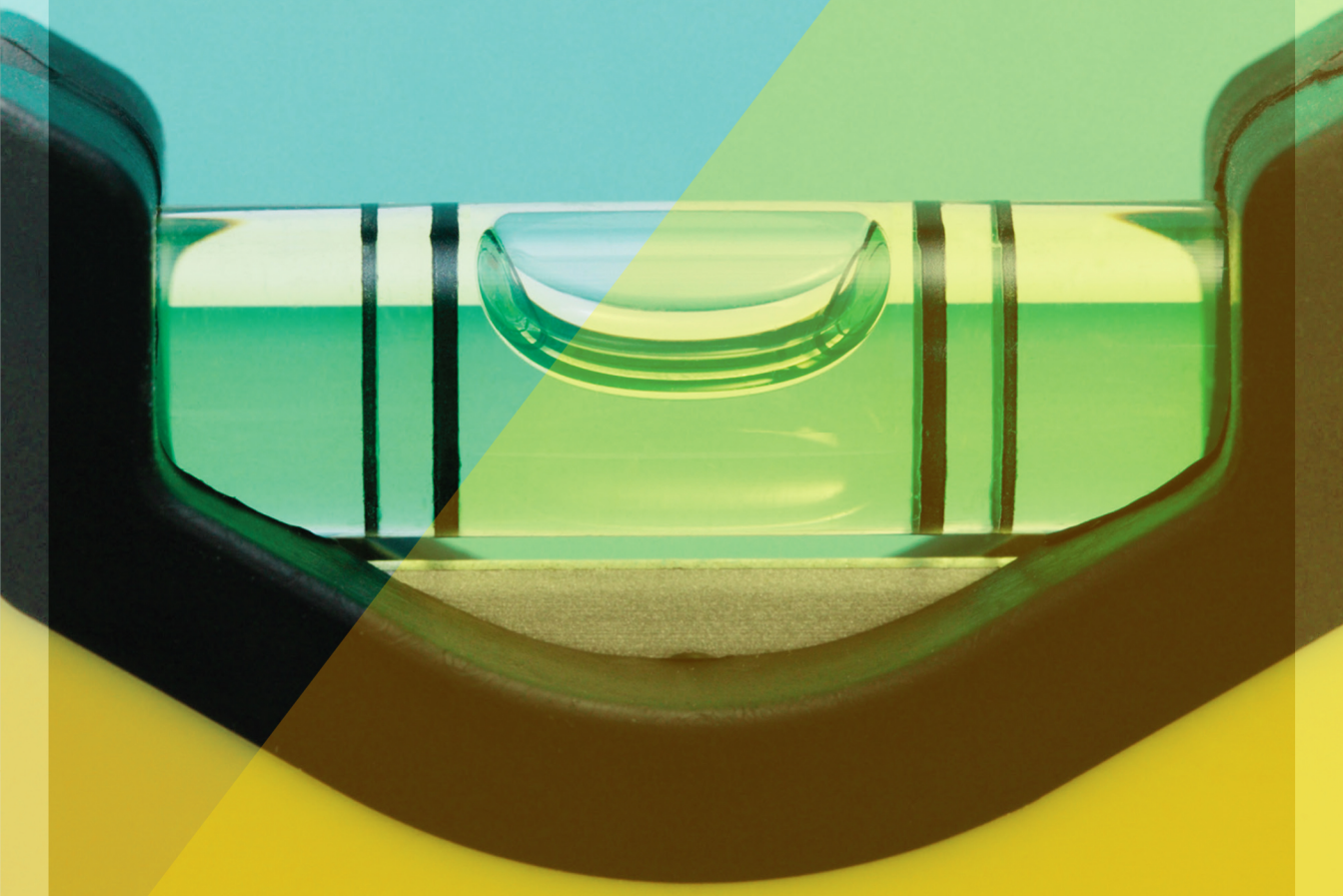
/Theory/In/Practice

The Art of Application Performance Testing



O'REILLY®

Ian Molyneaux

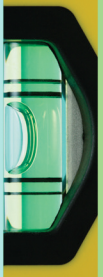
Software Engineering

The Art of Application Performance Testing

"Ian has maintained a vendor-agnostic methodology beautifully in this material. The metrics and graphs, along with background information provided in his case studies, eloquently convey to the reader, 'Methodology above all, tools at your discretion...' Ian's expertise shines through throughout the entire reading experience."

–Matt St. Onge, Enterprise Solution Architect,
HCL Technologies America / Teradyne

Businesses today live and die by network applications and web services. Because of the increasing complexity of these programs and the pressure to deploy them quickly, many professionals don't take the time to ensure that they'll perform well or scale effectively. *The Art of Application Performance Testing* explains the complete life cycle of the testing process and demonstrates best practices to help you plan, gain approval for, coordinate, and conduct performance tests on your applications.



You'll learn how to:

- Set realistic performance testing goals
- Implement an effective application performance testing strategy
- Interpret performance test results
- Use automated performance testing tools
- Test traditional local applications, web-based applications, and web services (SOAs)
- Recognize and resolves issues that are often overlooked in performance tests

Written by a consultant with 30 years of experience in the IT industry and over 12 years experience with performance testing, this easy-to-read book is illustrated with real-world examples and packed with practical advice. *The Art of Application Performance Testing* thoroughly explains the pitfalls of an inadequate testing strategy and offers you a robust, structured approach for ensuring that your applications perform well and scale effectively when the need arises.

Ian Molyneaux, EMEA SME (Subject Matter Expert) for Application Performance Assurance at Compuware, has held many roles in IT over the past 30 years. A techie at heart, he's shied away from anything management related.

US $34.99 **CAN $34.99**
ISBN: 978-0-596-52066-3
9 780596 520663 53499
O'REILLY® www.oreilly.com

The Art of Application Performance Testing

Ian Molyneaux

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

The Art of Application Performance Testing
by Ian Molyneaux

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor: Andy Oram
Production Editor: Adam Witwer
Production Services: Newgen Publishing and Data Services

Cover Designer: Mark Paglietti
Interior Designer: Marcia Friedman
Illustrator: Robert Romano

January 2009: First Edition.

Revision History for the First Edition:
2009-01-12 First release
2013-09-20 Second release
See *http://oreilly.com/catalog/errata.csp?isbn=9780596520663* for release details.

ISBN: 978-0-596-52066-3

[LSI]

1379519766

CONTENTS

PREFACE

This book is written by an experienced application performance specialist for the benefit of those who would like to become specialists or have started working at application performance testing.

Businesses in today's world live and die by the performance of mission-critical software applications. Sadly, many applications are deployed without being adequately tested for scalability and performance. Effective performance testing identifies performance bottlenecks in a timely fashion and tells you where they are located in the application landscape.

The Art of Application Performance Testing addresses an urgent need in the marketplace for reference material on this subject. However, this is *not* a book on how to tune technology X or optimize technology Y. I've intentionally stayed well away from specific technologies except where they actually affect how you go about performance testing. My intention is to provide a commonsense guide that focuses on planning, execution, and interpretation of results and is based on a decade of experience in performance testing projects.

In the same vein, I won't touch on any particular industry performance testing methodology because—truth be told—they don't exist. Application performance testing is a unique discipline and is crying out for its own set of industry standards. I'm hopeful that this book may in some small way act as a catalyst for the appearance of formal processes.

Although I work for a company that's passionate about performance, this book is tool- and vendor-neutral. The processes and strategies described here can be used with any professional automated testing solution.

Hope you like it!

—Ian Molyneaux, December 2008

Audience

Although this book is for anyone interested in learning about application performance testing, it especially targets seasoned software testers and project managers looking for guidance in implementing an effective application performance testing strategy.

The book assumes that readers have some familiarity with software testing techniques, though not necessarily performance-related ones.

As a further prerequisite, effective performance testing is really possible only with the use of automation. Therefore, to get the most from the book you should have some experience of automated testing tools.

About This Book

Based on a number of my jottings (that never made it to the white paper stage) and ten years of hard experience, this book is designed to explain why it is so important to performance test any application before deploying it. The book leads you through the steps required to implement an effective application performance testing strategy.

Here are brief summaries of the book's chapters and appendixes.

Chapter 1, *Why Performance Test?*, discusses the rationale behind application performance testing and looks at performance testing in the IT industry from a historical perspective.

Chapter 2, *The Fundamentals of Effective Application Performance Testing*, introduces the building blocks of effective performance testing and explains their importance.

Chapter 3, *The Process of Performance Testing*, suggests a best-practice approach. It builds on Chapter 2, applying its requirements to a model for application performance testing.

Chapter 4, *Interpreting Results: Effective Root-Cause Analysis*, teaches effective root-cause analysis. It discusses the typical output of a performance test and how to perform effective analysis.

Chapter 5, *Application Technology and Its Impact on Performance Testing*, discusses the effects of particular software environments on testing. The approach is generic, so many details regarding your applications will depend on the characteristics of the technologies you use.

Appendix A, *Transaction Examples*, provides examples of how to prepare application transactions for inclusion in a performance test.

Appendix B, *POC and Performance Test Quick Reference*, reiterates the practical steps presented in the book.

Appendix C, *Automated Tool Vendors*, lists sources for the technologies required by performance testing. It is not an endorsement and is not intended to be complete.

Appendix D, *Sample KPI Monitoring Templates*, provides some examples of the sort of Key Performance Indicators you would use to monitor server and network performance as part of a typical performance test configuration.

Appendix E, *Sample Project Plan*, provides an example of a typical performance test plan based on Microsoft Project.

Conventions Used in This Book

The following typographical conventions will be used.

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings and also within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

> TIP
>
> **This icon signifies a tip, suggestion, or general note.**

> CAUTION
>
> **This icon indicates a warning or caution.**

Glossary

The following terms are used in this book.

Application landscape
: A generic term describing the server and network infrastructure required to deploy a software application.

ICA
: Citrix proprietary protocol, Independent Computing Architecture.

ITIL
: Information Technology Infrastructure Library.

ITPM
: Information Technology Portfolio Management.

ITSM
: Information Technology Service Management.

JMS
: Java Message Service (formerly Java Message Queue).

Load injector
: A PC or server used as part of an automated performance testing solution to simulate real end-user activity.

IBM/WebSphere MQ
: IBM's Message Oriented Middleware.

POC
: Proof of Concept, a term used to describe a pilot project often included as part of the sales cycle. The intention is to compare the proposed software solution to a customer's, current application and so employ a familiar frame of reference. *Proof of value* is another term for a POC or Proof of Concept.

SOA
: Service-Oriented Architecture

Transaction
: A set of end-user actions that represent typical application activity. A typical transaction might be: log in, navigate to a search dialog, enter a search string, click the search button, and log out. Transactions form the basis of automated performance testing.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*The Art of Application Performance Testing* by Ian Molyneaux. Copyright 2009 Ian Molyneaux, 978-0-596-52066-3."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://safari.oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707 829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596520663

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our web site at:

http://www.oreilly.com

Acknowledgments

Many thanks to everyone at O'Reilly who helped to make this book possible and put up with the fumbling efforts of a novice author. These include editor Andy Oram, assistant editor Isabel Kunkle, managing editor Marlowe Shaeffer, Robert Romano for the figures and artwork,

Jacquelynn McIlvaine and Karen Crosby for setting up my blog and providing me with the materials to start writing, and Karen Crosby and Keith Fahlgren for setting up the DocBook repository and answering all my questions.

In addition I would like to thank my employer, Compuware Corporation, for their kind permission to use screenshots from a number of their performance solutions to help illustrate points in this book.

I would also like to thank the following specialists for their comments and assistance on a previous draft: Peter Cole, President and CTO of Greenhat, for his help with understanding and expanding on the SOA performance testing model; Adam Brown of Quotium; David Collier-Brown of Sun Microsystems; Matt St. Onge; Paul Gerrard, principal of Gerrard consulting; Francois MacDonald of Compuware's Professional Services division; and Alexandre Mechain from Compuware France.

Finally, I would like to thank the many software testers and consultants whom I have worked with over the last decade. Without your help, this book would not have been written!

CHAPTER ONE

Why Performance Test?

Faster than a speeding bullet . . .

—Superman, Action Comics

This chapter poses some fundamental questions concerning the subject of this book. What is performance? Why carry out performance testing in the first place? Here I also define when an application is considered performant versus nonperformant and then discuss some common causes of a suboptimal end-user experience.

Nonperformant (i.e., badly performing) applications generally don't deliver their intended benefit to the organization. That is, they create a net cost of time, money, and loss of kudos from the application users and therefore can't be considered reliable assets. If an application is not delivering benefits, its continued existence is definitely on shaky ground—not to mention that of the architects, designers, coders, and testers (hopefully there were some!).

Performance testing is a neglected cousin of unit, functional, and system testing, which are well understood in most businesses and where the maturity level is high in many organizations. It is strange but true to say that executives do not appreciate the importance of performance testing. This has changed little over the past ten years despite the best efforts of consultants like myself and the many highly publicized failures of key software applications.

What Is Performance? The End-User Perspective

When is an application considered to be performing well?

My years of working with customers and performance teams suggest that the answer is ultimately one of perception. A well-performing application is one that lets the end user carry out a given task without undue *perceived* delay or irritation. Performance really is in the eye of the beholder.

With a performant application, users are never greeted with a blank screen during login and can achieve what they set out to accomplish without letting their attention wander. Casual visitors browsing a web site can find what they are looking for and purchase it without experiencing too much frustration, and the call-center manager is not being harassed by complaints of poor performance from the operators.

It sounds simple enough, and you may have your own thoughts on what constitutes good performance. But no matter how you define it, many applications struggle to deliver an acceptable level of performance.

Of course, when I talk about an application I'm actually referring to the sum of the whole, since an application is made up of many component parts. At a high level we can define these as the application software plus the application landscape. The latter includes the servers required to run the software as well as the network infrastructure that allows all the application components to communicate.

If any of these areas has problems, application performance is likely to suffer.

You might think that *all* we need do to ensure good application performance is observe the behavior of each of these areas under load and stress and correct any problems that occur. The reality is very different because this approach is often "too little, too late" and so you end up dealing with the symptoms of performance problems rather than the cause.

Performance Measurement

So how do we go about measuring performance? We've discussed end-user perception, but in order to accurately measure performance there are a number of key indicators that must be taken into account. These indicators are part of the performance requirements discussed further in Chapter 2 but for now we can divide them into two types: *service-oriented* and *efficiency-oriented*.

Service-oriented indicators are *availability* and *response time*; they measure how well (or not) an application is providing a service to the end users. Efficiency-oriented indicators are *throughput* and *utilization*; they measure how well (or not) an application makes use of the application landscape. We can define these terms briefly as follows:

Availability

The amount of time an application is available to the end user. Lack of availability is significant because many applications will have a substantial business cost for even a small outage. In performance testing terms, this would mean the complete inability of an end user to make effective use of the application.

Response time

The amount of time it takes for the application to respond to a user request. For performance testing, one normally measures *system response time*, which is the time between the user's requesting a response from the application and a complete reply arriving at the user's workstation.

Throughput

The rate at which application-oriented events occur. A good example would be the number of hits on a web page within a given period of time.

Utilization

The percentage of the theoretical capacity of a resource that is being used. Examples include how much network bandwidth is being consumed by application traffic and the amount of memory used on a server when a thousand visitors are active.

Taken together, these indicators can provide us with an accurate idea of how an application is performing and its impact, in capacity terms, on the application landscape.

Performance Standards

By the way, if you were hoping I could point you to a generic industry standard for good and bad performance, you're out of luck because no such guide exists. There have been various informal attempts to define a standard, particularly for browser-based applications. For instance, you may have heard the term "minimum page refresh time." I can remember a figure of 20 seconds being bandied about, which rapidly became 8 seconds. Of course, the application user (and the business) wants "instant response" (in the words of the Eagles band, "Everything all the time"), but this sort of performance is likely to remain elusive.

Many commercial Service Level Agreements (SLAs) cover infrastructure performance rather than the application itself, and they often address only specific areas such as network latency or server availability.

The following list summarizes research conducted in the late 1980s (Martin 1988) that attempted to map user productivity to response time. The original research was based largely on green-screen text applications, but its conclusions are probably still relevant.

Greater than 15 seconds

This rules out conversational interaction. For certain types of applications, certain types of users may be content to sit at a terminal for more than 15 seconds waiting for the answer

to a single simple inquiry. However, to the busy call-center operator or futures trader, delays of more than 15 seconds may seem intolerable. If such delays can occur, the system should be designed so that the user can turn to other activities and request the response at some later time.

Greater than 4 seconds

These delays are generally too long for a conversation requiring the end-user to retain information in short-term memory (end-user's memory, not the computer's!). Such delays would inhibit problem-solving activity and frustrate data entry. However, after the completion of a transaction, delays of 4 to 15 seconds can be tolerated.

2 to 4 seconds

A delay longer than 2 seconds can be inhibiting to operations that demand a high level of concentration. A wait of 2 to 4 seconds at a terminal can seem surprisingly long when the user is absorbed and emotionally committed to completing the task at hand. Again, a delay in this range may be acceptable after a minor closure. It may be acceptable to make a purchaser wait 2 to 4 seconds after typing in her address and credit card number, but not at an earlier stage when she is comparing various product features.

Less than 2 seconds

When the application user has to remember information throughout several responses, the response time must be short. The more detailed the information to be remembered, the greater the need for responses of less than 2 seconds. Thus, for complex activities such as browsing camera products that vary along multiple dimensions, 2 seconds represents an important response-time limit.

Subsecond response time

Certain types of thought-intensive work (such as writing a book), especially with applications rich in graphics, require very short response times to maintain the users' interest and attention for long periods of time. An artist dragging an image to another location must be able to act instantly on his next creative thought.

Deci-second response time

A response to pressing a key of seeing the character displayed on the screen or to clicking a screen object with a mouse must be almost instantaneous: less than 0.1 second after the action. Many computer games require extremely fast interaction.

As you can see, the critical response time barrier seems to be 2 seconds. Response times greater than this have a definite impact on productivity for the average user, so our nominal page refresh time of 8 seconds for Internet applications is certainly less than ideal.

The Internet Effect

The explosive growth of the Internet has contributed in no small way to the need for applications to perform at warp speed. Many (or is that most?) businesses now rely on cyberspace for a good deal of their revenue in what is probably the most competitive