J2EE Performance Testing With BEA WebLogic Server

国外IT精品丛书

J2EE性能测试

涵盖了为评估和了解J2EE应用程序的性能所需的所有内容

〔美〕 Peter Zadrozny
Philip Aston
Ted Osborne 著

张文耀 叶茂盛 陈爱国 等译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

EXPERT

J2EE Performance Testing With BEA WebLogic Server

J2EE性能测试

Peter Zadrozny
〔美〕 Philip Aston 著
Ted Osborne

张文耀 叶茂盛 陈爱国 等译

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 提 要

本书针对J2EE应用程序性能测试，介绍了一种性能测试方法和一个辅助测试工具——The Grinder，以及该方法和工具在实践当中的应用。全书共7章，可分为4个部分：第一部分（第1章）详细介绍了书中提出的性能测试方法，以及与性能测试有关的概念和观点；第二部分（第2章）介绍了辅助测试工具The Grinder的使用；第三部分（第3章）阐述了如何黑盒测试一个完整的J2EE应用程序的性能；第四部分（第4章至第7章）介绍了设计应用程序时的性能测试，阐述了在应用程序设计过程中各种选择决定的性能比较测试，以及评价某些设计决定性能代价的方法，涉及的内容有HTTP、servlet、EJB设计模式和JMS体系结构等等。

本书观点新颖，内容详实，给出了大量的测试实例与对比分析，介绍的方法和操作切实可行，同时提供了详细的测试环境、测试脚本和测试代码，这些内容稍加修改就可以在读者自己的测试中使用。因此，本书是一本难得的专门的应用程序性能测试参考资料，可以供应用程序设计人员、开发人员和系统管理员学习使用，也可供计算机专业的学生以及关心性能的决策者和相关技术人员阅读参考。

版权贸易合同登记号：01-2002-4220

图书在版编目（CIP）数据

J2EE性能测试/（美）扎德罗兹尼（Zadrozny, P.）等著；张文耀等译.—北京：电子工业出版社，2003.4
书名原文：J2EE Performance Testing with BEA WebLogic Server
ISBN 7-5053-8598-4

Ⅰ. J… Ⅱ. ①扎… ②张… Ⅲ. JAVA语言－程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2003）第019098号

责任编辑：春 丽
印 刷：北京天竺颖华印刷厂
出版发行：电子工业出版社 http://www.phei.com.cn
北京市海淀区万寿路173信箱 邮编：100036
北京市海淀区翠微东里甲2号 邮编：100036
经 销：各地新华书店
开 本：787×1092 1/16 印张：22.5 字数：570千字
版 次：2003年4月第1版 2003年4月第1次印刷
定 价：39.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换，若书店售缺，请与本社发行部联系。联系电话：（010）68279077

作 者 简 介

Peter Zadrozny

Peter Zadrozny自1976年开始使用UNIX操作系统（Research Version 6）以来，一直在计算机行业工作。从那时起他就一直使用前沿技术开发集成应用程序，涉及世界范围内各种各样的工作。目前他是BEA System欧洲、中东和非洲部的首席技术专家。

Peter还是“Professional J2EE Programming with BEA WebLogic Server”（Wrox Press）一书的合著者、WebLogic Developer's Journal杂志的创办编辑（Founding Editor）。他拥有Caracas的Universidad Simón Bolivar的计算机工程学位。

Philip Aston

Philip Aston是BEA Professional Services的高级顾问，专攻WebLogic Server。在这个位置上，Phil帮助BEA的顾客使用J2EE技术设计详细的体系结构、证明相关概念并实现应用程序。他还为WebLogic Developer's Journal撰写文章。在业余时间Phil担当起领导开发员的角色，负责The Grinder，即本书使用的Java负载生成工具。他希望有一天储蓄了足够的钱就放弃软件而周游世界。

感谢在Parallax和BEA的新老同事们。

Ted Osborne

Ted Osborne（www.tedosborne.com）是一位独立的软件咨询员、开发人员和作者，致力于建立在J2EE平台上的应用程序性能测量。他是主要行业会议上常见的演讲人，也是“Professional EJB”（Wrox Press）的合著者。在Empirix（后来成为Teradyne的Softbridge部门）的时候，Ted是创造www.TestMyBeans.com（现在的Empirix的Bean-test）的创办成员之一，那是世界上第一个针对Enterprise JavaBean中间件的自动的负载测试工具。从那时起Ted已经碰到了成百上千的J2EE开发员、项目管理员、QA职员和其他世界范围内的专家，了解他们的EJB和J2EE应用程序性能问题。Ted拥有Berklee大学音乐学院的Jazz Composition学位。

我要感谢Tony Davis和Peter Zadrozny给了我投稿的机会，感谢Dan O'Connor内行的洞察力，感谢George Friedman所有慷慨的指导；同时要感谢我的妻子Colette，感谢她的挚爱、耐心与支持。

审校组

Joakim Dahlstedt

Joakim Dahlstedt是BEA Systems Inc.的Java Runtime Products Group的CTO，他在那里负责JVM未来的发展方向。他是JRockit的创立者，也是主要的体系结构设计人员之一。JRockit是BEA在Q1 2002中得到的JVM。从1996年以来他一直从事Java和动态运行优化方面的工作。

Bjarki Hólm

Bjarki Hólm是deCODE Genetics的软件工程师，那是一个位于冰岛雷克雅未克的基于人口的基因组公司。他主要感兴趣的领域是Java和Oracle，在他的工作中经常把两者结合起来。在Wrox，Bjarki已经与人合作写了4本书并帮助审阅了一些其他作者的书。当他没有工作或写作时，他喜欢投入自己的业余爱好，其中包括徒步旅行和业余的机器人建造。

Staffan Larsen

Staffan Larsen是JRockit Java Virtual Machine的体系结构设计者/设计员之一，同时也是该公司的创立者之一。他已经从事了数年分析Java性能问题、测试大范围的Java应用程序的工作。目前他致力于针对BEA系统的JRockit Java Virtual Machine项目。

Daniel O'Connor

Daniel O'Connor是MVCSoft Inc.的总裁/CEO，是MVCSoft Persistence Engine的作者。MVCSoft Persistence Engine为J2EE应用程序服务器提供了可插入的EJB 2.0容器管理的持久性。

Andrew Sliwkowski

Andrew Sliwkowski从事WebLogic Server Clusters的可伸缩性测试，此外还给顾客提供任务关键的应用程序支持。

献给

我的儿女Gracie、Johannes和Kathelijne

我的妻子Graciela，感谢她无限的忍耐与难以置信的支持

Peter

致　谢

感谢Steve Allen, BEA Systems欧洲、中东和非洲部的高级副总裁。你建立了性能实验室并支持了最终产生本书内容的所有研究工作。非常感谢你！

特别感谢Ted Osborne，他承担了在非常短的时间内编写EJB章节和JazzCat应用程序的挑战。

感谢Phil Aston创作了第二代The Grinder并编写有关的章节。此外他还编写了JMS章节中使用的Grinder插件、servlet章节的测试程序。他的贡献遍及全书。

我必须感谢Tony Davis，是他通过不同的编辑周期，收集并编辑来自不同投稿者、审校者和我自己的所有内容，并把它们融合到一起形成本书的最终版本。同样感谢审校组的人员，感谢他们的重要贡献与“不同意见”。

我还必须感谢Bjarki Hólm和Gareth Chapman帮助我进行了HTTP会话对象和有状态的EJB比较，感谢Paul Crerand对JMS章节的贡献，感谢Andrew Sliwkowski提供了附录C中的ECperf资料，感谢Staffan Larsen对JVM工作方式的见解，感谢Thomas Kyte提供的一些Oracle专门技术，感谢Gopalan Suresh Raj提供了附加的JMS资料。最后，我要感谢BEA支持团体中的Richard Wallace和Jay Backory提供的所有帮助。

译 者 序

在应用程序的开发过程中，我们经常面临“应用程序运行速度如何？适用规模有多大？”等等与性能相关的问题。然而由于每一个应用程序都是独一无二的，我们没法笼统地回答这样的问题，所以获取有关性能答案的惟一方法就是：在自己特定的环境中亲自测试它。

本书针对J2EE应用程序性能测试介绍了一种性能测试方法和一个性能测试工具。借助该工具和方法，J2EE开发人员可以针对自己的应用程序获取可靠的、有意义的性能数据。该方法提供的性能测试框架描述了如何定义性能测度和性能目标、需要采取的测试、测试数据的收集与分析以及最终执行的测试方式。为了配合该方法的使用，还介绍了一个优秀的基于Java的性能测试工具——The Grinder，这是一个负载生成与数据收集工具。

全书围绕着该测试方法和测试工具展开，详细地叙述了以下几个方面的内容：

- 本书提出的性能测试方法，以及与性能测试有关的基本概念和观点；
- 性能测试工具The Grinder的使用；
- 如何黑盒测试一个完整的J2EE应用程序的性能；
- 应用程序在设计阶段的性能测试，这主要分析J2EE API不同方面的性能代价，以及某种设计决策对总体性能的影响，内容涵盖了HTTP、Servlet、EJB设计模式和JMS体系结构等等。

本书在阐述这些内容的过程中融入了大量的测试案例，详细地介绍了测试环境的建立、测试脚本和代码的编写、测试过程的具体操作步骤，并给出了丰富的性能测试结果与分析，具有很高的指导参考价值，可供下载的代码稍做调整就可以用于自己的应用程序性能测试。但是正如作者一再强调的那样，不能从这些结果推断其他应用程序的性能，必须亲自测试以便获取自己应用程序的性能数据。

本书的性能测试虽然使用了BEA WebLogic Server，但是介绍的技术可以用于任何J2EE应用程序和任何J2EE兼容的应用程序服务器；介绍的方法简单、灵活，不附属于任何特定的测试工具，个人可以自由使用自己喜欢的工具。本书作为J2EE应用程序性能测试方面的专著，既有理论又有实践，不仅可以供应用程序设计人员、开发人员和系统管理员学习使用，也可供计算机专业的学生以及关心性能的决策者和技术人员阅读参考。

本书作者亲自参与了本书测试方法和测试工具的开发与研究，观点新颖，内容详实，条理清晰。本书由张文耀、叶茂盛、陈爱国、廖频、黄早亮、苏菲和饶莹等人翻译，译者在忠实于原文的基础上，力求保持原文的风貌。然而，由于译者水平有限且时间仓促，错误与不妥之处在所难免，恳请广大读者谅解，并欢迎批评指正。

简　介

每个J2EE应用程序都是真正独一无二的。这使得从一个应用程序推断另一个应用程序的性能非常困难，甚至不可能，而不管它们的功能是何其相似。为了掌握你自己应用程序的性能，你必须按照自己的性能定义亲自测试它。

就性能而言，我们经常提到的一些问题有：

- 应用程序的运行有多快？
- 它将适用于多大的规模？
- 应用程序服务器的性能是什么？

就像开篇所指出的那样没有明确的答案，我们也不能提供一个解决所有与性能有关的问题的清单。但是，我们认为我们所能做的是提供一个工具和一种性能测试方法，让J2EE开发人员针对他们自己的应用程序，获取可靠的有意义的性能数据。本书描述了这两个方面以及我们所承担的性能研究与研究结果。这些结果只是用来强调文章开始所说情况的真实性。

我们的测试方法提供了一个清晰的框架。该框架描述了如何定义性能测度和目标、需要采取的测试、收集数据的方法，以及最终真正执行测试的方式。该方法非常简单、灵活，而且能够适应于任何以其他语言写的应用程序。

我们选择了使用一个称为The Grinder的负载生成/数据收集工具（随同相关的软件）来说明该方法的优点。The Grinder是一个基于Java的工具，可以在公共域获得。尽管在商业和公共领域都有各种各样的负载生成工具，但是作者直接参与了The Grinder的开发，所以它成了一个顺理成章的选择。然而，这并不意味着你必须使用该软件。相反，该测试方法不附属于任何特定的工具，因此可以自由使用你感觉最舒适的工具。

虽然本书的性能测试是使用BEA WebLogic Server进行的，但是介绍的技术可以用于任何J2EE应用程序和任何J2EE兼容的应用程序服务器。可供下载的代码中的某些组件（如配置文件）是专用于WebLogic Server的，但是改编该代码并在你自己的环境中运行测试是一件比较简单的事情。此外，我们所做的某些环境优化将利用该服务器的专有特性，但是其他应用程序服务器也很可能提供了类似的优化。

在本简介的剩余部分，我们将尝试确切地说明我们的方法和工具所能达到的目标，并指出它们与其他可用工具和方法的关系。我们还将为本书的剩余部分提供一个清晰的简要介绍。

J2EE性能测试

我们的方法和工具可以用于以下两个基本情况：

- 性能测试一个完整的应用程序。
- 性能设计——分析J2EE API不同方面的性能代价，以及某种设计决策对总体性能的影响。

在第一种情况下，我们将应用程序看做一个黑盒子，使用该方法和The Grinder以测试该应用程序在不同负载下的性能。我们可以调查应用程序产生的每个请求的性能（简单的JSP请求、调用整个实体bean的会话bean、与数据库交互以便更新记录的实体bean，等等）。我们能够分析数据，发现限制性的请求，并确定改善性能的机会。

虽然以上这些是有帮助的，但我们建议在开发周期中尽可能早地进行测试。这样可以使用该方法以及从中获得的数据来帮助你进行性能设计，而不是“事后”的性能测试。每个小的组成应用程序的子系统都将直接影响整个性能。通过早期测试，将能够在设计中使用最佳规划惯例，并回答将这些惯例应用于你惟一的应用程序时的一些基本问题，比如“应该使用哪个J2EE API？”、“是否应该避免使用静态会话bean？”、“实体bean是否太慢？”、“哪种J2EE设计模式适合于我的情况？”，等等。

本书用了一章介绍黑盒应用程序性能测试，用了五章探讨与流行的J2EE API（Servlet、EJB和JMS）的使用有关的共同问题及其设计选择。我们将展示如何评估选择对象，并利用我们所提出的性能结果，设置一个一般的相对性能期望值。然而，最终的结论是你不能想当然地假定我们的结果将适合于你的环境，你必须获取自己的性能数据。

什么是性能

为了理解自己的应用程序的性能，你必须按照自己的性能定义亲自测试它。这是非常重要的一点。我们经常被问到的常见问题之一是：“我的应用程序在你的服务器上的性能将怎么样（在我们的实例中，服务器就是WebLogic Server）？”这通常指望我们提供一个有魔力的数字，描绘整个性能。实际上，我们的回答是性能依赖于应用程序以及性能的确切含义。

J2EE是一组广泛的API，甚至一个相对简单的J2EE应用程序都可以以多种方式编写。例如，让我们考虑一个简单的数据库访问应用程序。从前端开始，你可以有一组JSP和servlet处理与终端用户或客户端的通信。该前端使用JDBC可以直接和数据库通信。然而，开发人员可以选择让该前端调用一个无状态的会话bean。然后该无状态的会话bean使用JDBC API与数据库通信，或者是将该访问委托给一个实体bean。实体bean可以使用容器管理的持久性（Container-Managed Persistence，CMP）或者是Bean管理的持久性（Bean-Managed Persistence，BMP），等等。

甚至在使用同一个API集合的时候，一个开发人员也可能以完全不同于别人的方式使用它们。而且我们知道在开发人员之间其代码的质量存在巨大的差别。即使代码是同样的，输入数据也很可能不同。这将不可避免地造成不同的使用模式，进而产生不同的性能结果。就像我们开始所说的那样，每个应用程序都是惟一的，获取清晰的真实的有关其性能的答案的惟一方法是，在你自己的特定环境中亲自测试它。

这把我们带回到实际的性能定义。在有些情况下，好的性能意味着能够支持巨大的用户数量；在另外一些情况下，用户负载可能比较小，好的性能可能仅仅是运行得尽可能快。为了更深入地分析，我们可以将开发的应用程序分成两个基本的类别：

- 交互式的，即当终端用户与应用程序同步交互时。
- 批处理或后端应用程序，即当不需要直接与终端用户交互时。

交互式应用程序

对于交互式应用程序，性能一般是通过大小和规划问题的容量来定义，例如应用程序能够处理的同时发生的用户数量。或者是更具体地讲，就是能够支持指定的用户数，而同时满足一些关键的参数。这些“必需的参数”可能包括特定的部署硬件，或者是最大响应时间（应用程序完成任何来自终端用户给定请求所花费的时间）。

从终端用户的角度看，关键的性能属性是响应时间。在我们的经验中，用户倾向于通过感觉到的响应速度来测量应用程序的性能。非常有趣的是，我们发现在决定用户对应用程序质量的感知方面，真正的功能或者是易于使用的趋势所产生的影响要比响应时间小。我们将响应时间看做表达应用程序性能必需的一个基本的关键统计量。

响应时间直接受到同时与应用程序交互的用户数的影响。因此在处理响应时间时，必须考虑同时发生的用户数量。从操作员的角度看，他们需要一些信息，帮助他们按照容量规划目标确定应用程序的大小。他们主要想知道在硬件资源被耗尽之前，应用程序能够处理的用户数量。随着用户负载的增加，测试应该指出工作繁忙的硬件系统组件。这可以反过来告知如何在应用程序服务器、数据库服务器和网络之间最佳地分割硬件预算资源。此外，该信息还能够帮助确定最优的部署配置。前端（servlet和JSP）可以运行在一个应用程序服务器上，而事务逻辑（EJB、JMS队列等等）运行在另一个服务器上。这些实例可能运行在同一台计算机上或者是分开的计算机上，所有计算机都可以使用相同或者是不同的操作系统。在这个类别中经常问到的问题就是可以在多处理器计算机中使用的应用程序服务器的实例数量。

基于这种讨论，我们可以容易地推断需要一个测度来反映这两个度量标准：同时发生的用户数量和响应时间。实际上，在刚开始开发我们的方法时，我们就曾设法发明一种新的统计量，以一种简单的方式表达该信息，但是这被证明是徒劳的。

作为替代，我们的方法采用了简洁短小的性能陈述，可方便地传达应用程序的能力与限制（limit）。例如，如果最大响应时间是1000毫秒，那么我们可以达成如下陈述：

> 该应用程序能够以1000毫秒的最大响应时间处理750个同时发生的活跃用户；峰值时刻有800用户，响应时间下降7%。

我们强烈地感觉到类似这样的简单陈述，正确地传达了整个被测应用程序的性能和界限情况，同时最小化了可能的误解与混淆。该陈述还描述了响应时间被超过时观察到的行为。在进行性能规划时，这是非常有价值的信息。

后端应用程序

当应用程序的主要接口是面向用户的时，基于响应时间和用户数的性能陈述是有意义的。然而，当应用程序具有与另一个系统的接口时，例如一个需要处理的JMS消息队列，这样的度量标准可能就不太适用。

当我们详细研究批处理或后端应用程序的需求时，我们发现最有用的、使用得最普遍的性能统计量是吞吐量。表达吞吐量性能最流行的方式之一是每秒的事务处理（Transactions Per Second，TPS）。但是如果没有完全准确地理解其含义以及所测量的内容，那么该度量标准可能导致一些问题。

一个常见的误解是TPS中的T真正代表的含义。什么是事务处理？人们通常不能理解在他们的上下文环境中事务处理是什么。每个人都认为他们在讨论同一件事情，而在实践时却不是。我们就遇到两组开发人员并发同一个项目却使用不同的“事务处理”定义。一个组将其定义为单个查询，而另一组则将其定义为一组特定的查询。有时候，操作组将以完全不同的方式定义它！这当然会导致混淆，结果可能是应用程序按照某个组的定义满足性能需求，而不是另一个组的需求。

当分析J2EE API时，我们在本书中广泛地使用了吞吐量度量标准。我们认为它为系统吞吐量提供了一个有用的比较测度，前提是清楚地说明了上下文。在研究servlet时，我们定义事务处理为一个请求——因此吞吐量是servlet在一个设定的时间周期内（一秒）执行的同样请求的数量。当分析JMS时，吞吐量就是消息（message），依次类推。在存在清晰的事务处理概念的地方，并且测试条件是不变的，TPS能够就某些代码修改是否改善或降低应用程序模块的相对性能提供有帮助的指导。

然而，我们必须提醒读者有关该度量标准的几个注意事项。吞吐量不是一个速度测度，它差不多是一个容量（capacity）测度——这是一个常见的误解。它不是一个像响应时间那样原始的清晰的度量标准。测试条件的小改变可能对获取的吞吐量数值产生巨大影响——而且它并不一定意味着应用程序执行得更好或更糟。

吞吐量并不总是提供应用程序性能的完整描述。一个应用程序可能执行10000 TPS，但是终端用户的响应时间是什么？用户是否需要等20分钟才能得到一个响应，或者仅仅是300毫秒？这样，为了提供完整的性能描述，吞吐量通常与其他测度一起使用，特别是在测试交互式应用程序的时候。

最后，该度量标准本身并不传达计算机上负载的情况。也就是说，它没有告诉我们该计算机的容量是否100%地使用，还是仅仅使用了20%。我们认为这样的测度应该与吞吐量一起使用。

上下文测试方法

本书标题为“J2EE性能测试”，看起来非常直观，但实际上这是非常仔细选择的结果，以便最好地反映其内容的真正意图与焦点。为了理解这一说法，你只需要考虑一下经常使用的那些与应用程序软件性能相关的诸多表达就可以，如测试、基准测试（benchmarking）、轮廓（profiling）和调整（tuning）等等，此处只列举少数几个。尽管它们似乎经常交替使用，但是这些术语中的每一个都有明确的含义。某种程度上很容易理解为什么会这样，因为每个人对在任何特定的时刻哪方面的性能与他的工作最相关都有他自己的看法。

在本书中，性能测试的含义是使用一种规定的方法收集你自己的应用程序的性能统计数字。但是，了解其他这类术语的含义并讨论在这些领域中如何使用本书的方法是有帮助的：

- **基准测试（Benchmarking）**——一般来讲，基准测试是在各种不同的环境和工作负载下记录应用程序性能的过程。
- **轮廓（Profiling）**——这涉及到精确地调查应用程序将大部分计算周期花费在什么地方，以及应用程序如何高效地使用系统资源。
- **调整（Tuning）**——测试、基准测试和轮廓都反馈给调整过程，后者是优化应用程

序和环境获取最大性能的过程。

基准测试

我们的工具集和方法可以用来基准测试一个应用程序。就像前面所描述的那样，在开发应用程序的时候，你可能在一种配置中测试它，然后再调换一个组件重新测试，或者是进行一种环境调整（比如增加JVM的堆栈空间）然后再测试。这就是基本的基准测试过程。事实上，我曾经考虑将本书称为“J2EE性能基准测试”，但是我们最终决定放弃该术语，以免将本书要着手达到的目标与行业基准测试相混淆，比如ECperf （http://java.sun.com/j2ee/ecperf/）和TPC-W （http://www.tpc.org/）。

行业基准测试的问题是，它们是非常明确的应用程序，试图测量类似于J2EE那样的产品或基础结构的一般特性。而我们强调不能从一个应用程序推断另一个应用程序的性能结果，不管它们是多么的相似。

例如，ECperf是官方的J2EE基准测试。那是一个复杂的EJB应用程序，为测量J2EE应用程序服务器的性能和伸缩性而设计的。它主要面对的是J2EE应用程序服务器供应商（如BEA Systems、IBM和iPlanet），以便他们能够用它来现实他们产品的性能。

我们在附录C中为感兴趣的读者提供了更全面的关于我们的工具和方法与ECperf的比较。

其他基准测试套件，例如TPC-W（Transaction Processing Performance Council - Web e-Commerce，事务处理性能委员会——Web e-Commerce分会），测量与Web服务器吞吐量相关的性能。这对于开发应用程序服务器的工程师可能有帮助，并且可能在销售该服务器的时候用得着，这有点像数据库公式使用TPC事务处理基准那样。最终它可能有助于购买的决策过程，但是一般不能提供对J2EE性能状态的合理意见。

此外，我们强调不应使用任何一般的基准测试结果来猜测你的应用程序的性能状态。组成应用程序的每个子系统都对其性能有直接影响，为了真正了解其性能你必须测试你的应用程序。

轮廓

就像上面描述的那样，轮廓在本质上是对应用程序正在进行的工作及其如何使用系统资源的深入分析。主要目标是突出系统中潜在的性能瓶颈。我们的测试方法将使你能够确定应用程序中具体的限制性质的区域。然后可以使用一个免费赠送的轮廓工具详细分析这些区域。在本书中除了The Grinder外，我们还使用了各种各样的工具来研究系统的每个组成部分：

- **数据库**——在Oracle数据库中，我们使用轮廓工具SQL_TRACE和TKPROF。两者都是作为标准提供的，是输出应用程序执行的SQL语句以及该SQL的执行方式。
- **WebLogic服务器**——我们使用WebLogic控制台详细查看WLS的内部情况。
- **J2EE应用程序**——我们利用一个称为Introscope的工具，该工具来自Wily Technology（http://www.wilytech.com），用来帮助准确查明应用程序中组件级的瓶颈。另一个

流行的J2EE应用程序轮廓工具是JProbe（http://www.sitraka.com/software/jprobe/）。

调整

一个典型的J2EE应用程序将建立在一个应用程序服务器的基础上，但这不是组成应用程序惟一的子系统。此外，还有数据库、Java虚拟机（Java Virtual Machines，JVM）、操作系统、TCP/IP堆栈、Web服务器、网络、路由器和现行的计算机硬件。所有这些子系统都有调节器或参数，可以调整它们以最大化应用程序的性能。在本书中我们将使用利用我们的方法初步测试的结果来调整环境的各个方面，比如JVM和WebLogic服务器本身。这给予我们最适宜的环境，以便运行正规的性能测试。

在我们根据测试测度和相关的轮廓提供应用程序级的性能改善建议时，我们并没有承担任何实际的调整工作，因为那不是本书的焦点所在。

本书的使用

本书的基本原则是：你自己的特定的J2EE应用程序是独一无二的，你必须亲自测试它，我们将提供让你这样做的工具集和方法，而且那只需要你付出这本书的价钱！

虽然我们将给出原始性能数据，但实际上我们将集中于比较的性能——原因是原始的性能数字高度依赖于应用程序服务器、JVM以及底层的操作系统和硬件。我们想要给开发人员、系统管理员和应用程序建造师提供比较后的代价，这种代价是使用在每个J2EE API中都可用的不同特性而付出的。在我们的方法以及从中获取的数据的基础上，你将能够做出有见识的业务决定，或者是帮助潜在的顾客做出他们的决定。在给定可供选择的办法时，你将能够实际地评定使用某个特定功能的相对代价是否值得那个价格。你将能够掌握你自己的应用程序的性能。

在本书所有测试中使用的代码都可以从Expert Press的Web站点（http://www.expert-press.com）上得到，因此可以在你自己的环境中运行性能测试。但请始终记住本书的意图不是让你从原始性能数据推断有关你自己应用程序原始性能的某些假设。我们所能做的是建立一个一般的期望标准——其思想是为你自己独特的应用程序获取该数据。

本书的组织

下面的清单给出了本书剩余部分的组织结构：

- 第1章详细描述我们提出的测试方法。
- 第2章是我们在本书中使用的负载生成工具The Grinder的用户手册。
- 第3章是一个详细的关于如何获取交互式应用程序性能统计数据的实例研究。
- 第4章探索HTTP协议和Servlet API用途的各个方面，研究可供选择的不同方法，并探讨使用它们的代价。
- 第5章考虑了各种不同EJB设计模式的性能，并探讨了将它们应用于例子应用程序JazzCat时的相关性和代价。
- 第6章分析JMS点到点模型，略微谈到消息大小的影响、消息应答的用途以及使用消息持久性的效果。

- 第7章使用一些建立在最常用主题上的实例探讨JMS出版与订阅模型。
- 附录A包含The Grinder的参数清单。
- 附录B描述了在本书测试中使用的计算机、操作系统和软件。
- 附录C是本书提供的测试方法和工具集与ECperf基准测试应用程序的比较。

除了一些设置指令的交叉引用外，为了避免不必要的重复，描述性能研究的章节（第3章以后）是单独设计的，可以按任何顺序阅读。然而，为了完全掌握该测试方法和工具集并了解它们能做什么不能做什么而依次阅读本简介、第1章和第2章是很重要的。

客户支持

如果你希望直接查询本书的问题，那么请发电子邮件给support@expert-press.com，并在电子邮件的主题中带上本书的标题和ISBN号的后四位数字。

欢迎与我们联系

为了方便与我们联系，我们已开通了网站（www.medias.com.cn）。您可以在本网站上了解我们的新书介绍，并可通过读者留言簿直接与我们沟通，欢迎您向我们提出您的想法和建议。

目 录