

## Performance Modeling and Design of Computer Systems

Computer systems design is full of conundrums:

- Given a choice between a single machine with speed  $s$ , or  $n$  machines each with speed  $s/n$ , which should we choose?
- If both the arrival rate and service rate double, will the mean response time stay the same?
- Should systems really aim to balance load, or is this a convenient myth?
- If a scheduling policy favors one set of jobs, does it necessarily hurt some other jobs, or are these “conservation laws” being misinterpreted?
- Do greedy, shortest-delay, routing strategies make sense in a server farm, or is what is good for the individual disastrous for the system as a whole?
- How do high job size variability and heavy-tailed workloads affect the choice of a scheduling policy?
- How should one trade off energy and delay in designing a computer system?
- If 12 servers are needed to meet delay guarantees when the arrival rate is 9 jobs/sec, will we need 12,000 servers when the arrival rate is 9,000 jobs/sec?

Tackling the questions that systems designers care about, this book brings queueing theory decisively back to computer science. The book is written with computer scientists and engineers in mind and is full of examples from computer systems, as well as manufacturing and operations research. Fun and readable, the book is highly approachable, even for undergraduates, while still being thoroughly rigorous and also covering a much wider span of topics than many queueing books.

Readers benefit from a lively mix of motivation and intuition, with illustrations, examples, and more than 300 exercises – all while acquiring the skills needed to model, analyze, and design large-scale systems with good performance and low cost. The exercises are an important feature, teaching research-level counterintuitive lessons in the design of computer systems. The goal is to train readers not only to customize existing analyses but also to invent their own.

Mor Harchol-Balter is an Associate Professor in the Computer Science Department at Carnegie Mellon University. She is a leader in the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, having served as technical program committee chair in 2007 and conference chair in 2013.

Cambridge University Press

978-1-107-02750-3 - Performance Modeling and Design of Computer Systems: Queueing Theory in Action

Mor Harchol-Balter

Frontmatter

[More information](#)

---

Cambridge University Press

978-1-107-02750-3 - Performance Modeling and Design of Computer Systems: Queueing Theory in Action

Mor Harchol-Balter

Frontmatter

[More information](#)

# Performance Modeling and Design of Computer Systems

*Queueing Theory in Action*

**Mor Harchol-Balter**

*Carnegie Mellon University, Pennsylvania*



**CAMBRIDGE**  
UNIVERSITY PRESS

Cambridge University Press

978-1-107-02750-3 - Performance Modeling and Design of Computer Systems: Queueing Theory in Action

Mor Harchol-Balter

Frontmatter

[More information](#)

CAMBRIDGE UNIVERSITY PRESS

Cambridge, New York, Melbourne, Madrid, Cape Town,  
Singapore, São Paulo, Delhi, Mexico City

Cambridge University Press

32 Avenue of the Americas, New York, NY 10013-2473, USA

[www.cambridge.org](http://www.cambridge.org)

Information on this title: [www.cambridge.org/9781107027503](http://www.cambridge.org/9781107027503)

© Mor Harchol-Balter 2013

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2013

Printed in the United States of America

*A catalog record for this publication is available from the British Library.*

*Library of Congress Cataloging in Publication Data*

Harchol-Balter, Mor, 1966–

Performance modeling and design of computer systems : queueing theory in action / Mor Harchol-Balter.

pages cm

Includes bibliographical references and index.

ISBN 978-1-107-02750-3

1. Transaction systems (Computer systems) – Mathematical models. 2. Computer systems – Design and construction – Mathematics. 3. Queueing theory.

4. Queueing networks (Data transmission) I. Title.

QA76.545.H37 2013

519.8'2–dc23 2012019844

ISBN 978-1-107-02750-3 Hardback

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party Internet websites referred to in this publication and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

Cambridge University Press

978-1-107-02750-3 - Performance Modeling and Design of Computer Systems: Queueing Theory in Action

Mor Harchol-Balter

Frontmatter

[More information](#)

---

*To my loving husband Andrew, my awesome son Danny,  
and my parents, Irit and Micha*

*I have always been interested in finding better designs for computer systems, designs that improve performance without the purchase of additional resources. When I look back at the problems that I have solved and I look ahead to the problems I hope to solve, I realize that the problem formulations keep getting simpler and simpler, and my footing less secure. Every wisdom that I once believed, I have now come to question: If a scheduling policy helps one set of jobs, does it necessarily hurt some other jobs, or are these “conservation laws” being misinterpreted? Do greedy routing strategies make sense in server farms, or is what is good for the individual actually disastrous for the system as a whole? When comparing a single fast machine with  $n$  slow machines, each of  $1/n$ th the speed, the single fast machine is typically much more expensive – but does that mean that it is necessarily better? Should distributed systems really aim to balance load, or is this a convenient myth? Cycle stealing, where machines can help each other when they are idle, sounds like a great idea, but can we quantify the actual benefit? How much is the performance of scheduling policies affected by variability in the arrival rate and service rate and by fluctuations in the load, and what can we do to combat variability? Inherent in these questions is the impact of real user behaviors and real-world workloads with heavy-tailed, highly variable service demands, as well as correlated arrival processes. Also intertwined in my work are the tensions between theoretical analysis and the realities of implementation, each motivating the other. In my search to discover new research techniques that allow me to answer these and other questions, I find that I am converging toward the fundamental core that defines all these problems, and that makes the counterintuitive more believable.*

# Contents

---

<i>Preface</i>	xvii
<i>Acknowledgments</i>	xxiii

## I Introduction to Queueing

<b>1 Motivating Examples of the Power of Analytical Modeling</b>	<b>3</b>
1.1 What Is Queueing Theory?	3
1.2 Examples of the Power of Queueing Theory	5
<b>2 Queueing Theory Terminology</b>	<b>13</b>
2.1 Where We Are Heading	13
2.2 The Single-Server Network	13
2.3 Classification of Queueing Networks	16
2.4 Open Networks	16
2.5 More Metrics: Throughput and Utilization	17
2.6 Closed Networks	20
2.6.1 Interactive (Terminal-Driven) Systems	21
2.6.2 Batch Systems	22
2.6.3 Throughput in a Closed System	23
2.7 Differences between Closed and Open Networks	24
2.7.1 A Question on Modeling	25
2.8 Related Readings	25
2.9 Exercises	26

## II Necessary Probability Background

<b>3 Probability Review</b>	<b>31</b>
3.1 Sample Space and Events	31
3.2 Probability Defined on Events	32
3.3 Conditional Probabilities on Events	33
3.4 Independent Events and Conditionally Independent Events	34
3.5 Law of Total Probability	35
3.6 Bayes Law	36
3.7 Discrete versus Continuous Random Variables	37
3.8 Probabilities and Densities	38
3.8.1 Discrete: Probability Mass Function	38
3.8.2 Continuous: Probability Density Function	41
3.9 Expectation and Variance	44
3.10 Joint Probabilities and Independence	47

3.11	Conditional Probabilities and Expectations	49
3.12	Probabilities and Expectations via Conditioning	53
3.13	Linearity of Expectation	54
3.14	Normal Distribution	57
3.14.1	Linear Transformation Property	58
3.14.2	Central Limit Theorem	61
3.15	Sum of a Random Number of Random Variables	62
3.16	Exercises	64
<b>4</b>	<b>Generating Random Variables for Simulation</b>	<b>70</b>
4.1	Inverse-Transform Method	70
4.1.1	The Continuous Case	70
4.1.2	The Discrete Case	72
4.2	Accept-Reject Method	72
4.2.1	Discrete Case	73
4.2.2	Continuous Case	75
4.2.3	Some Harder Problems	77
4.3	Readings	78
4.4	Exercises	78
<b>5</b>	<b>Sample Paths, Convergence, and Averages</b>	<b>79</b>
5.1	Convergence	79
5.2	Strong and Weak Laws of Large Numbers	83
5.3	Time Average versus Ensemble Average	84
5.3.1	Motivation	85
5.3.2	Definition	86
5.3.3	Interpretation	86
5.3.4	Equivalence	88
5.3.5	Simulation	90
5.3.6	Average Time in System	90
5.4	Related Readings	91
5.5	Exercise	91

### III The Predictive Power of Simple Operational Laws: “What-If” Questions and Answers

<b>6</b>	<b>Little’s Law and Other Operational Laws</b>	<b>95</b>
6.1	Little’s Law for Open Systems	95
6.2	Intuitions	96
6.3	Little’s Law for Closed Systems	96
6.4	Proof of Little’s Law for Open Systems	97
6.4.1	Statement via Time Averages	97
6.4.2	Proof	98
6.4.3	Corollaries	100
6.5	Proof of Little’s Law for Closed Systems	101
6.5.1	Statement via Time Averages	101
6.5.2	Proof	102
6.6	Generalized Little’s Law	102



## CONTENTS

ix

6.7	Examples Applying Little's Law	103
6.8	More Operational Laws: The Forced Flow Law	106
6.9	Combining Operational Laws	107
6.10	Device Demands	110
6.11	Readings and Further Topics Related to Little's Law	111
6.12	Exercises	111
<b>7</b>	<b>Modification Analysis: "What-If" for Closed Systems</b>	<b>114</b>
7.1	Review	114
7.2	Asymptotic Bounds for Closed Systems	115
7.3	Modification Analysis for Closed Systems	118
7.4	More Modification Analysis Examples	119
7.5	Comparison of Closed and Open Networks	122
7.6	Readings	122
7.7	Exercises	122

**IV From Markov Chains to Simple Queues**

<b>8</b>	<b>Discrete-Time Markov Chains</b>	<b>129</b>
8.1	Discrete-Time versus Continuous-Time Markov Chains	130
8.2	Definition of a DTMC	130
8.3	Examples of Finite-State DTMCs	131
8.3.1	Repair Facility Problem	131
8.3.2	Umbrella Problem	132
8.3.3	Program Analysis Problem	132
8.4	Powers of $\mathbf{P}$ : $n$ -Step Transition Probabilities	133
8.5	Stationary Equations	135
8.6	The Stationary Distribution Equals the Limiting Distribution	136
8.7	Examples of Solving Stationary Equations	138
8.7.1	Repair Facility Problem with Cost	138
8.7.2	Umbrella Problem	139
8.8	Infinite-State DTMCs	139
8.9	Infinite-State Stationarity Result	140
8.10	Solving Stationary Equations in Infinite-State DTMCs	142
8.11	Exercises	145
<b>9</b>	<b>Ergodicity Theory</b>	<b>148</b>
9.1	Ergodicity Questions	148
9.2	Finite-State DTMCs	149
9.2.1	Existence of the Limiting Distribution	149
9.2.2	Mean Time between Visits to a State	153
9.2.3	Time Averages	155
9.3	Infinite-State Markov Chains	155
9.3.1	Recurrent versus Transient	156
9.3.2	Infinite Random Walk Example	160
9.3.3	Positive Recurrent versus Null Recurrent	162
9.4	Ergodic Theorem of Markov Chains	164

9.5	Time Averages	166
9.6	Limiting Probabilities Interpreted as Rates	168
9.7	Time-Reversibility Theorem	170
9.8	When Chains Are Periodic or Not Irreducible	171
9.8.1	Periodic Chains	171
9.8.2	Chains that Are Not Irreducible	177
9.9	Conclusion	177
9.10	Proof of Ergodic Theorem of Markov Chains*	178
9.11	Exercises	183
<b>10</b>	<b>Real-World Examples: Google, Aloha, and Harder Chains*</b>	<b>190</b>
10.1	Google's PageRank Algorithm	190
10.1.1	Google's DTMC Algorithm	190
10.1.2	Problems with Real Web Graphs	192
10.1.3	Google's Solution to Dead Ends and Spider Traps	194
10.1.4	Evaluation of the PageRank Algorithm	195
10.1.5	Practical Implementation Considerations	195
10.2	Aloha Protocol Analysis	195
10.2.1	The Slotted Aloha Protocol	196
10.2.2	The Aloha Markov Chain	196
10.2.3	Properties of the Aloha Markov Chain	198
10.2.4	Improving the Aloha Protocol	199
10.3	Generating Functions for Harder Markov Chains	200
10.3.1	The z-Transform	201
10.3.2	Solving the Chain	201
10.4	Readings and Summary	203
10.5	Exercises	204
<b>11</b>	<b>Exponential Distribution and the Poisson Process</b>	<b>206</b>
11.1	Definition of the Exponential Distribution	206
11.2	Memoryless Property of the Exponential	207
11.3	Relating Exponential to Geometric via $\delta$ -Steps	209
11.4	More Properties of the Exponential	211
11.5	The Celebrated Poisson Process	213
11.6	Merging Independent Poisson Processes	218
11.7	Poisson Splitting	218
11.8	Uniformity	221
11.9	Exercises	222
<b>12</b>	<b>Transition to Continuous-Time Markov Chains</b>	<b>225</b>
12.1	Defining CTMCs	225
12.2	Solving CTMCs	229
12.3	Generalization and Interpretation	232
12.3.1	Interpreting the Balance Equations for the CTMC	234
12.3.2	Summary Theorem for CTMCs	234
12.4	Exercises	234

## CONTENTS

xi

<b>13 M/M/1 and PASTA</b>	<b>236</b>
13.1 The M/M/1 Queue	236
13.2 Examples Using an M/M/1 Queue	239
13.3 PASTA	242
13.4 Further Reading	245
13.5 Exercises	245
<b>V Server Farms and Networks: Multi-server, Multi-queue Systems</b>	
<b>14 Server Farms: M/M/k and M/M/k/k</b>	<b>253</b>
14.1 Time-Reversibility for CTMCs	253
14.2 M/M/k/k Loss System	255
14.3 M/M/k	258
14.4 Comparison of Three Server Organizations	263
14.5 Readings	264
14.6 Exercises	264
<b>15 Capacity Provisioning for Server Farms</b>	<b>269</b>
15.1 What Does Load Really Mean in an M/M/k?	269
15.2 The M/M/ $\infty$	271
15.2.1 Analysis of the M/M/ $\infty$	271
15.2.2 A First Cut at a Capacity Provisioning Rule for the M/M/k	272
15.3 Square-Root Staffing	274
15.4 Readings	276
15.5 Exercises	276
<b>16 Time-Reversibility and Burke's Theorem</b>	<b>282</b>
16.1 More Examples of Finite-State CTMCs	282
16.1.1 Networks with Finite Buffer Space	282
16.1.2 Batch System with M/M/2 I/O	284
16.2 The Reverse Chain	285
16.3 Burke's Theorem	288
16.4 An Alternative (Partial) Proof of Burke's Theorem	290
16.5 Application: Tandem Servers	291
16.6 General Acyclic Networks with Probabilistic Routing	293
16.7 Readings	294
16.8 Exercises	294
<b>17 Networks of Queues and Jackson Product Form</b>	<b>297</b>
17.1 Jackson Network Definition	297
17.2 The Arrival Process into Each Server	298
17.3 Solving the Jackson Network	300
17.4 The Local Balance Approach	301
17.5 Readings	306
17.6 Exercises	306
<b>18 Classed Network of Queues</b>	<b>311</b>
18.1 Overview	311
18.2 Motivation for Classed Networks	311

<b>18.3</b>	Notation and Modeling for Classed Jackson Networks	314
<b>18.4</b>	A Single-Server Classed Network	315
<b>18.5</b>	Product Form Theorems	317
<b>18.6</b>	Examples Using Classed Networks	322
<b>18.6.1</b>	Connection-Oriented ATM Network Example	322
<b>18.6.2</b>	Distribution of Job Classes Example	325
<b>18.6.3</b>	CPU-Bound and I/O-Bound Jobs Example	326
<b>18.7</b>	Readings	329
<b>18.8</b>	Exercises	329
<b>19</b>	<b>Closed Networks of Queues</b>	<b>331</b>
<b>19.1</b>	Motivation	331
<b>19.2</b>	Product-Form Solution	333
<b>19.2.1</b>	Local Balance Equations for Closed Networks	333
<b>19.2.2</b>	Example of Deriving Limiting Probabilities	335
<b>19.3</b>	Mean Value Analysis (MVA)	337
<b>19.3.1</b>	The Arrival Theorem	338
<b>19.3.2</b>	Iterative Derivation of Mean Response Time	340
<b>19.3.3</b>	An MVA Example	341
<b>19.4</b>	Readings	343
<b>19.5</b>	Exercises	343
<b>VI Real-World Workloads: High Variability and Heavy Tails</b>		
<b>20</b>	<b>Tales of Tails: A Case Study of Real-World Workloads</b>	<b>349</b>
<b>20.1</b>	Grad School Tales . . . Process Migration	349
<b>20.2</b>	UNIX Process Lifetime Measurements	350
<b>20.3</b>	Properties of the Pareto Distribution	352
<b>20.4</b>	The Bounded Pareto Distribution	353
<b>20.5</b>	Heavy Tails	354
<b>20.6</b>	The Benefits of Active Process Migration	354
<b>20.7</b>	Pareto Distributions Are Everywhere	355
<b>20.8</b>	Exercises	357
<b>21</b>	<b>Phase-Type Distributions and Matrix-Analytic Methods</b>	<b>359</b>
<b>21.1</b>	Representing General Distributions by Exponentials	359
<b>21.2</b>	Markov Chain Modeling of PH Workloads	364
<b>21.3</b>	The Matrix-Analytic Method	366
<b>21.4</b>	Analysis of Time-Varying Load	367
<b>21.4.1</b>	High-Level Ideas	367
<b>21.4.2</b>	The Generator Matrix, $\mathbf{Q}$	368
<b>21.4.3</b>	Solving for $\mathbf{R}$	370
<b>21.4.4</b>	Finding $\vec{\pi}_0$	371
<b>21.4.5</b>	Performance Metrics	372
<b>21.5</b>	More Complex Chains	372
<b>21.6</b>	Readings and Further Remarks	376
<b>21.7</b>	Exercises	376

## CONTENTS

xiii

<b>22</b>	<b>Networks with Time-Sharing (PS) Servers (BCMP)</b>	<b>380</b>
22.1	Review of Product-Form Networks	380
22.2	BCMP Result	380
22.2.1	Networks with FCFS Servers	381
22.2.2	Networks with PS Servers	382
22.3	M/M/1/PS	384
22.4	M/Cox/1/PS	385
22.5	Tandem Network of M/G/1/PS Servers	391
22.6	Network of PS Servers with Probabilistic Routing	393
22.7	Readings	394
22.8	Exercises	394
<b>23</b>	<b>The M/G/1 Queue and the Inspection Paradox</b>	<b>395</b>
23.1	The Inspection Paradox	395
23.2	The M/G/1 Queue and Its Analysis	396
23.3	Renewal-Reward Theory	399
23.4	Applying Renewal-Reward to Get Expected Excess	400
23.5	Back to the Inspection Paradox	402
23.6	Back to the M/G/1 Queue	403
23.7	Exercises	405
<b>24</b>	<b>Task Assignment Policies for Server Farms</b>	<b>408</b>
24.1	Task Assignment for FCFS Server Farms	410
24.2	Task Assignment for PS Server Farms	419
24.3	Optimal Server Farm Design	424
24.4	Readings and Further Follow-Up	428
24.5	Exercises	430
<b>25</b>	<b>Transform Analysis</b>	<b>433</b>
25.1	Definitions of Transforms and Some Examples	433
25.2	Getting Moments from Transforms: Peeling the Onion	436
25.3	Linearity of Transforms	439
25.4	Conditioning	441
25.5	Distribution of Response Time in an M/M/1	443
25.6	Combining Laplace and z-Transforms	444
25.7	More Results on Transforms	445
25.8	Readings	446
25.9	Exercises	446
<b>26</b>	<b>M/G/1 Transform Analysis</b>	<b>450</b>
26.1	The z-Transform of the Number in System	450
26.2	The Laplace Transform of Time in System	454
26.3	Readings	456
26.4	Exercises	456
<b>27</b>	<b>Power Optimization Application</b>	<b>457</b>
27.1	The Power Optimization Problem	457
27.2	Busy Period Analysis of M/G/1	459
27.3	M/G/1 with Setup Cost	462

27.4	Comparing ON/IDLE versus ON/OFF	465
27.5	Readings	467
27.6	Exercises	467
<b>VII Smart Scheduling in the M/G/1</b>		
<b>28</b>	<b>Performance Metrics</b>	<b>473</b>
28.1	Traditional Metrics	473
28.2	Commonly Used Metrics for Single Queues	474
28.3	Today's Trendy Metrics	474
28.4	Starvation/Fairness Metrics	475
28.5	Deriving Performance Metrics	476
28.6	Readings	477
<b>29</b>	<b>Scheduling: Non-Preemptive, Non-Size-Based Policies</b>	<b>478</b>
29.1	FCFS, LCFS, and RANDOM	478
29.2	Readings	481
29.3	Exercises	481
<b>30</b>	<b>Scheduling: Preemptive, Non-Size-Based Policies</b>	<b>482</b>
30.1	Processor-Sharing (PS)	482
30.1.1	Motivation behind PS	482
30.1.2	Ages of Jobs in the M/G/1/PS System	483
30.1.3	Response Time as a Function of Job Size	484
30.1.4	Intuition for PS Results	487
30.1.5	Implications of PS Results for Understanding FCFS	487
30.2	Preemptive-LCFS	488
30.3	FB Scheduling	490
30.4	Readings	495
30.5	Exercises	496
<b>31</b>	<b>Scheduling: Non-Preemptive, Size-Based Policies</b>	<b>499</b>
31.1	Priority Queueing	499
31.2	Non-Preemptive Priority	501
31.3	Shortest-Job-First (SJF)	504
31.4	The Problem with Non-Preemptive Policies	506
31.5	Exercises	507
<b>32</b>	<b>Scheduling: Preemptive, Size-Based Policies</b>	<b>508</b>
32.1	Motivation	508
32.2	Preemptive Priority Queueing	508
32.3	Preemptive-Shortest-Job-First (PSJF)	512
32.4	Transform Analysis of PSJF	514
32.5	Exercises	516
<b>33</b>	<b>Scheduling: SRPT and Fairness</b>	<b>518</b>
33.1	Shortest-Remaining-Processing-Time (SRPT)	518
33.2	Precise Derivation of SRPT Waiting Time*	521

Cambridge University Press

978-1-107-02750-3 - Performance Modeling and Design of Computer Systems: Queueing Theory in Action

Mor Harchol-Balter

Frontmatter

[More information](#)

## CONTENTS

xv

<b>33.3</b>	Comparisons with Other Policies	523
<b>33.3.1</b>	Comparison with PSJF	523
<b>33.3.2</b>	SRPT versus FB	523
<b>33.3.3</b>	Comparison of All Scheduling Policies	524
<b>33.4</b>	Fairness of SRPT	525
<b>33.5</b>	Readings	529
<i>Bibliography</i>		531
<i>Index</i>		541

Cambridge University Press

978-1-107-02750-3 - Performance Modeling and Design of Computer Systems: Queueing Theory in Action

Mor Harchol-Balter

Frontmatter

[More information](#)

---



# Preface

---

## The ad hoc World of Computer System Design

The design of computer systems is often viewed very much as an art rather than a science. Decisions about which scheduling policy to use, how many servers to run, what speed to operate each server at, and the like are often based on *intuitions* rather than mathematically derived formulas. Specific policies built into kernels are often riddled with secret “voodoo constants,”<sup>1</sup> which have no explanation but seem to “work well” under some benchmarked workloads. Computer systems students are often told to *first* build the system and *then* make changes to the policies to improve system performance, rather than first creating a formal model and design of the system on paper to ensure the system meets performance goals.

Even when trying to evaluate the performance of an *existing* computer system, students are encouraged to *simulate* the system and spend many days running their simulation under different workloads waiting to see what happens. Given that the search space of possible workloads and input parameters is often huge, vast numbers of simulations are needed to properly cover the space. Despite this fact, mathematical models of the system are rarely created, and we rarely characterize workloads stochastically. There is no formal analysis of the parameter space under which the computer system is likely to perform well versus that under which it is likely to perform poorly. It is no wonder that computer systems students are left feeling that the whole process of system evaluation and design is very ad hoc. As an example, consider the trial-and-error approach to updating resource scheduling in the many versions of the Linux kernel.

## Analytical Modeling for Computer Systems

But it does not have to be this way! These same systems designers could mathematically model the system, stochastically characterize the workloads and performance goals, and then analytically derive the performance of the system as a function of workload and input parameters. The fields of *analytical modeling* and *stochastic processes* have existed for close to a century, and they can be used to save systems designers huge numbers of hours in trial and error while improving performance. Analytical modeling can also be used in conjunction with simulation to help guide the simulation, reducing the number of cases that need to be explored.

---

<sup>1</sup> The term “voodoo constants” was coined by Prof. John Ousterhout during his lectures at the University of California, Berkeley.

Unfortunately, of the hundreds of books written on stochastic processes, almost none deal with computer systems. The examples in those books and the material covered are oriented toward operations research areas such as manufacturing systems, or *human* operators answering calls in a call center, or some assembly-line system with different priority jobs.

In many ways the analysis used in designing manufacturing systems is not all that different from computer systems. There are many parallels between a human operator and a computer server: There are faster human operators and slower ones (just as computer servers); the human servers sometimes get sick (just as computer servers sometimes break down); when not needed, human operators can be sent home to save money (just as computer servers can be turned off to save power); there is a startup overhead to bringing back a human operator (just as there is a warmup cost to turning on a computer server); and the list goes on.

However, there are also many differences between manufacturing systems and computer systems. To start, computer systems workloads have been shown to have extremely high variability in job sizes (service requirements), with squared coefficients of variation upward of 100. This is very different from the low-variability service times characteristic of job sizes in manufacturing workloads. This difference in variability can result in performance differences of orders of magnitude. Second, computer workloads are typically preemptible, and time-sharing (Processor-Sharing) of the CPU is extremely common. By contrast, most manufacturing workloads are non-preemptive (first-come-first-serve service order is the most common). Thus most books on stochastic processes and queueing omit chapters on Processor-Sharing or more advanced preemptive policies like Shortest-Remaining-Processing-Time, which are very much at the heart of computer systems. Processor-Sharing is particularly relevant when analyzing server farms, which, in the case of computer systems, are typically composed of Processor-Sharing servers, not First-Come-First-Served ones. It is also relevant in any computing application involving bandwidth being shared between users, which typically happens in a processor-sharing style, not first-come-first-serve order. Performance metrics may also be different for computer systems as compared with manufacturing systems (e.g., power usage, an important metric for computer systems, is not mentioned in stochastic processes books). Closed-loop architectures, in which new jobs are not created until existing jobs complete, and where the performance goal is to maximize throughput, are very common in computer systems, but are often left out of queueing books. Finally, the particular types of interactions that occur in disks, networking protocols, databases, memory controllers, and other computer systems are very different from what has been analyzed in traditional queueing books.

### The Goal of This Book

Many times I have walked into a fellow computer scientist's office and was pleased to find a queueing book on his shelf. Unfortunately, when questioned, my colleague was quick to answer that he never uses the book because "The world doesn't look like an M/M/1 queue, and I can't understand anything past that chapter." The problem is that

the queueing theory books are not “friendly” to computer scientists. The applications are not computer-oriented, and the assumptions used are often unrealistic for computer systems. Furthermore, these books are abstruse and often impenetrable by anyone who has not studied graduate-level mathematics. In some sense this is hard to avoid: If one wants to do more than provide readers with formulas to “plug into,” then one has to *teach* them to derive their own formulas, and this requires learning a good deal of math. Fortunately, as one of my favorite authors, Sheldon Ross, has shown, it *is* possible to teach a lot of stochastic analysis in a fun and simple way that does not require first taking classes in measure theory and real analysis.

My motive in writing this book is to improve the design of computer systems by introducing computer scientists to the powerful world of queueing-theoretic modeling and analysis. Personally, I have found queueing-theoretic analysis to be extremely valuable in much of my research including: designing routing protocols for networks, designing better scheduling algorithms for web servers and database management systems, disk scheduling, memory-bank allocation, supercomputing resource scheduling, and power management and capacity provisioning in data centers. Content-wise, I have two goals for the book. First, I want to provide enough applications from computer systems to make the book relevant and interesting to computer scientists. Toward this end, almost half the chapters of the book are “application” chapters. Second, I want to make the book mathematically rich enough to give readers the ability to actually *develop new queueing analysis*, not just apply existing analysis. As computer systems and their workloads continue to evolve and become more complex, it is unrealistic to assume that they can be modeled with known queueing frameworks and analyses. As a designer of computer systems myself, I am constantly finding that I have to invent new queueing concepts to model aspects of computer systems.

## How This Book Came to Be

In 1998, as a postdoc at MIT, I developed and taught a new computer science class, which I called “Performance Analysis and Design of Computer Systems.” The class had the following description:

In designing computer systems one is usually constrained by certain performance goals (e.g., low response time or high throughput or low energy). On the other hand, one often has many choices: One fast disk, or two slow ones? What speed CPU will suffice? Should we invest our money in more buffer space or a faster processor? How should jobs be scheduled by the processor? Does it pay to migrate active jobs? Which routing policy will work best? Should one balance load among servers? How can we best combat high-variability workloads? Often answers to these questions are counterintuitive. Ideally, one would like to have answers to these questions before investing the time and money to build a system. This class will introduce students to analytic stochastic modeling, which allows system designers to answer questions such as those above.

Since then, I have further developed the class via 10 more iterations taught within the School of Computer Science at Carnegie Mellon, where I taught versions of the

class to both PhD students and advanced undergraduates in the areas of computer science, engineering, mathematics, and operations research. In 2002, the Operations Management department within the Tepper School of Business at Carnegie Mellon made the class a qualifier requirement for all operations management students.

As other faculty, including my own former PhD students, adopted my lecture notes in teaching their own classes, I was frequently asked to turn the notes into a book. This is “version 1” of that book.

## Outline of the Book

This book is written in a question/answer style, which mimics the Socratic style that I use in teaching. I believe that a class “lecture” should ideally be a long sequence of bite-sized questions, which students can easily provide answers to and which lead students to the right intuitions. In reading this book, it is extremely important to try to answer each question *without* looking at the answer that follows the question. The questions are written to remind the reader to “think” rather than just “read,” and to remind the teacher to ask questions rather than just state facts.

There are exercises at the end of each chapter. The exercises are an integral part of the book and should not be skipped. Many exercises are used to illustrate the application of the theory to problems in computer systems design, typically with the purpose of illuminating a key insight. All exercises are related to the material covered in the chapter, with early exercises being straightforward applications of the material and later exercises exploring extensions of the material involving greater difficulty.

The book is divided into seven parts, which mostly build on each other.

**Part I** introduces queueing theory and provides motivating examples from computer systems design that can be answered using basic queueing analysis. Basic queueing terminology is introduced including closed and open queueing models and performance metrics.

**Part II** is a probability refresher. To make this book self-contained, we have included in these chapters all the probability that will be needed throughout the rest of the book. This includes a summary of common discrete and continuous random variables, their moments, and conditional expectations and probabilities. Also included is some material on generating random variables for simulation. Finally we end with a discussion of sample paths, convergence of sequences of random variables, and time averages versus ensemble averages.

**Part III** is about operational laws, or “back of the envelope” analysis. These are very simple laws that hold for all well-behaved queueing systems. In particular, they do not require that any assumptions be made about the arrival process or workload (like Poisson arrivals or Exponential service times). These laws allow us to quickly reason at a high level (averages only) about system behavior and make design decisions regarding what modifications will have the biggest performance impact. Applications to high-level computer system design are provided throughout.

**Part IV** is about Markov chains and their application toward stochastic analysis of computer systems. Markov chains allow a much more detailed analysis of systems by representing the full space of possible states that the system can be in. Whereas the operational laws in Part III often allow us to answer questions about the overall mean number of jobs in a system, Markov chains allow us to derive the probability of exactly  $i$  jobs being queued at server  $j$  of a multi-server system. Part IV includes both discrete-time and continuous-time Markov chains. Applications include Google's PageRank algorithm, the Aloha (Ethernet) networking protocol, and an analysis of dropping probabilities in finite-buffer routers.

**Part V** develops the Markov chain theory introduced in Part IV to allow the analysis of more complex networks, including server farms. We analyze networks of queues with complex routing rules, where jobs can be associated with a "class" that determines their route through the network (these are known as BCMP networks). Part V also derives theorems on capacity provisioning of server farms, such as the "square-root staffing rule," which determines the minimum number of servers needed to provide certain delay guarantees.

The fact that Parts IV and V are based on Markov chains necessitates that certain "Markovian" (memoryless) assumptions are made in the analysis. In particular, it is assumed that the service requirements (sizes) of jobs follow an Exponential distribution and that the times between job arrivals are also Exponentially distributed. Many applications are reasonably well modeled via these Exponential assumptions, allowing us to use Markov analysis to get good insights into system performance. However, in some cases, it is important to capture the high-variability job size distributions or correlations present in the empirical workloads.

**Part VI** introduces techniques that allow us to replace these Exponential distributions with high-variability distributions. Phase-type distributions are introduced, which allow us to model virtually any general distribution by a *mixture of Exponentials*, leveraging our understanding of Exponential distributions and Markov chains from Parts IV and V. Matrix-analytic techniques are then developed to analyze systems with phase-type workloads in both the arrival process and service process. The M/G/1 queue is introduced, and notions such as the Inspection Paradox are discussed. Real-world workloads are described including heavy-tailed distributions. Transform techniques are also introduced that facilitate working with general distributions. Finally, even the service order at the queues is generalized from simple first-come-first-served service order to time-sharing (Processor-Sharing) service order, which is more common in computer systems. Applications abound: Resource allocation (task assignment) in server farms with high-variability job sizes is studied extensively, both for server farms with non-preemptive workloads and for web server farms with time-sharing servers. Power management policies for single servers and for data centers are also studied.

**Part VII**, the final part of the book, is devoted to scheduling. Smart scheduling is extremely important in computer systems, because it can dramatically improve system performance without requiring the purchase of any new hardware. Scheduling is at the heart of operating systems, bandwidth allocation in networks, disks, databases, memory hierarchies, and the like. Much of the research being done in the computer systems

area today involves the design and adoption of new scheduling policies. Scheduling can be counterintuitive, however, and the analysis of even basic scheduling policies is far from simple. Scheduling policies are typically evaluated via simulation. In introducing the reader to analytical techniques for evaluating scheduling policies, our hope is that more such policies might be evaluated via analysis.

We expect readers to mostly work through the chapters in order, with the following exceptions: First, any chapter or section marked with a star (\*) can be skipped without disturbing the flow. Second, the chapter on transforms, Chapter 25, is purposely moved to the end, so that most of the book does not depend on knowing transform analysis. However, because learning transform analysis takes some time, we recommend that any teacher who plans to cover transforms introduce the topic a little at a time, starting early in the course. To facilitate this, we have included a large number of exercises at the end of Chapter 25 that do not require material in later chapters and can be assigned earlier in the course to give students practice manipulating transforms.

Finally, we urge readers to please check the following websites for new errors/software:

<http://www.cs.cmu.edu/~harchol/PerformanceModeling/errata.html>

<http://www.cs.cmu.edu/~harchol/PerformanceModeling/software.html>

Please send any additional errors to [harchol@cs.cmu.edu](mailto:harchol@cs.cmu.edu).

## Acknowledgments

---

Writing a book, I quickly realized, is very different from writing a research paper, even a very long one. Book writing actually bears much more similarity to teaching a class. That is why I would like to start by thanking the three people who most influenced my teaching. Manuel Blum, my PhD advisor, taught me the art of creating a lecture out of a series of bite-sized questions. Dick Karp taught me that you can cover an almost infinite amount of material in just one lecture if you spend enough time in advance simplifying that material into its cleanest form. Sheldon Ross inspired me by the depth of his knowledge in stochastic processes (a knowledge so deep that he never once looked at his notes while teaching) and by the sheer clarity and elegance of both his lectures and his many beautifully written books.

I would also like to thank Carnegie Mellon University, and the School of Computer Science at Carnegie Mellon, which has at its core the theme of interdisciplinary research, particularly the mixing of theoretical and applied research. CMU has been the perfect environment for me to develop the analytical techniques in this book, all in the context of solving hard applied problems in computer systems design. CMU has also provided me with a never-ending stream of gifted students, who have inspired many of the exercises and discussions in this book. Much of this book came from the research of my own PhD students, including Sherwin Doroudi, Anshul Gandhi, Varun Gupta, Yoongu Kim, David McWherter, Takayuki Osogami, Bianca Schroeder, Adam Wierman, and Timothy Zhu. In addition, Mark Crovella, Mike Kozuch, and particularly Alan Scheller-Wolf, all longtime collaborators of mine, have inspired much of my thinking via their uncanny intuitions and insights.

A great many people have proofread parts of this book or tested out the book and provided me with useful feedback. These include Sem Borst, Doug Down, Erhun Ozkan, Katsunobu Sasanuma, Alan Scheller-Wolf, Thrasyvoulos Spyropoulos, Jarod Wang, and Zachary Young. I would also like to thank my editors, Diana Gillooly and Lauren Cowles from Cambridge University Press, who were very quick to answer my endless questions, and who greatly improved the presentation of this book. Finally, I am very grateful to Miso Kim, my illustrator, a PhD student at the Carnegie Mellon School of Design, who spent hundreds of hours designing all the fun figures in the book.

On a more personal note, I would like to thank my mother, Irit Harchol, for making my priorities her priorities, allowing me to maximize my achievements. I did not know what this meant until I had a child of my own. Lastly, I would like to thank my husband, Andrew Young. He won me over by reading all my online lecture notes and doing every homework problem – this was his way of asking me for a first date. His ability to understand it all without attending any lectures made me believe that my lecture notes might actually “work” as a book. His willingness to sit by my side every night for many months gave me the motivation to make it happen.