

COMPUTER AND COMMUNICATION SCIENCES

PERFORMANCE EVALUATION OF COMPUTER AND COMMUNICATION SYSTEMS

Jean-Yves Le Boudec



EPFL Press
Distributed by CRC Press

COMPUTER AND COMMUNICATION SCIENCES

PERFORMANCE EVALUATION
OF COMPUTER AND
COMMUNICATION SYSTEMS

Jean-Yves Le Boudec



EPFL Press

A Swiss academic publisher distributed by CRC Press



Taylor and Francis Group, LLC
6000 Broken Sound Parkway, NW, Suite 300,
Boca Raton, FL 33487

Distribution and Customer Service
orders@crcpress.com

www.crcpress.com

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress.

This book is published under the editorial direction of
Professor Serge Vaudenay (EPFL).

The authors and publisher express their thanks to the Ecole polytechnique fédérale
de Lausanne (EPFL) for its generous support towards the publication of this book.

Cover artwork: Elias Le Boudec

EPFL Press

is an imprint owned by Presses polytechniques et universitaires romandes, a Swiss
academic publishing company whose main purpose is to publish the teaching and
research works of the Ecole polytechnique fédérale de Lausanne (EPFL) and other
universities and institutions of higher learning.

Presses polytechniques et universitaires romandes
EPFL – Rolex Learning Center
Post office box 119
CH-1015 Lausanne, Switzerland
E-mail: ppur@epfl.ch
Phone: 021/693 21 30
Fax: 021/693 40 27

www.epflpress.org

© 2010, First edition, EPFL Press, Lausanne (Switzerland)
ISBN 978-2-940222-40-7 (EPFL Press)
ISBN 978-1-4398-4992-7 (CRC Press)

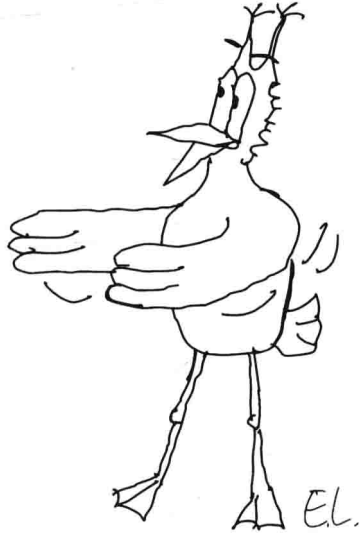
Printed in Italy

All right reserved (including those of translation into other languages). No part of
this book may be reproduced in any form – by photoprint, microfilm, or any other
means – nor transmitted or translated into a machine language without written
permission from the publisher.

PERFORMANCE EVALUATION
OF COMPUTER AND
COMMUNICATION SYSTEMS

Preface

Performance evaluation is often the critical part of evaluating the results of a research project. Many of us are familiar with simulations, but it is frequently difficult to address questions like: Should I eliminate the beginning of the simulation in order for the system to become stabilized? I simulate a random way-point model but the average speed in my simulation is not as expected. What happened? The reviewers of my study complained that I did not provide confidence intervals. How do I go about this? I would like to characterize the fairness of my protocol. Should I use Jain's Fairness Index or the Lorenz Curve Gap? I would like to fit a distribution to the flow sizes that I measured, but all my measurements are truncated to a maximum value; how do I account for the truncation?



This book groups a set of lecture notes for a course given at EPFL. It contains all the material needed by an engineer who wishes to evaluate the performance of a computer or communication system. More precisely, with this book and some accompanying practicals, you will be able to answer the above and other questions, evaluate the performance of computer and communication systems and master the theoretical foundations of performance evaluation and of the corresponding software packages.

In the past, many textbooks on performance evaluation have given the impression that this is a complex field, with large amounts of baroque queuing theory excursions, which can be exercised only by performance evaluation experts. This is not necessarily the case. In contrast, performance evaluation can and should be performed by any computer engineering specialist who designs a system. When a plumber installs pipes in our house, one expects her to properly size their diameters; the same holds for computer engineers.

This book is not intended for the performance evaluation specialist. It is addressed to *any computer engineer or scientist* who is active in the development or operation of software or hardware systems. The required background is an elementary course in probability and one in calculus.

The objective of this book is therefore to make performance evaluation usable by all computer engineers and scientists. The foundations of performance

evaluation reside in statistics and queuing theory. Therefore, *some* mathematics are involved and the text cannot be overly simplified. However, it turns out that much of the complications are not in the general theories, but in the exact solution of specific models. For example, certain textbooks on statistics (but none of the ones cited in the reference list) develop various solution techniques for specific models, the vast majority of which are encapsulated in commercially or freely available software packages like Matlab, S-PLUS, Excel, Scilab or R.

To avoid this pitfall, we focus first on the *what* before the *how*. Indeed, the most difficult question in a performance analysis is often “what to do”; once you know what to do, it is less difficult to find a way with your usual software tools or by shopping the web. For example, what do we do when we fit a model to data using least square fitting (Chapter 3)? What is a confidence interval? What is a prediction interval (Chapter 2)? What is the congestion collapse pattern (Chapter 1)? What is the null hypothesis in a test and what does the result of a test *really* mean (Chapter 4)? What is an information criterion (Chapter 5)? If no failure appears out of n experiments, what confidence interval can I give for the failure probability (Chapter 2)?

Second, regarding the *how*, we looked for solution methods that are as universal as possible, i.e. that apply to many situations, whether simple or complex. There are several reasons for this. Firstly, one should use only methods and tools that one understands, and a good engineer should first invest some time in learning tools and methods that he/she will use more often. Secondly, brute force and a computer can do a lot more than one often seems to believe. This philosophy is in sharp contrast to some publications on performance evaluation. For example, computing confidence or prediction intervals can be made simple and systematic if we use the median and not the mean; if we have to employ the mean, the use of the likelihood ratio statistic is quite universal and requires little intellectual sophistication regarding the model. Thus, we focus on generic methods such as: the use of filters for forecasting (Chapter 5), bootstrap and Monte-Carlo simulations for evaluating averages or prediction intervals (Chapter 6), the likelihood ratio statistic for tests (Chapter 2, Chapter 4), importance sampling (Chapter 6), least-square and ℓ^1 -norm minimization methods (Chapter 3).

When presenting solutions, we try *not* to hide their limitations and the cases where they do not work. Indeed, some frustrations experienced by young researchers can sometimes be attributed to false expectations about the power of various methods.

We give a coverage of queuing theory that attempts to strike a balance between depth and relevance. During a performance analysis, one is often confronted with the dilemma: should we use an approximate model for which exact solutions exist, or approximate solutions for a more exact model? We propose four topics (deterministic analysis, operational laws, single queues, queuing networks) which provide a good balance. We illustrate in a case study how the four topics can be utilized to provide different insights on a queuing question. For queuing networks, we give a unified treatment, which is perhaps the first of its kind at this level of synthesis. We show that complex topics such as

queues with concurrency (MSCCC queues) or networks with bandwidth sharing (Whittle networks) all fit in the same framework of product form queuing networks. Results of this kind have been traditionally presented as separate; unifying them simplifies the student's job and provides new insights.

We develop the topic of Palm calculus, also called "the importance of the viewpoint", which is so central to queuing theory, as a topic of its own. Indeed, this topic has so many applications to simulation and to system analysis in general, that it is a very good time investment. Here too, we focus on general purpose methods and results, in particular the large-time heuristic for mapping various viewpoints (Chapter 7).

Chapter 1 gives a methodology and serves as an introduction to the rest of the book. Performance patterns are also described, i.e. facts that repeatedly appear in various situations, and the knowledge of which considerably helps the performance evaluation.

Chapter 2 demonstrates how to summarize experimental or simulation results, as well as how to quantify their accuracy. It also serves as an introduction to a scientific use of the statistical method, i.e. pose a model and verify its assumptions. In Chapter 3, we present general methods for fitting an explanatory model to data and the concept of heavy tail. Chapter 4 describes the techniques of tests, and Chapter 5 those of forecasting. These four chapters give a coverage of modern statistics useful to our field.

Chapter 6 discusses discrete event simulation and several important, though simple issues such as the need for transient removal, for confidence intervals, and classical simulation techniques. We also discuss importance sampling, which is very useful for computing estimates of rare events; we give a simple, though quite general and broadly applicable method.

Chapter 7 describes Palm calculus, which relates the varying viewpoints resulting from measurements done by different operators. Here, we discuss freezing simulations, a phenomenon which can be a problem for even simple simulations if one is not aware of it. We also present how to perform a perfect simulation of stochastic recurrences. Chapter 8 discusses patterns specific to queuing, classical solution methods for queuing networks, and, perhaps more important, operational analysis for rapid evaluation.

The appendix gives background information that cannot yet be easily found elsewhere, such as a Fourier-free quick crash course on digital filters (used in Chapter 5) and confidence intervals for quantiles.

Performance evaluation is primarily an art, and involves using sophisticated tools such as mathematical packages, measurement tools and simulation tools. See the web site of the EPFL lecture on Performance Evaluation for some examples of *practicals*, implemented in Matlab and designed around this book.

The text is intended for self-study. Proofs are not given when there are easily accessible references (these are indicated in the text); otherwise they can be found in appendixes at the end of the chapters.

The *Index* collects all terms and expressions that are highlighted in the text like *this* and also serves as a notation list.

Acknowledgements

I would like to thank Anthony Davison for allowing me to access a beta version of his book “Statistical Models” as well as Richard Weber, who made his lecture notes freely available on the web and allowed me to use them as constituent material in an early version of this course. I am grateful to François Baccelli and Pierre Brémaud who helped me obtain some understanding of their fields. Many thanks go to Mourad Kara for discussions and input, to Irina Baltcheva, Manuel Flury, Olivier Gallay, Assane Gueye, Paul Hurley, Ruben Merz, Božidar Radunović, Gianluca Rizzo, Slaviša Sarafijanović, Milan Vojnović, Utkarsh Upadhyay and Jonas Wagner for various input and comments. I thank Scouac, Pilou and their friends for visiting our pages here and there. Last but not least, I thank Elias for the artwork.

Jean-Yves Le Boudec, EPFL

Contents

Preface	v
CHAPTER 1 Methodology	1
1.1 What is Performance Evaluation?	1
1.1.1 Load	2
1.1.2 Metric	2
1.1.3 The Different Goals of Performance Evaluation . . .	4
1.2 Factors	5
1.2.1 The Hidden Factor Paradox	6
1.2.2 Simpson's Paradox	7
1.3 Evaluation Methods	9
1.4 The Scientific Method	9
1.5 Performance Patterns	12
1.5.1 Bottlenecks	12
1.5.2 Congestion Collapse	13
1.5.3 Competition Side Effect	15
1.5.4 Latent Congestion Collapse	17
1.6 Review	19
1.6.1 Check-List	19
1.6.2 Review Questions	20
CHAPTER 2 Summarizing Performance Data and Confidence Intervals	23
2.1 Summarized Performance Data	24
2.1.1 Histograms and Empirical CDF	24
2.1.2 Means, Medians and Quantiles	25
2.1.3 Coefficient of Variation and Lorenz Curve Gap . . .	27
2.1.4 Fairness Indices	28
2.2 Confidence Intervals	34
2.2.1 What is a Confidence Interval?	34
2.2.2 Confidence Interval for the Median and Other Quantiles	35
2.2.3 Confidence Interval for the Mean	37
2.2.4 Confidence Intervals for Fairness Indices and the Bootstrap Method	40
2.2.5 Confidence Interval for Success Probability	42

2.3	The Independence Assumption	44
2.3.1	What does iid mean?	44
2.3.2	How do I know in Practice if the iid Assumption is Valid?	45
2.3.3	What Happens if the iid Assumption does not hold?	47
2.4	Prediction Interval	50
2.4.1	Prediction for an iid Sample based on Order Statistic	51
2.4.2	Prediction for a Normal iid Sample	52
2.4.3	The Normal Assumption	54
2.5	Which Summarization to use?	56
2.5.1	Robustness	57
2.5.2	Compactness	61
2.6	Other Aspects of Confidence Prediction Intervals	61
2.6.1	Intersection of Confidence, Prediction Intervals	61
2.6.2	The Meaning of Confidence	62
2.7	Proofs	62
2.8	Review	64
2.8.1	Summary	64
2.8.2	Review Questions	65
CHAPTER 3	Model Fitting	67
3.1	Model Fitting Criteria	68
3.1.1	What is Model Fitting?	68
3.1.2	Least Squares Correspond to Gaussian, Equal Variance	71
3.1.3	ℓ^1 Norm Minimization Corresponds to Laplace Noise	72
3.2	Linear Regression	75
3.3	Linear Regression with ℓ^1 Norm Minimization	79
3.4	Choosing a Distribution	82
3.4.1	Shape	83
3.4.2	Skewness and Kurtosis	86
3.4.3	Power Laws, Pareto Distribution and Zipf's Law	87
3.4.4	Hazard Rate	89
3.4.5	Fitting a Distribution	91
3.4.6	Censored Data	91
3.4.7	Combinations of Distributions	94
3.5	Heavy Tail	96
3.5.1	Definition	96
3.5.2	Heavy Tail and Stable Distributions	97
3.5.3	Heavy Tail in Practice	99
3.5.4	Testing for a Heavy Tail	101
3.5.5	Application Example: The Workload Generator SURGE	103

3.6	Proofs	105
3.7	Review	106
3.7.1	Review Questions	106
3.7.2	Useful Matlab Commands	106
CHAPTER 4	Tests	107
4.1	The Neyman Pearson Framework	108
4.1.1	The Null Hypothesis and the Alternative	108
4.1.2	Critical Region, Size and Power	110
4.1.3	p -value of a Test	113
4.1.4	Tests are just Tests	113
4.2	Likelihood Ratio Tests	114
4.2.1	Definition of Likelihood Ratio Tests	114
4.2.2	Student Test for Single Sample (or Paired Data)	116
4.2.3	The Simple Goodness of Fit Test	117
4.3	ANOVA	119
4.3.1	Analysis of Variance (ANOVA) and F -tests	119
4.3.2	Testing for a Common Variance	124
4.4	Asymptotic Results	126
4.4.1	Likelihood Ratio Statistic	126
4.4.2	Pearson Chi-squared Statistic and Goodness of Fit	126
4.4.3	Test of Independence	130
4.5	Other Tests	131
4.5.1	Goodness of Fit Tests based on Ad-Hoc Pivots	131
4.5.2	Robust Tests	134
4.6	Proofs	137
4.7	Review	139
4.7.1	Tests are just Tests	139
4.7.2	Review Questions	139
CHAPTER 5	Forecasting	141
5.1	What is Forecasting?	142
5.2	Linear Regression	142
5.3	The Overfitting Problem	145
5.3.1	Use of Test Data	147
5.3.2	Information Criterion	148
5.4	Differencing the Data	149
5.4.1	Differencing and De-seasonalizing Filters	149
5.4.2	Computing Point Prediction	152
5.4.3	Computing Prediction Intervals	153
5.5	Fitting Differenced Data to an ARMA Model	155
5.5.1	Stationary but non iid Differenced Data	155
5.5.2	ARMA and ARIMA Processes	156
5.5.3	Fitting an ARMA Model	160
5.5.4	Forecasting	162

5.6	Sparse ARMA and ARIMA Models	167
5.6.1	Constrained ARMA Models	167
5.6.2	Holt-Winters Models	168
5.7	Proofs	173
CHAPTER 6	Discrete Event Simulation	175
6.1	What is a Simulation?	176
6.1.1	Simulated Time and Real Time	176
6.1.2	Simulation Types	176
6.2	Simulation Techniques	179
6.2.1	Discrete Event Simulation	179
6.2.2	Stochastic Recurrence	183
6.3	Computing the Accuracy of Stochastic Simulations	186
6.3.1	Independent Replications	186
6.3.2	Computing Confidence Intervals	186
6.3.3	Non-Terminating Simulations	188
6.4	Monte Carlo Simulation	188
6.5	Random Number Generators	191
6.6	How to Sample from a Distribution	194
6.6.1	CDF Inversion	194
6.6.2	Rejection Sampling	196
6.6.3	Ad-Hoc Methods	200
6.7	Importance Sampling	201
6.7.1	Motivation	201
6.7.2	The Importance Sampling Framework	202
6.7.3	Selecting An Importance Sampling Distribution	205
6.8	Proofs	209
6.9	Review	211
CHAPTER 7	Palm Calculus, or the Importance of the Viewpoint	213
7.1	An Informal Introduction	214
7.1.1	Event-versus-Time Averages	214
7.1.2	The Large Time Heuristic	216
7.1.3	Two Event Clocks	218
7.1.4	Arbitrary Sampling Methods	220
7.2	Palm Calculus	223
7.2.1	Hypotheses	223
7.2.2	Definitions	223
7.2.3	Interpretation as Time and Event Averages	226
7.2.4	The Inversion and Intensity Formulas	227
7.3	Other Useful Palm Calculus Results	230
7.3.1	Residual Time and Feller's Paradox	230
7.3.2	The Rate Conservation Law and Little's Formula	233
7.3.3	Two Event Clocks	240

7.4	Simulation Defined as Stochastic Recurrence	242
7.4.1	Stochastic Recurrence, Modulated Process	242
7.4.2	Freezing Simulations	243
7.4.3	Perfect Simulation of Stochastic Recurrence	245
7.5	Application to Markov Chain Models and the PASTA Property	250
7.5.1	Embedded Sub-Chain	250
7.5.2	PASTA	252
7.6	Appendix: Quick Review of Markov Chains	254
7.6.1	Markov Chain in Discrete Time	254
7.6.2	Markov Chain in Continuous Time	256
7.6.3	Poisson and Bernoulli	257
7.7	Proofs	258
7.8	Review Questions	262

CHAPTER 8 Queuing Theory for Those who cannot Wait 263

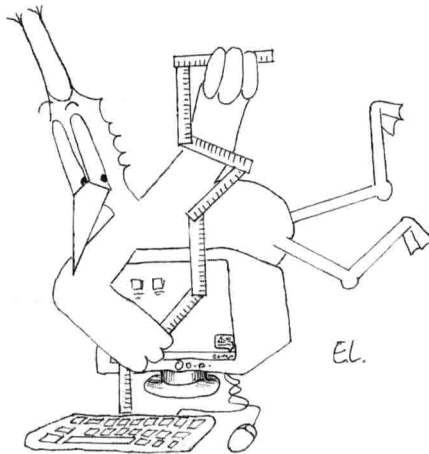
8.1	Deterministic Analysis	264
8.1.1	Description of a Queuing System with Cumulative Functions	264
8.1.2	Reich's Formula	266
8.2	Operational Laws for Queuing Systems	268
8.2.1	Departures and Arrivals See Same Averages (DASSA)	268
8.2.2	Little's Law and Applications	269
8.2.3	Networks and Forced Flows	270
8.2.4	Bottleneck Analysis	271
8.3	Classical Results for a Single Queue	273
8.3.1	Kendall's Notation	273
8.3.2	The Single Server Queue	274
8.3.3	The Processor Sharing Queue, $M/GI/1/PS$	279
8.3.4	Single Queue with B Servers	281
8.4	Definitions for Queuing Networks	283
8.4.1	Classes, Chains and Markov Routing	283
8.4.2	Catalog of Service Stations	285
8.4.3	The Station Function	293
8.5	The Product-Form Theorem	299
8.5.1	Product Form	299
8.5.2	Stability Conditions	300
8.6	Computational Aspects	302
8.6.1	Convolution	303
8.6.2	Throughput	303
8.6.3	Equivalent Service Rate	306
8.6.4	Suppression of Open Chains	309
8.6.5	Arrival Theorem and MVA Version 1	311
8.6.6	Network Decomposition	314
8.6.7	MVA Version 2	319

8.7	What This Tells Us	321
8.7.1	Insensitivity	321
8.7.2	The Importance of Modeling Closed Populations	323
8.8	Mathematical Details about Product-Form Queuing Networks	325
8.8.1	Phase-Type Distributions	325
8.8.2	Micro and Macro States	326
8.8.3	Micro to Macro: Aggregation Condition	328
8.8.4	Local Balance in Isolation	328
8.8.5	The Product Form Theorem	329
8.8.6	Networks with Blocking	330
8.9	Case Study	331
8.9.1	Deterministic Analysis	332
8.9.2	Single Queue Analysis	333
8.9.3	Operational Analysis	334
8.9.4	Queuing Network Analysis	335
8.9.5	Conclusions	338
8.10	Proofs	338
8.11	Review	342
8.11.1	Review Questions	342
8.11.2	Summary of Notation	343
ANNEX	A Tables	345
ANNEX	B Parametric Estimation, Large Sample Theory	351
B.1	Parametric Estimation Theory	351
B.1.1	The Parametric Estimation Framework	351
B.1.2	Maximum Likelihood Estimator (MLE)	352
B.1.3	Efficiency and Fisher Information	353
B.2	Asymptotic Confidence Intervals	354
B.3	Confidence Interval in Presence of Nuisance Parameters	358
ANNEX	C Gaussian Random Vectors in \mathbb{R}^n	363
C.1	Notation and a Few Results of Linear Algebra	363
C.1.1	Notation	363
C.1.2	Linear Algebra	364
C.2	Covariance Matrix of a Random Vector in \mathbb{R}^n	364
C.2.1	Definitions	364
C.2.2	Properties of Covariance Matrix	365
C.2.3	Choleski's Factorization	366
C.2.4	Degrees of Freedom	366
C.3	Gaussian Random Vector	367
C.3.1	Definition and Main Properties	367
C.3.2	Diagonal Form	368

	C.4	Foundations of ANOVA	369
	C.4.1	Homoscedastic Gaussian Vector	369
	C.4.2	Maximum Likelihood Estimation for Homoscedastic Gaussian Vectors	370
	C.5	Conditional Gaussian Distribution	371
	C.5.1	Schur Complement	371
	C.5.2	Distribution of \vec{X}_1 given \vec{X}_2	371
	C.5.3	Partial Correlation	372
	C.6	Proofs	373
ANNEX	D	Digital Filters	377
	D.1	Calculus of Digital Filters	377
	D.1.1	Backshift Operator	377
	D.1.2	Filters	378
	D.1.3	Impulse response and Dirac Sequence	379
	D.1.4	Composition of Filters, Commutativity	379
	D.1.5	Inverse of Filter	380
	D.1.6	AR(∞) Representation of Invertible Filter	381
	D.1.7	Calculus of Filters	381
	D.1.8	z Transform	382
	D.2	Stability	383
	D.3	Filters with Rational Transfer Function	383
	D.3.1	Definition	383
	D.3.2	Poles and Zeroes	385
	D.4	Predictions	387
	D.4.1	Conditional Distribution Lemma	387
	D.4.2	Predictions	387
	D.5	Log Likelihood of Innovation	389
	D.6	Matlab Commands	389
	D.7	Proofs	391
ANNEX	E	Bibliography	393
ANNEX	F	Index	401

Methodology

Perhaps the most difficult part when carrying out a performance evaluation is knowing where to start. In this chapter, we propose a methodology, i.e. a set of recommendations, that is valid for any performance evaluation study. We stress the importance of factors, in particular hidden ones, and the need to use the scientific method. We also discuss a few frequent performance patterns, as a means of quickly focusing on important issues.



1.1 What is Performance Evaluation?

In the context of this book, performance evaluation involves quantifying the service delivered by a computer or a communication system. For example, we might be interested in: comparing the power consumption of several server farm configurations; knowing the response time experienced by a customer performing a reservation over the Internet; comparing compilers for a multiprocessor machine.

In all cases it is important to carefully define the *load* and the *metric*, and to be aware of the performance evaluation *goals*.

1.1.1 Load

An important feature of computer or communication systems is that their performance depends dramatically on the *workload* (or simply *load*) that they are subjected to. The load characterizes the quantity and the nature of requests submitted to the system. Consider for instance the problem of quantifying the performance of a web server. We could characterize the load by a simple concept such as the number of requests per second. This is called the *intensity of the workload*. In general, the performance deteriorates when the intensity increases, but this deterioration is often sudden. The reason for this is the non-linearity of queuing systems – a *performance pattern* example that is discussed in Section 1.5 and Chapter 8.

The performance of a system depends not only on the intensity of the workload, but also on its nature. On a web server, for example, all requests are not equivalent: some web server softwares might perform well with *get* requests for frequently used objects, and less well with requests that require database access. For other web servers, things might be different. This issue is addressed by using standardized mixes of web server requests. These are generated by a *benchmark*, defined as a load generation process that intends to mimic a typical user behavior. Chapter 3 presents a study of how such a benchmark can be constructed.

1.1.2 Metric

A performance *metric* is a measurable quantity that precisely captures what we want to measure – it can take on many forms. There is no general definition of a performance metric: it is system dependent, and its definition requires a good understanding of the system and its users. We will often mention examples where the metric corresponds to throughput (the number of tasks completed per unit of time), power consumption (the integral of the electrical energy consumed by the system, per time unit), or response time (i.e. the time elapsed between a start and an end event). For each performance metric, we may be interested in the average, 95-percentile, worst-case, etc., as explained in Chapter 2.

Example 1.1 Windows versus Linux

Chen and co-authors compare Windows to Linux in [25]. As metric, they use the number of CPU cycles, the number of instructions, the number of data read/write operations required by a typical job. The load was generated by various benchmarks: “syscall” generates elementary operations (system calls); “memory read” generates references to an array; an application benchmark runs a popular application.

It is also important to be aware of the experimental conditions under which the metric is measured, as illustrated by the following example.