1 fast?

**M/M/k**

\[ \rho = \frac{\lambda}{k \mu} \]

**M/M/1fast**

\[ \rho = \frac{\lambda}{k \mu} \]

**QUESTION:** Which is best for minimizing $E[T]$?
Many slow or 1 fast?

\[ E[T] \]

\[ \Pr\{T > t\} \]

- \[ M/M/10 \]
- \[ M/M/1 fast \]

\[ \lambda \]

\begin{tabular}{|c|c|c|c|c|c|}
\hline
\( \lambda \) & 0.5 & 1.0 & 2.0 & 3.0 & 4.0 & 6.0 & 8.0 & 10.0 \\
\hline
\( E[T] \) & 0.2 & 0.6 & 1.0 & 2.0 & 3.0 & 4.0 & 6.0 & 8.0
\hline
\end{tabular}
Many slow or 1 fast: Revisited

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

$$C_G^2 = 100$$
Many slow or 1 fast: Revisited

\[ E[T] \]

\[ \lambda \]

\[ M/G/1 \text{fast} \]

\[ M/G/10 \]
Consider the following example:

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

$P_Q = \text{Probability an arrival has to queue} = 20\%$
QUESTION: If arrival rate becomes $10^6$ times higher, how many servers do we need to keep $P_Q$ the same?

Poisson Proc.

$\lambda = 9 \times 10^6 \frac{\text{jobs}}{\text{sec}}$

(a) $9.1 \times 10^6$
(b) $10 \times 10^6$
(c) $11 \times 10^6$
(d) $12 \times 10^6$
(e) $13 \times 10^6$
(f) none
Proportional Scaling is Overkill

$M/M/2$

$2\lambda \Rightarrow \begin{array}{c}
\mu \\
\mu
\end{array}$

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

$M/M/4$

$4\lambda \Rightarrow \begin{array}{c}
\mu \\
\mu \\
\mu \\
\mu
\end{array}$

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

$M/M/8$

$8\lambda \Rightarrow \begin{array}{c}
\mu \\
\mu \\
\mu \\
\mu
\end{array}$

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

**P**$_Q$

- $M/M/2$
- $M/M/4$
- $M/M/8$

$E[T_Q]$

- $M/M/2$
- $M/M/4$
- $M/M/8$
Proportional Scaling is Overkill

$M/M/2$

$2\lambda \Rightarrow$

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

$M/M/4$

$4\lambda \Rightarrow$

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

$M/M/8$

$8\lambda \Rightarrow$

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

More servers at same system load $\Rightarrow$ lower $P_Q$ $\Rightarrow$ lower $E[T_Q]$

high $\rho \Rightarrow$ high $E[T_Q]$, given enough servers
Back to Capacity Provisioning

Let $R$ be the minimum number of servers for stability.

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

“Square root staffing”
[Halpin, Whitt OR 1981]

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]

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

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.

Response Time: $T$

$$\frac{\lambda}{\text{sec}} \quad \lambda \quad \mu \quad \text{jobs} \quad \mu \quad \text{sec}$$

**Thm: [Welch '64]**

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

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.

\( \lambda k \) \( \text{jobs/sec} \)

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

\( E[S] = 1 \quad E[\text{Setup}] = 100 \)

\[ E[T]_{M/M/k/setup} \]

Setup matters less as \( k \) increases.

This is why dynamic power mgmt works!

\[ [\text{Gandhi, Doroudi, Harchol-Balter, Scheller-Wolf Sigmetrics 2013}] \]
Dynamic Power Mgmt Implementation

- 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

[Gandhi, Harchol-Balter, Raghunathan, Kozuch TOCS 2012]
Facebook has adopted AutoScale

- **AlwaysOn**
  - $T_{95}=291\text{ms}$, $P_{avg}=2,323\text{W}$

- **AutoScale**
  - $T_{95}=491\text{ms}$, $P_{avg}=1,297\text{W}$

Within 30% of OPT power on all our traces!
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.

Response time, $T$

- 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]
1. **Round-Robin**
   Go to host with fewest number of jobs.

2. **Join-Shortest-Queue**
   Go to host with least total work.

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.

---

**Response time, T**

- 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

1. **Round-Robin**
   Go to host with fewest # jobs.

2. **Join-Shortest-Queue**
   Go to host with least total work.

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!
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.]
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**
   - Go to the host with the fewest number of jobs.

2. **Join-Shortest-Queue**
   - Go to the host with the least total work.

3. **Least-Work-Left**
   - Go to the host with the 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 distribution.

Response time, $T$
1. **Round-Robin**
   - Go to host with fewest number of jobs.

2. **Join-Shortest-Queue**
   - Go to host with the shortest queue.

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

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

**QUESTION:**
What is the best of these for PS server farms?
1. Round-Robin

2. Join-Shortest-Queue
   Go to host with 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.

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.
- Proportional scaling is overkill! Square-root staffing.
- Dynamic power mgmt works because setup time (and high load) hurt less in large systems.
- Best dispatching policies aim to mitigate effect of job size variability.
THANK YOU!

www.cs.cmu.edu/~harchol/