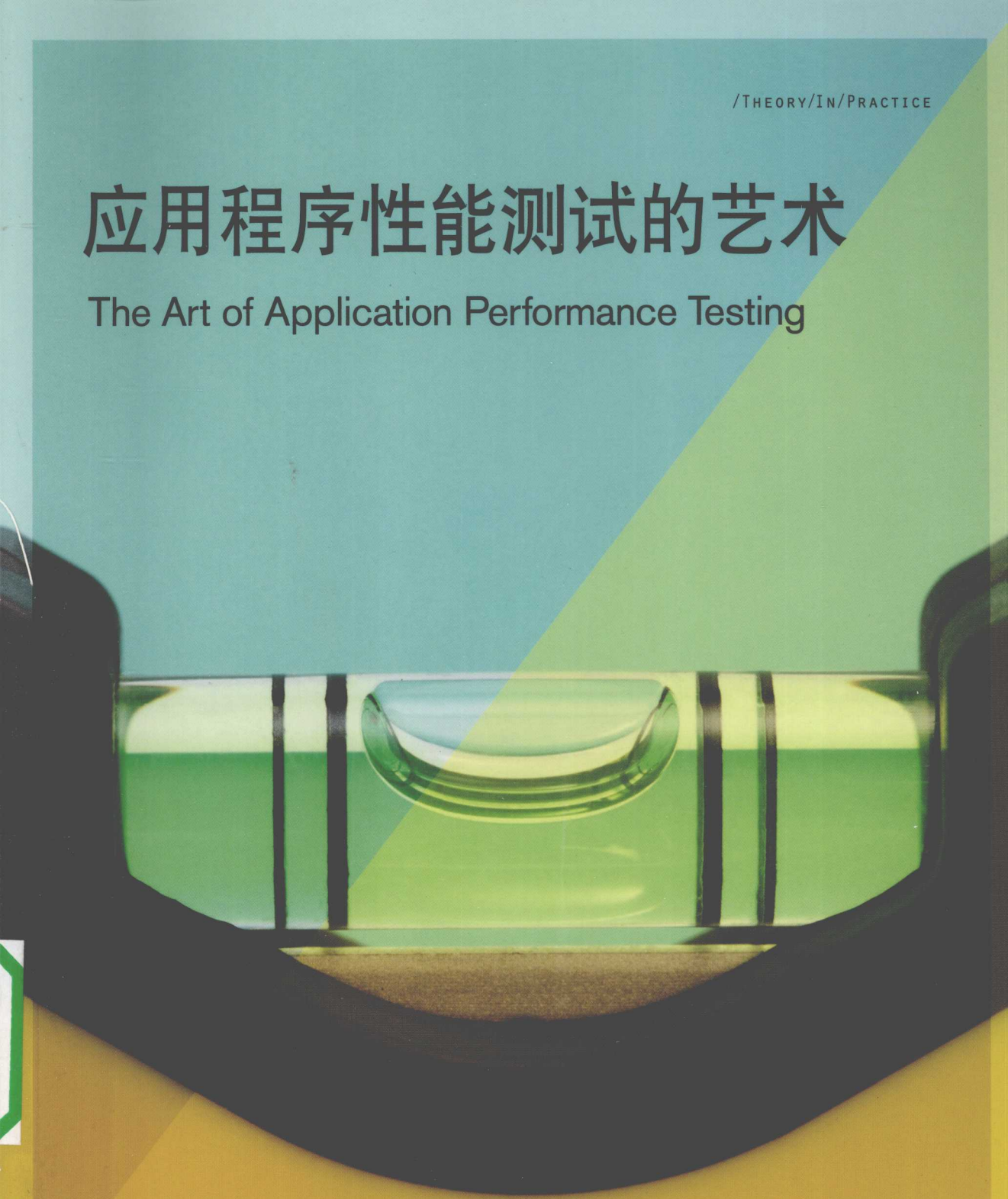
/THEORY/IN/PRACTICE

应用程序性能测试的艺术

The Art of Application Performance Testing



O'REILLY®

机械工业出版社
China Machine Press

HZ BOOKS

Ian Molyneaux 著
李刚 陈宇星 等译

应用程序性能测试的艺术

Ian Molyneaux 著

李刚 陈宇星 等译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc.授权机械工业出版社出版

机械工业出版社

图书在版编目（CIP）数据

应用程序性能测试的艺术/（新）莫里纽克斯（Molyneaux, I.）著；李刚，陈宇星等译. －北京：机械工业出版社，2009.9

书名原文：The Art of Application Performance Testing

ISBN 978-7-111-27382-0

I. 应... II. ①莫... ②李... ③陈... III. 主页制作－程序设计 IV. TP393.092

中国版本图书馆CIP数据核字（2009）第090175号

北京市版权局著作权合同登记

图字：01-2009-1577号

书　　名/ 应用程序性能测试的艺术
书　　号/ ISBN 978-7-111-27382-0
责任编辑/ 李东震
封面设计/ Mark Paglietti，张健
出版发行/ 机械工业出版社
地　　址/ 北京市西城区百万庄大街22号（邮政编码100037）
印　　刷/ 北京京师印务有限公司印刷
开　　本/ 178毫米×233毫米　16开本　9.25印张
版　　次/ 2010年1月第1版　2010年1月第1次印刷
印　　数/ 0001~4000册
定　　价/ 29.00元（册）

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：(010) 88378991; 88361066
购书热线：(010) 68326294; 88379649; 68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com

O'Reilly Media, Inc.介绍

为了满足读者对网络和软件技术知识的迫切需求，世界著名计算机图书出版机构O'Reilly Media, Inc.授权机械工业出版社，翻译出版一批该公司久负盛名的英文经典技术专著。

O'Reilly Media, Inc.是世界上在 Unix、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时也是联机出版的先锋。

从最畅销的*The Whole Internet User' Guide & Catalog*（被纽约公共图书馆评为20世纪最重要的50本书之一）到GNN（最早的Internet门户和商业网站），再到WebSite（第一个桌面PC的Web服务器软件），O'Reilly Media, Inc.一直处于Internet发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc.是最稳定的计算机图书出版商 —— 每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc.具有深厚的计算机专业背景，这使得O'Reilly Media, Inc.形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc.所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc.还有许多固定的作者群体 —— 他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media, Inc.依靠他们及时地推出图书。因为O'Reilly Media, Inc.紧密地与计算机业界联系着，所以O'Reilly Media, Inc.知道市场上真正需要什么图书。

译者序

跨入21世纪，全球信息化建设大踏步地深入到我们生活的每一个角落，软件开发技术得到了快速的发展，同时用户对应用程序的质量要求也越来越高。从可以准确完成单一任务到要求系统具备多业务协同处理能力，并能做到实时响应，这无疑对现有的技术提出了更高的要求，无疑也是个不小的挑战。在这样的大背景下软件测试技术也得到快速发展，“应用程序性能测试技术”作为保障应用程序性能的一项技术也越来越受到重视。

应用程序性能测试通常需要考虑几个阶段：性能需求的获取、性能需求分析、建立性能测试模型、性能测试开发、性能测试设计，以及性能分析、调优等；另外性能测试还有一个特点：“在性能测试过程中一般要引入性能测试工具，也就是说往往性能测试是不能以纯手工来完成的”。因此现在也存在着一个误区，很多人认为应用程序的性能测试就是软件性能测试工具的使用方法；觉得掌握了软件性能测试工具就掌握了应用程序性能测试的技术。现在为了满足这样一群人的需求，市面上也出了一些单纯介绍性能测试工具的书籍。译者以为：任何工具（包括性能测试工具），其本质只是完成某项任务的手段。衡量是否掌握完成任务的技术本质还是在于：“我们对于这项任务的分析、设计、实施整个过程解决方案的能力”。很高兴向大家推荐本书，此书在讨论应用程序性能测试的过程中并没有谈任何“性能测试工具”，通过对性能测试生命周期中每个阶段的充分分析，给出实施性能测试合理的方法；另外本书也没有基于某行业或者某技术，而是论述了一种普遍适用的性能测试解决方案。另外在本书中还提及到如何对性能测试进行分析，这个分析包括测试前期的分析（POC过程），测试结果的分析方法。这两种方法很少在现有的国内性能测试书籍中看到。

翻译从来不是一件轻松的事情，在这里我很感谢我们团队的精诚合作，我们的团队是在领测软件测试论坛（*http://bbs.ltesting.net/*）上认识的一帮志同道合的朋友，国内很多软件测试专家都在这里交流、提高。我的ID“阳光”。另外领测国际科技（北京）有限公司（*http://www.ltesting.com.cn*）也是我现在服务的公司，我们可以为您提供软件测试培训与软件测试相关服务的全面解决方案。

在此我特别感谢团队中的一名译者，我们都叫他“导演”，真名陈宇星，在本书翻译过程中我们一起对稿件进行了几次审核，感谢他的敬业精神和对团队的奉献。另外感谢参与这次翻译的其他成员：张彦兵（冰岩）、刘林（小林子）、李家国（家国）、袁礼。

在翻译过程中他们都付出了自己的艰辛。同时我也感谢我们的家人和朋友在翻译此书过程中给予的支持。

在这里我也代表我的团队感谢机械工业出版社华章公司的陈冀康编辑，在此书的翻译过程中得到他的很多指点和支持。在此对他及其公司一起完成这项任务的编辑表示衷心的感谢。

译者

2009年6月6日

目录

前言 1

第1章 为什么要进行性能测试 7

1.1 以最终用户的眼光看待性能 7

1.2 糟糕的性能：为何如此普遍 11

1.3 总结 16

第2章 有效应用程序性能测试的基本原则 17

2.1 选择合适的性能测试工具 18

2.2 设计合适的性能测试环境 23

2.3 制定切合实际的性能指标 29

2.4 确保在性能测试过程中应用程序足够稳定 36

2.5 做到代码冻结 37

2.6 识别并确认关键业务的事务 38

2.7 提供高质量的足够的测试数据 41

2.8 确保准确的性能测试设计 43

2.9 确定服务器和网络的关键性能指标 50

2.10 安排足够的时间确保有效的性能测试 53

2.11 总结 54

第3章 性能测试过程 55
3.1 概念验证 55
3.2 从需求到性能测试 58
3.3 案例学习1：网上银行 66
3.4 案例学习2：呼叫中心 73
3.5 总结 79
第4章 结果解析：有效的根源问题分析 80
4.1 过程分析 80
4.2 性能测试输出的类型 82
4.3 根本原因分析 93
4.4 分析报告检查列表 98
4.5 总结 101
第5章 应用程序采用的技术对性能测试的影响 102
5.1 Ajax 102
5.2 Citrix 103
5.3 HTTP协议 105
5.4 Java 107
5.5 Oracle 108
5.6 SAP 109
5.7 SOA 110
5.8 Web 2.0 110
5.9 怪异的应用技术 112
附录A 银行事务案例 115
附录B POC及性能测试快速参考 118
附录C 自动化测试工具厂商 127
附录D KPI监控模板实例 131
附录E 项目计划的例子 134

前言

本书由经验丰富的软件性能测试专家编写；本书的编写是为了帮助那些希望成为此领域专家的读者，并对已经从业于软件性能测试领域的读者给予一定的指导。

当今世界，企业的兴亡依赖于关键任务软件的性能状况。然而，不幸的是，很多软件未经过全面的伸缩性以及性能测试就配置并应用起来了。有效的性能测试能够及时地找到性能瓶颈，并指出问题所在。

本书旨在满足市场上对于性能测试参考资料的急迫需求。然而本书并非着重于如何调试X技术或者优化Y技术。我有意避开具体的技术问题，除非这项技术会实际地影响到读者执行性能测试。我意在为读者提供常识性的指导，重点在于测试计划、测试执行以及测试结果分析，这些都是基于我十年来在性能测试项目上的经验。

同样，本书不会论及具体行业上性能测试的方法，因为，事实上，这种方法并不存在。软件性能测试是一门独特的学科，它迫切需要一套适用于自身的行业标准。希望本书能为正规测试过程的出现尽绵薄之力。

虽然我在一个热衷于软件性能的公司工作，但是本书既不能作为工具书用，也不和任何厂商挂钩。所以本书中所表述的测试过程和测试策略可以应用于任何专业的自动化测试解决方案中。

希望您能喜欢这本书！

——Ian Molyneaux，2008年12月

致读者

虽然本书是面向任何对软件性能测试感兴趣的读者，但是它更倾向于为资深的软件测试员和项目经理提供参考，以助于他们更有效地执行软件性能测试策略。本书要求读者熟悉一定的软件测试技术，即使那些技术与性能测试毫不相干。

高效的性能测试需要应用必要的自动化工具，因此如果您想在本书中学到更多东西，您还需要有使用自动化测试工具的经验。

关于本书

本书基于我大量笔记（虽然没有印刷出版）以及十年艰辛的经历，旨在说明在软件配置之前对其进行性能测试的重要性。本书会展示执行一个有效的性能测试策略的必要步骤。

以下简要总结一下本书的章节和附录。

第1章， 本章讨论软件性能测试的基本原理，从历史的角度关注IT行业中的性能测试。

第2章， 介绍有效性能测试的架构基础，以及它们的重要性。

第3章， 介绍一个基于第2章的典范实践方法，将自己的需求应用于一个软件性能测试模型。

第4章， 讲授有效的根本原因分析。论述一个性能测试的典型输出，以及如何进行有效的分析。

第5章， 论述特殊软件环境对测试的影响。这是一个一般性的方法，所以很多关于软件的详细情况视应用的技术特点而定。

附录A，为如何准备性能测试中的软件事务处理提供一个例子。

附录B，重申本书中所讲述的实用测试步骤。

附录C，为性能测试所需要的技术列出来源，它们并非标准，也不一定完整。

附录D，提供某种关键性能指标的例子，您可以用来监视服务器和网络性能，将其作为一个典型的性能测试配置。

附录E，提供一个基于微软项目的典型性能测试计划的例子。

本书体例

以下是印刷符号在本书中的约定用法：

斜体（*Itailc*）

表示新的条目，网址、电子邮件地址、文件名和文件扩展名。

宽体（Constant width）

用于程序列表，文本中涉及的程序元素，如变量或函数名、数据库、数据类型、环境变量、语句、应用、关键词以及模块。

宽粗体（**Consrant width bold**）

显示命令或其他应该由用户逐个输入的文本。

宽斜体（*Consrant width italic*）

显示应该由用户提供的值或上下文确定的值进行替换的文本。

技巧

这表示一个技巧、建议或者一般注解。

注意

这表示一个警告或者提示读者谨慎。

术语

以下是用于本书的术语。

应用架构（Application landscape）：

一个通用术语，描述了服务器和网络基础设施需要部署的应用软件。

ICA（Independent Computing Architecture）独立计算架构：

恩杰（译注1）开发的一种私有协议。

ITIL（Information Technology Infrastructure Library）：

信息技术基础设施库。

ITPM（Information Technology Portfolio Management）：

信息技术组合管理（译注2）。

ITSM（Information Technology Service Management）：

信息技术服务管理（译注3）。

译注1：关于恩杰请参见：*http://www.citrix.com.cn/*。

译注2：《IT Portfolio Management Step-by-step: Unlocking The Business Value Of Technology》。

译注3：相关信息可以参见*http://www.itsm.info/home.htm*。

JMS（Java Message Service）：

Java消息服务（Java消息队列）

负载生成器（Load injector）[译注4]

负载生成器是指一台用于模拟用户真实活动的PC（个人电脑）或者服务器，它是自动化性能测试解决方案的一部分。

IBM/WebSphere MQ（译注5）：

IBM消息中间件产品。

POC（Proof of Concept）：

概念验证本词常用于描述一个属于销售环节的试点项目，把一个提出来的软件解决方案比作一个客户的当前的应用程序，并由此应用一个框架作为参考。价值证明是POC的另一种说法。

SOA（Service-Oriented Architecture）：

面向服务的架构。

事务（Transaction）

事务体现典型应用程序最终用户活动的一系列操作。一个典型的事务必须是：登录，找到搜索窗口，输入搜索字符，点击搜索按钮，然后退出。大量事务组成自动化测试的基础。

书中代码使用规定

本书是用来帮助您完成工作的。总体来说，您可以在程序和文档里使用本书中的代码，而不需要联系我们取得授权，除非您要复制大批的代码。例如，利用本书中的代码块写一个程序的话，不需要征求我们的授权。把O'Reilly的书中的范例做成光盘出售的话，则需要授权。引用本书或书中的范例代码来回答问题不需要授权，从本书中合并大量代码，放到您自己的出版文档里需要授权。

我们感谢您在自己的文档里写上引用版权声明，但是并不要求您这么做。一个版权声明的写法包括：题目、作者、出版社和ISBN（国际标准图书编号）。例如："《The Art of Application Performance Testing》，作者：Ian Molyneaux，Copyright 2009 Ian Molyneaux，978-0-596-52066-3。"

译注4："Load injector"有的时候我们也称其为"负载生成器"，比如在LoadRunner测试工具中，我们称其为负载生成机，为"负载生成器"。

译注5："消息中间件及WebSphere MQ入门"：*http://www.ibm.com/developerworks/cn/websphere/library/techarticles/loulijun/MQnewer/MQnewer.html*。

如果您觉得您引用本书代码范例的程度超出上述标准，需要授权，请联系*permissions@oreilly.com*。

联系方法

如果您对本书的意见和问题，请联系出版社：

美国：

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

中国：

北京市西城区西直门南大街2号成铭大厦C座807室（100035）
奥莱利技术咨询（北京）有限公司

我们为本书设有专门网址，那里有我们列出的勘误表，范例以及任何附加的信息。您可以点击此页面：

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

要为本书提意见或者求解技术问题，请发邮件到：

bookquestions@oreilly.com。

想了解更多关于本书会议、资源中心、O'Reilly网络请到此网站：

http://www.oreilly.com
http://www.oreilly.com.cn。

鸣谢

非常感谢曾在O'Reilly为本书的面世给予帮助的每一位朋友，感谢您们容忍一个初出茅庐的作者在写作方面的笨手笨脚的摸索。这些朋友是：编辑Andy Oram，助理编辑Isabel Kunkle，主编Marlowe Shaeffer，插图和美工Robert Romano，Jacquelynn McIlvaine 和Karen Crosby；感谢你们为我开了博客并且为我准备写作的材料；感谢Karen Crosby和Keith Fahlgren为我建立DocBook库并且为我解疑答惑。

另外，我要感谢我所在的Compuware公司；感谢他们允许我用他们性能测试解决方案中的大量截图来解释我书中的要点。

我还要感谢以下专家学者，他们给我的初稿提供了大量的帮助和参考意见，他们是：Greenhat公司董事长和首席技术官Peter Cole先生，他帮助我理解并扩展了SOA性能测试模型；Qunotium公司的Adam Brown先生、Sun微系统公司的David Collier-Brown先生、Matt St. Onge先生、杰拉德咨询公司的主管Paul Gerrard先生、Compuware公司专业服务部的Francois MacDonald先生以及Compuware法国分公司的Alexandre Mechain先生。

最后，感谢这十多年来和我并肩工作的软件测试工程师们和咨询师们，如果没有你们的帮助，这本是写不出来的！

第1章

为什么要进行性能测试

快过飞驰的子弹……

——动作漫画《超人》

本章提出了与本书所有需要讨论主题相关的一些根本性问题。什么是性能测试？为什么执行性能测试至关重要？在本章中，我也定义了什么是好的性能体验、什么是不好的性能体验，并且讨论了一些导致最终用户体验不佳的共同因素。

性能不佳的应用程序（例如，性能表现极差）通常无法实现企业预期利益，也就是说，企业为此（程序性能）花费了时间和金钱，但是却在应用此程序的用户中失去了信誉，因此（这样的程序）不能视为可靠的资产。

如果一个应用程序不能够为企业带来效益，那么它的存在就会岌岌可危，更不用说（与这个程序相关的）那些架构师、设计师、程序员和测试员了。（但愿有那么一些！）

对于大多数的企业和那些成熟度较高的组织来说，单元测试、功能测试和系统测试是很容易理解的，然而性能测试却往往容易忽略。管理层并不能体会到性能测试的重要性，尽管这很奇怪，但确实是这样的。在过去的十年里，尽管许多像我这样的顾问为此做了大量的宣传努力，但仍然收效甚微。

1.1 以最终用户的眼光看待性能

一个应用程序在什么样的情况下才会被认为拥有好的性能呢？

我多年与客户和性能团队共事的经验表明“性能”是用户的一种最终感受。一个性能优异的应用程序，在最终用户执行某项任务时，程序不会产生过度的延迟而引起用户的不满。关于性能这件事，正所谓当局者迷，旁观者清（译注6）。

译注6：“这里的当局者指的是：系统的开发人员；旁观者指的是：最终用户”。

一个性能表现优异的应用程序，用户不会在登录系统的时候，迎来的却是一个空空如也的屏幕，并且用户可以流畅地完成他们预定的任务，而不会走神（译注7）。当用户在一个购物网站上寻找和购买他们所需要的东西时，他们可以随心所欲地浏览购物网站，而不会因为网站性能太差而感到很郁闷。

这听起来似乎并不难，对于究竟什么才算是好的性能，您应该也会有一套自己的看法。但无论您是怎么去定义它的，许多应用程序都在不断地改进，以使自己的性能提高到可以被用户接受的程度。

当然，由于一个应用程序是由很多部分所构成的，所以我所说的应用程序，实际上指的是一个应用程序中所有部分。从更高的层面上看，我们可以将这些组成部分定义为应用程序软件，加上应用架构，后者包括了运行软件所需的服务器硬件，以及能够保证所有应用组件间互相通信的网络基础设施。

当然一旦这些组成部分中的任何一个出现问题，整个系统的性能就将面临灾难。

您可能会简单地认为，为了保证应用程序拥有良好的性能，我们所要做的仅仅是观察应用程序的每个部分在负载压力下的运行状况，并且解决所出现的问题而已。然而现实情况却并不是这样，因为这种做法往往“太少”和“太迟”了，因此只能是治标不治本。以下的内容中将详细介绍如何进行性能评价。

1.1.1 性能度量

那么，我们应该如何去衡量系统的性能呢？我前面已经提到了最终用户的体验，但是为了更加准确地衡量性能，还有一些关键的指标需要考虑。这些关键指标在本书的第2章中还会继续深入探讨；在这里，我们不妨将它们划分为两种类型：服务型和效率型。

服务型指标有：可用性和响应时间。它们所衡量的是应用程序为用户服务效果的好坏。效率型指标有：吞吐量和利用率。它们所衡量的是应用程序在应用架构基础上发挥效率的高低。我们可以简单地定义如下。

可用性

它是应用程序对于最终用户的可用时间（译注8）。可用性不好是很严重的问题，因为即便是一个小小的故障也会导致大量的商务上的花费。在性能测试方面，这将意味着最终用户无法有效地使用该应用程序。

译注7：“这里说明，系统应该快速响应用户的请求”。

译注8：这里的“可用性”就是我们通常说的平均无故障时间。系统在不出现故障的情况下可以运行的最长时间。有的时候我们也叫它作系统的“可靠性”。

响应时间

应用程序对用户请求给出响应的时间。对于性能测试而言，一个最常用的衡量指标就是系统响应时间，即从用户端发出请求到接收到应用程序给出完整响应所经过的时间。

吞吐量

面向应用程序事件的发生频率。例如，在一段特定的时间内对某个Web页面点击的次数（译注9）。

利用率

占用系统资源的百分比。例如，当1000个用户同时在线时，在应用程序上消耗了多少网络带宽，以及在服务器上占用了多少内存等。

我们将以上两种类型的指标结合起来考虑，就能够较为准确地衡量一个应用程序的性能，以及它在应用架构基础上对系统能力造成的影响。

1.1.2 性能标准

如果您指望我在这里给出一个关于性能好坏的行业标准，那我很遗憾地告诉您，没有这样的指导标准存在。不过，业内倒是有许多非正式的标准，试图对系统的性能好坏做出评价，尤其是针对基于B/S架构的应用程序。举例来说，您可能听说过“页面最小刷新时间”这种说法；我记得它从20秒迅速提高到8秒。当然了，用户和企业都希望系统能够“即时响应”（用老鹰乐队的话来说就是：“随时待命”），但这样的性能要求是很难达到的。

许多商业服务水平协议（SLA）涵盖基础设施的性能，而不是应用程序本身，他们往往只涉及具体的领域，如网络延迟或服务器可用性等。

以下的内容列出了在20世纪80年代后期，马丁（1988年）研究如何使用响应时间来衡量

译注9：这里作者举的例子指的就是我们在性能测试中通常说的一个指标“每秒钟点击数（Hits/s）”，单位时间向Web服务器发出的HTTP请求数（这里的请求包括HTML资源的请求和非HTML资源的请求）。HTTP请求包括以下类型：

1. GET：从服务器上发送给客户端指定的资源。
2. PUT：从客户端存储数据到一个指定的服务器资源。
3. DELETE：从服务器上删除指定的资源。
4. POST：发送客户端数据给服务器。
5. HEAD：对于指定的资源只发送HTTP header响应信息，即不传输主体数据。