分类号：TP311.5　　单位代码：10335

密　级：无　　学　　号：Z134325199

浙江大学

硕士学位论文

中文论文题目 ：基于 web 的医疗保险信息管理系统的性能测试与分析

英文论文题目：Performance Test and Analysis of Policy-Holder Medical Insurance Information System Based on Web

申请人姓名：王艳灵

指导教师：李善平 教授

合作导师：

专业学位类别：工程硕士

专业学位领域：软件工程

所在学院：软件学院

论文提交日期 2015 年 04 月 20 日

题目

作者姓名　浙江大学

基于 Web 的医疗保险信息管理系统的性能测试与分析

论文作者签名:_____

指导教师签名:_____

论文评阅人 1：_____

评阅人 2：_____

评阅人 3：_____

评阅人 4：_____

评阅人 5：_____

答辩委员会主席：_____

委员 1：_____

委员 2：_____

委员 3：_____

委员 4：_____

委员 5：_____

答　　辩　　日　　期　　：

Performance Test and Analysis of Policy-Holder Medical Insurance Information System Based on Web

Author's signature:_____

Supervisor's signature:_____

Thesis reviewer 1：_____

Thesis reviewer 2：_____

Thesis reviewer 3：_____

Thesis reviewer 4：_____

Thesis reviewer 5：_____

Chair：_____
(Committee of oral defence)

Committeeman 1：_____

Committeeman 2：_____

Committeeman 3：_____

Committeeman 4：_____

Committeeman 5：_____

Date of oral defence：_____

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 **浙江大学** 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解 **浙江大学** 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 **浙江大学** 可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

摘要

Web 应用系统采用了浏览器/服务器模式与用户进行信息交互，它以其便捷性和灵活性而被广泛的使用。该系统一般可以被分为三层结构，即客户端、web 服务器和数据库，然而随着系统规模不断的扩大以及复杂性的增加，使得每一层次以及之间极易发生故障，为了保证系统在生产环境中能够正确且高效的运行，这就有必要在部署实施之前对其进行软件测试，特别是有必要对其性能进行测试。相比于其他应用系统，Web 系统中的不同的构件有着各自独特的构造设计，而且 Web 系统有其并发性、分布式等特点，这给系统的性能测试增加了新的难题。

本文以基于 web 的医疗保险信息管理系统为基础,探讨性能测试的相关内容，包括：性能测试的概念、性能指标、测试目的及过程、测试工具、国内外性能测试研究现状等。

由于医疗保险信息管理系统模块繁多，所以在性能实施阶段，本文选取了其中的定价 Pricing 子系统为研究对象，采用 JMeter 测试工具对其进行性能测试的实施与分析。测试实施的主要内容包括：选择测试工具、设计测试方案、搭建测试环境、设计测试场景。本文应用了并发瓶颈测试、内存泄露测试、功能性能测试、实际生产环境性能等测试场景，并分别讨论了各种场景中的测试策略、测试目标、测试结果及结论，最终得出系统中存在的性能瓶颈，为系统调优提供依据。

关键词： Web 系统，性能测试，JMeter，响应时间，服务器症状

Abstract

Web application systems use the browser/server mode to conduct information interaction with the users, it was widely used for its convenience and flexibility. The structure of the system generally can be divided into three layers, namely, the client, the web server and the database. However, with the increasing scale and complexity of the system, each level occurs failure extremely easily as well as between. In order to keep the system be able to operate correctly and efficiently in a production environment, which requires testing before the deployment, especially the performance testing. Compared to other client system, the Web system has its characteristics of isomerism, concurrency, distributed, which brings new challenges to the system performance testing.

Based on the project - the medical insurance information management system based on web, studied the related content of the performance test, including: the concept of performance testing, performance indicators, purpose and process of testing, test tools, performance testing research status at home and abroad, and other.

Due to the medical insurance information management system module is complex, this article selects the Pricing subsystem as the research object in the implementation stage of the performance testing, using JMeter as the test tool for the implementation of the performance testing and analysis. The implementation of the performance testing mainly include: choosing testing tools, developing test environment, designing the testing scheme, designing test scenarios. In the paper, there are many scenarios, such as concurrent bottleneck function testing, leak testing, function performance testing, the actual production performance test scenarios and so on. Then discuss the test strategy, test targets, test results and conclusions in the various scenarios. Finally conclude that exist performance bottlenecks in the system to provide the basis for system tuning.

Key Words： Web system, The performance test, JMeter, Response time, Server symptoms

目录

图目录

表目录

第1章 绪论

1.1 课题研究的背景及意义

随着软件行业的发展，其应用规模变得更大。Web 系统由于其便捷性得到了迅速的发展，随着系统用户数量的不断增加，高并发访问的 Web 系统对系统性能的要求会更加的苛刻。整个社会对软件质量的期望值变得更高，软件性能测试就变成了提高软件质量的一个比较重要的保障。基于此，本文结合医疗保险信息系统的特点以及软件测试的管理，研究如何在 web 项目的开发过程中进行性能测试，以此提高软件开发的整体质量。

1.2 医疗保险信息管理系统性能测试的提出

目前，作为和每个人切身利益相关的医疗保险信息系统被大家尤为关注，那么这其中，医疗保险的各项信息的整合和管理就变得比较复杂，传统的手工模式根本无法应付这项工程，于是，医疗保险信息系统就很自然的出现在人们的眼帘。随着各种信息管理系统 web 应用的兴起，基于 B/S 模式的软件最近几年有了快速的发展，而且日益成为将来软件开发部署的趋势，所以，这里的医疗保险信息系统也是基于这种模式的。当开发人员将一个 web 软件开发出来并展示在用户面前时，尤其是在软件将被部署到生产环境之前，用户通常会产生以下的疑问：软件对使用者的请求的响应时间有多大？这个 web 软件可不可以经得起大量并发使用者的同时访问请求？软件系统的整体性能情况如何？系统在长时间的不停地工作下，它的运行状态能够保持稳健与否？系统是否存在性能瓶颈？是什么制约了系统的性能？用户的疑问也正是 web 系统性能测试需要解决的问题，然而如何才能有效地执行 web 软件性能测试，目前还没有一个系统和完整的解决方案[1]。那么，如何对一个基于 web 的医疗保险信息管理系统有效的进行性能测试，更是没有一个系统而完整的答案。此外，由于紧凑的开发计划和复杂的系统架构，web 应用的测试经常是被忽视的[2]。所以开发的软件质量有好有坏，参差不齐，严重的会产生软件危机。如果想对这些软件的质量进行把关，或者说尽可能的提高软件质量，就需要在软件开发过程中对软件进行测试，因此一些企业或者软件开发人员会对进行测试。即使进行了测试，其关注点也主要放在功能测试上，但功能

测试只是对系统的基本功能做相关的测试，并没有考虑系统性能相关问题，一旦因为性能问题系统崩溃，将会给用户带来很不愉快的上网体验，也会因此对相关机构和企业产生一定损失。然而，随着软件的发展，人们对 web 系统的性能测试更加的看重，web 系统的性能已经变成系统测试中一个及其不可缺少的部分。

针对基于 web 的医疗保险信息管理系统的性能测试的专门研究又不是很系统，因此，本文的研究就是源于上述这种需求，为 web 的医疗保险信息管理系统进行基于目标的性能测试，从而同时尝试进行一些性能测试的研究，这些内容是很有意义的。

1.3 论文研究的目的

本文以 JMeter 为测试工具，对基于 web 的医疗保险信息管理系统进行性能测试，以发现性能瓶颈，以提出提高其质量的调优方案。

研究目标是：结合基于 web 的医疗保险信息管理系统的特征，不仅用测试工具执行测试脚本来验证系统是否能够达到软件性能指标的要求，更重要的是能够在软件部署之前发现软件性能缺陷，定位系统瓶颈，并能够进行系统调优解决系统缺陷，后者才是软件性能测试的真正目的。本文对医疗保险信息管理系统的性能测试目标可分为以下两种层次：

（1）性能测试总体目标。检验医疗保险信息管理系统是不是可以满足用户的性能指标的需求，找出其中可能存在的缺陷和性能瓶颈。运用测试工具进行性能测试并收集整理测试结果，分析产生缺陷的原因，提交总结报告，为软件开发人员对该系统性能调优提供依据。

（2）性能测试具体目标。此目标又可以分为：

①需要明确系统的总体性能指标参数，具体含有：交互的响应时间、可以承受的最多请求并发用户数、系统吞吐量等；

②能够明确系统在不同程度的负载及压力下，各个服务器的详细性能数据；

③判定系统的耐力强度；

以上的这些性能测试目标指导整个测试过程的执行，在此系统性能测试中是至关重要的。

1.4 论文研究的主要内容

本文主要是研究的是使用JMeter针对基于web的医疗保险信息系统性能进行测试，找出系统中需要改进的地方，使系统尽可能的完美。

本文主要内容包括：

1、业务场景选取

熟悉系统应用，根据系统开发的逻辑来了解整个系统的架构，这样才能知道测试中需要模拟的应该是什么样的请求。

了解和分析测试需求，确定需要的业务场景和和性能指标。

确定测试业务场景：针对客户方对软件系统的要求，分析系统架构以及业务流程，确定系统性能测试中需要关注的重点，主要包括系统的响应时间评估、系统并发瓶颈、系统内存泄露以及系统吞吐量等，其中还需要检测各个服务器运行的指标参数。

分析总体的业务流程，性能测试中需要覆盖主要的业务流程，这些流程中尤其需要覆盖各种的不同类型的业务以及操作，选取的测试业务场景需要具有代表性，例如：数据库中数据的查找，增加，修改，删除等操作。

分析性能测试指标：

（1）在各种测试场景确定之后，就需要根据这些场景选取系统性能的指标参数，主要包括：请求响应时间、最大并发用户数、服务器中资源（CPU、内存、硬盘灯）的使用率、系统吞吐量等；

（2）在确定了性能参数指标之后，就需要与用户沟通，凭借用户的需求以及经验确定系统中每个指标参数的值，如：用户对响应时间的期望值、系统能够承载的并发量以及在负载情况下服务器的硬件参数。

由于医疗保险信息系统的功能模块很多，基于论文篇幅原因，本文只考虑使用其中的定价模块作为研究对象，研究该模块测试场景的选取以及性能指标的确定等，性能测试指标考虑使用响应时间以及吞吐量。

2、性能测试实施

录制脚本、修改脚本参数、调试和执行脚本。

本文考虑，对定价模块可以从几个方面进行测试，如：并发瓶颈、内存泄露、功能性能等的测试。

3、性能测试结果分析

分析测试结果，获得执行脚本过程中所收集的性能指标值的分析报告，可以通过分析报告中的数据，得知系统的性能情况。

最后，本文在总结性能测试结果，并分析性能瓶颈，得出建议优化措施，进而提高该系统的性能，使软件质量得到进一步的保障。

1.5 论文的组织结构

论文一共分成 6 个章节。

第一章，概述课题研究的背景和意义、医疗保险信息管理系统性能测试的提出、课题的研究目的、研究内容和本文组织结构。

第二章，首先概述了软件性能及相关的软件性能测试的概念、目的、过程、web 系统性能测试，然后介绍了软件测试国内外现状、软件性能测试方法，最后介绍了 QALoad、SilkPerformer 等软件测试工具。

第三章，对基于 web 的医疗保险信息系统进行概述，并分析分析系统性能测试的瓶颈，并对此系统提出相应的需求。

第四章，详细介绍了性能测试的实施，首先简要描述了性能测试工具选择以及性能测试平台，然后详细叙述了测试的方案制定、环境设计、场景设计、场景的执行等。

第五章，对系统进行并发瓶颈、内存泄漏、功能性能、实际生产环境性能、排除网络传输的性能测试。

第六章，对本项目作相关的总结。

第2章 软件性能测试的研究

2.1 软件性能测试概述

2.1.1 软件性能概述

系统的性能是个不小的概念，覆盖范围特别宽广，它是指软件所拥有的一种非功能性的特征，它关注的不是软件能否实现某一个功能，而是指软件在完成实现这些个功能之上所表现出来的效率、及时性等能力。而对于软件性能的感受是主观的概念，由于每一个人的主体不一样，其对软件性能的感受以及关注的角度也不一样。

具体来说，软件性能包括软件系统的资源消耗、系统安全性、系统稳定程度、系统的可靠性以及稳定性等等，也即是性能是软件系统在所有参数指标方面的整体表现情况的评判。评判性能的指标参数很多，用户一般会使用请求响应时间、系统吞吐量等参数进行评估。而且对 Web 应用系统来说，系统的性能是软件质量中及其关键的。在软件行业通常对软件的性能概念中，具体说明以下六条指标：

①响应时间

响应时间是指软件在用户请求后，所执行处理请求所需要的时间[3]。这一指标也与每个人的直观感受有关，软件系统的会存在许多不同的功能模块，它们的处理逻辑不相同，它们对用户的相应时间也不一致，况且即使同一个功能由于输入参数的不同，也会产生响应时间不相同的现象。因而在衡量这一指标时，一般指的是系统中所有功能模块的平均响应时间或者最大响应时间。但是需要说明的是，响应时间绝对值不能直接反应系统性能的高低程度，这一指标还取决于用户对系统响应时间的接受程度。对于不同的软件系统，例如游戏软件和编译软件，响应时间同样的都是一分钟，但是对于游戏这样的响应时间无疑用户是不能接受的，但是对于编译软件来说，这样的响应时间甚至表明该软件系统性能很高。

在基于 web 的 B/S 架构软件系统中，软件开发仅仅只是在服务器端，呈现给用户的响应时间包含了服务器响应时间以及浏览器呈现服务器数据所用的时间。后者与浏览器以及用户所处的计算机配置有关，而软件测试中更关注的是前者，而前者还包括了延迟时间以及网络传输时间。

②应用延迟时间

应用延迟时间指的是软件系统处理用户请求的真实时间，如图 2.1 所示响应时间=网络传输时间+应用延迟时间，即（N1+N2+N3）+（A1+A2+A3），而其中系统延迟时间包括 A1、A2、A3 时间总和。因此应用延迟时间更能评价软件性能。

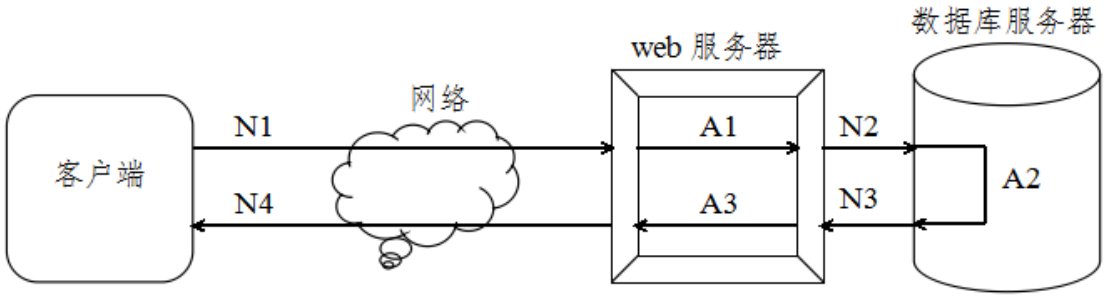


图 2.1 应用响应时间及延迟时间

③吞吐量

吞吐量指的是软件在单位时间中能够处理请求的数量。在并发多用户使用的系统中，如果单个用户的平均响应时间是 t，那么 n 个用户使用系统时，此时的平均响应时间不会是 n*t，而会远远小于这个数值。不同系统的平均响应时间随着用户量增加而增长的趋势各不相同，所以采用吞吐量来衡量软件系统的性能。

④并发用户数

并发用户数即软件系统能够同时承受的最大用户数目。与这个概念有关联的还有系统用户数以及同时在线用户数。其实，服务器所受的压力不止取决于并发用户量还与业务场景有关。以下计算公式能够衡量这一指标：

公式（2.1）平均并发用户数 $C = nL / T$

公式（2.1）并发用户数峰值 $C' \approx C + 3\sqrt{C}$

其中公式（2.1）中，C 代表使用者并发量的期望值，n 代表 Login Session 的个数，L 代表 Login Session 长度的期望值，T 表示观测的时间间隔大小。

其中公式中（2.2），C'表示使用者并发量的峰值，这个公式设想使用者 Login Session 与泊松分布向吻合。

⑤资源使用率

资源使用率表示是一段时间内系统资源平均被占用比列。对于数量只为一的资源，资源利用率指的是资源被占用的时间与整段时间的比例，对于数量不唯一的资源，资源利用率指的是一段时间内所使用资源的数量与所有资源数量的比例。

⑥每秒请求数目和会话数目

每秒会话请求数目是指每秒钟进入 web 服务器中 GET 或者 POST 请求的数目。每秒会话数目是指每秒钟到达并访问网站的用户数量。

对于不同的软件角色，如：系统管理员、软件开发人员、用户等，他们所关注软件性能侧重面也不一样，具体不同点如下所述：

从使用者角度来说，他们关注的是系统对使用者所发送请求产生答复所使用时间；

从管理员角度来说，他会关注应用延迟时间、应用服务器的运行状态、系统并行用户数以及峰值、系统瓶颈等软件性能指标；

在开发人员方面考虑，在系统性能方面，他们比较在意软件系统对用户的响应时间、软件在性能方面的扩展性等，具体而言开发人员更加在意“如何通过调整代码之间的结构以及采用编码方式来提高软件性能”等，如表 2.1 所示：

表 2.1 软件研发者所关心的软件性能

软件研发人员关注的方面	所属环节
架构设计合理吗?	系统架构
数据库设计合理吗?	数据库设计
是否存在影响系统性能表现的代码?	代码
内存使用方式是否合理?	代码
线程同步方式是否合理?	代码设计
系统资源竞争是否合理?	代码设计

2.1.2 软件性能测试概述

软件性能测试能够在狭义以及广义两个角度进行描述，狭义概念上的性能测试表示在正常以及极端的运行负载以及使用环境中，采用成熟的自动化的测试技术及工具，模拟多重正常、峰值以及不寻常压力情况下应用的各性能参数是否能够满足生产性能的需求而执行的一项测试。从广义概念上将，性能测试则表示强度测试、压力测试、并发测试、可靠性测试、负载测试、配置测试、容量测试等，其中最有意义的是负载测试和压力测试。在处理软件性能瓶颈以及提升软件质量的各种方式中，测试软件性能测试是非常有效的手段，如今人们对软件性能测试已经有了越来越深入的了解。从本质上讲，软件性能测试是指一种标准，在软件部署到实际生产环境，通过使用相关的测试工具检测出系统中的错误以及应用瓶颈，为系统调优提供依据。

迄今为止，在提升系统性能的方式中，性能测试依然是很高效的方法，它在软件的性能指标参数检验、软件不足缺陷修复以及软件性能调优等方面都起到了着举足轻重的效果。性能测试原理如图 2.2 所示：

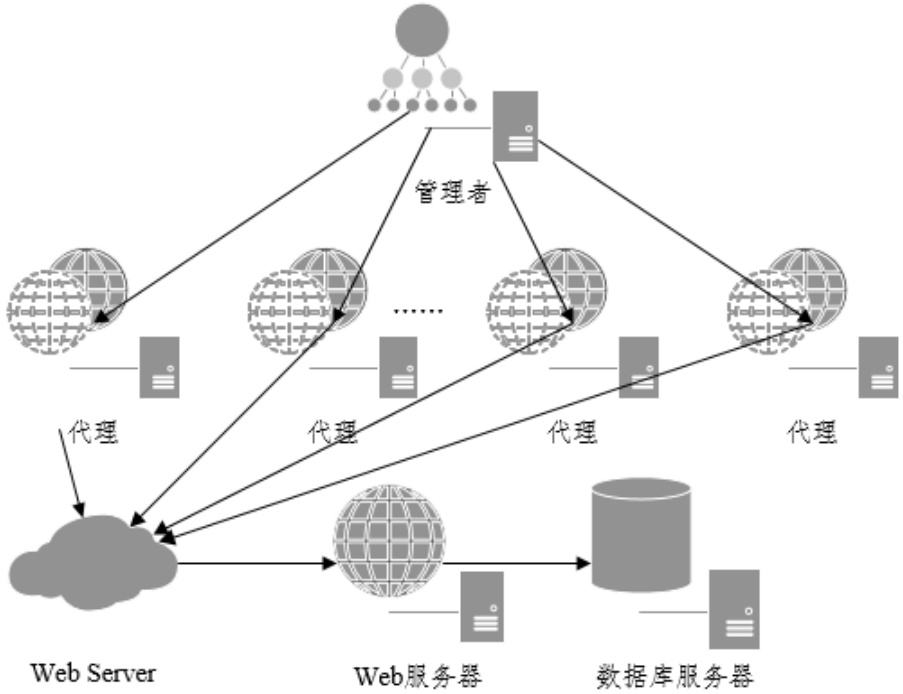


图 2.2 性能测试原理图

2.1.2.1 性能测试目的

性能测试目标能够划分成两个层次：基本目标和高级目标。

其中基本目标包括两个方面：检验应用的性能和检测软件中的问题。评估系统软件性能指的是测试软件系统在用户正常使用环境下的各个性能指标参数的状况以及应用能否达到用户的需求指标，主要有 HTTP 响应时间、应用延迟时间、吞吐量等指标。检测系统中的问题指的是检测出测试过程中系统出现的一些缺陷，如：内存泄露、数据存储等隐含缺陷[4]。

高级目标同样包含两个部分：检测系统中的缺陷瓶颈和系统调优。经过性能测试发现软件中隐含的性能缺陷以及瓶颈，明确这些指标能否满足了使用者的性能需要，收集性能测试结果以及从中研究出造成这些性能问题的因素，撰写性能测试报告，为软件开发人员提交软件改进以及性能优化的依据建议。基于以上的测试结果，软件研发人员能够根据测试中软件所表现出来的瓶颈，针对性地调整优化。

2.1.2.2 性能测试过程

根据软件工程的方法，软件性能测试可以分成五个步骤，如图 2.3 所示：

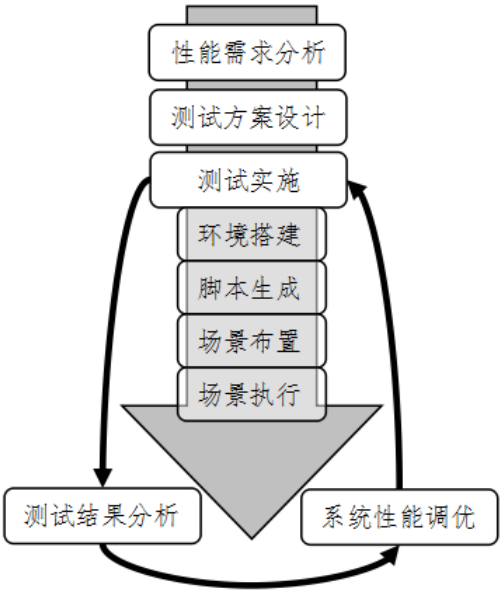


图 2.3 性能测试过程

1、性能需求分析阶段

该阶段的基本的过程有通过需求分析收集测试信息以及确定测试需求。该阶段是测试过程中很重要的一个阶段，它直接与性能测试的有效性相关联。测试需求可以从相关的文档中获取，所以在这个过程中，通过向软件开发方或者开发人员获取系统需求说明书，用户使用手册等以及通过当面交谈咨询提问的方式获取软件测试信息[5]，然后对收集到的信息进行过滤、筛选、排除，确定合格规范的测试需求，最终撰写出软件性能测试需求说明书。

性能测试需求分析基本上包括两个内容：一、确定测试目的；二、确定性能测试指标。其中测试目标包括三类：验证性能是否符合需求？检验软件性能的能力如何？以及进行软件优化以提高性能。其中性能符合性验证是检验系统是否满足设定的目标以及用户的应用要求。性能验证即是熟悉推断出系统的全部性能指标状况。确定性能指标主要是为后续测试提供一个标准，从而判断测试结果以及测试的结束[6]。

2、测试方案设计阶段

该阶段包括两个方面：确定测试场景以及监视指标。其中确定测试场景是在上一个阶段工作成果的前提之下，根据不同的测试需求，选取相应的测试业务，

并采用场景设计方法确定实际测试业务场景，包括实际应用情况，如：主要的业务流程、每个业务操作的概率及用户量等，其中在设计测试场景时需要主要思考的因素包括响应时间、并发用户数以及硬件使用情况率等参数。

3、测试实施阶段

该阶段主要是把上一个阶段的软件测试场景付诸于实施，然后通过分析确定系统缺点瓶颈。该阶段可以细分成两个部分：测试实施以及性能瓶颈分析。

其中，在测试实施阶段主要包括选用测试场景进行压力测试、负载测试等，同时还需要对系统资源进行监控。该过程还包含了测试之前的准备工作，如：创建测试环境、生成测试脚本、设计测试场景以及实施测试场计划[7]。

（1）测试环境搭建及数据准备

这里的测试环境涵盖了软件、硬件环境以及测试数据。为了测试结果的准确性，必需创建不受影响单独、真实的软硬件环境和网络环境并且配置测试软件。软件测试数据能够自身选取也能够让使用方提供，但是必须确保测试数据的合法规范，要避免无效不合法的垃圾数据，而且测试数据必须要包含了完整的业务流程的全部方面，以下就环境搭建和数据准备两个方面具体分析：

①环境搭建

测试环境搭建尽可能地与实际中的生产环境大体上一样，这样测试结果才有参考价值。

本系统的测试环境需要独立于其他的开发环境等，即需要确保被测试的系统在测试执行期间占用了测试环境的一切资源，需要杜绝测试环境与其他环境的混合使用，如开发人员使用测试环境进行开发测试或者开发环境中的网络公共使用测试环境的网络。在这些情况下，会干扰测试的执行进度，从而导致测试数据结果的不准确，甚至糟糕的是误导后续的系统调优，致使性能问题。综上所述，本系统所搭建的测试需要具有资源独占性。

同时还需要注意，随着系统测试的执行，期间测试环境可能会发生不断的改变，例如：测试人员的增加减少、软件更新了新的版本等，所以在搭建的测试环境的时候，需要考虑测试环境是可以重复使用的。然而，测试中的某些资源是可以有效控制的，这需要对其进行分类管理，这些资源包括：测试环境中的硬件资源信息、测试环境中的软硬件配置手册等。对这些可以控制资源的管理备份，目的在于当测试环境出现调整或者故障时，能够便于迅速还原测试环境。搭建完成之后的测试环境在软件测试过程中、测试软件安装之前都有必要对其进行备份，

同时还需要备份操作系统，在这样操作之后，会带来两个好处：1、当软件新版本发布，需要进行下一轮测试时，就可以直接还原测试环境，免得重新搭建测试环境带来的时间人力消耗；2、在测试过程中出现测试故障以及在现有测试环境中已经不能再进行测试时，可以进行数据恢复，免得由于丢失数据而造成的损失。

②测试数据准备

在性能测试的实施过程中，需要遍历所有主要的业务流程，而这些流程是由输入的参数数据所驱动，所以其间不可避免的就需要涉及到业务数据。况且数据的有效性以及合理性直接影响了性能测试的效果，所以数据的选取清洗、数据准备在测试之前就需要进行充分地考虑，而且这些数据要尽量地与真实的业务数据相吻合，而这些数据主要包括以下几种类型。

系统用户数据：真实使用者登录系统的用户账号、口令等；

业务数据：测试工具产生的虚拟用户需要模拟真实用户的实际流程动作，这个期间所使用的数据；

可重用数据：这类数据指的是虚拟用户在操作中只需要使用一次的随机数据，包括一些查询类的数据。这些数据可以重复的使用，所以在准备数据时，它们可以只需要设计一次；

不可重用数据：这类数据不可以重复使用，用完一次就不能被再次使用，主要包括一些唯一性标识性质的数据。

（2）测试脚本生成

测试脚本可以自己手工编写还可以由测试工具自动生成，但是必须要求脚本信息的有效性，为了使脚本与实际运行相吻合，还需要对脚本进行参数化、动态数据关联等操作；

（3）测试场景布置

依据设计好的测试场景配置各个测试场景，涵盖了虚拟用户数、性能指标的设置；

（4）测试场景执行

测试场景配置完成之后，就可以按照进行性能测试执行实施了。在测试过程中，有必要根据碰到过多的错误，而终止测试、查找原因。如果是非软件方面的原因，需要及时排除，重新测试；如果是碰到了软件方面的原因，需要记录测试结果，以便后续的结果分析。需要注意的时，在测试实施之前，需要将测试环境初始化到原始最初的状态。

在性能瓶颈分析部分，比较研究测试结果以及标准目标，明确软件系统性能

是否具有性能瓶颈。系统瓶颈主要包括三个方面软件代码级别、软件配置级别和操作系统级别。其中代码级别包括内存堆栈使用是否溢出、资源获取是否死锁等信息；软件配置级别包括数据库连接数以及内存分配等配置、虚拟机配置等信息；操作系统级别包括服务器的硬盘读写速度、内存大小、CPU 等。

4、测试结果分析阶段

该阶段为测试的所有环节中关键的环节，同时也为难点。首先，查看软件测试结果中是否出现错误以及超时信息，如果出现了，需要进一步查找原因，如：程序代码、算法、设置参数问题、硬件问题。

5、性能调优阶段

根据上诉软件测试阶段的实施以及瓶颈分析步骤之后，本阶段主要是根据系统调优的方式，调整优化系统性能瓶颈，修正性能缺陷。同样，与性能瓶颈相一致，性能调优也涵盖了三个方面：它们为代码级别调优、软件配置级别调优、操作系统级别调优。其中，操作系统级别调优主要检查操作系统配置是否合理，如果通过操作系统配置还不能提高性能，则需要考虑升级系统硬件。软件配置调优主要是通过合理配置应用系统中的数据库连接数或者虚拟机内存等参数来提高系统。代码级别的调优主要是通过组织代码编写结构、修补造成内存泄露的代码等方式来消除代码执行效率低的问题[8]。

2.1.2.3 Web 系统性能测试

随着互联网的发展，基于 web 的各种网络应用正以其广泛性和快捷易用性改变了人们的生活工作方式，而变得越来越受到企业的青睐。web 应用部署非常便利，而且用户只需要通过浏览器客户端即可以访问 web 系统应用，所以 web 应用系统正逐步取代以前的客户端/服务器（C/S）结构的应用。随着 web 软件的负责性以及规模的不断增加，如果应用的访问量不断增长，会使得 web 应用的负载急剧加重从而反应迟缓，严重的甚至网站瘫痪，这极度的影响了系统的正常运行以及用户的体验感受。因此，这就需要在 web 应用上线发布之前，利用性能测试技术及工具对其性能和行为进行测试，评估其负载能力、可靠性和稳定性。这样不仅可以增强系统的性能，而且还能够加快开发的进度，让用户对系统充满信心。由于 web 系统与传统的软件系统存在着很大的不同，所以对它们的测试也会有所不同，而且 web 系统的测试要比传统软件系统的测试要复杂很多。因此，对 web 系统的测试是一项艰巨重要的任务，而且该过程已经成为了 web 系统开发中一个

及其重要的组成部分，它直接决定了系统是否能够上线运行。

采用测试工具，Web 性能测试需要虚拟出一定数量的虚拟用户来仿照真实用户在被测试系统中的操作，观测系统在正常使用以及负载使用情况下各项指标参数的变化情况，从而可以推测出被测系统在一些场景中的性能表现以及确定系统的性能瓶颈，为后期的系统调优提供依据。Web 性能测试的目的在于测试系统在并发请求的情况下系统的所能承受的最大并发量，可以察觉系统的性能缺陷，从而为系统调优提供依据，这样能够确保系统在生产环境中的正常稳健的运行。同前文所述，web 系统性能指标有以下几点：响应时间、吞吐量、用户放弃率、用户行为。其中的前两点在前文中已经描述过，此处不再赘述。用户对一个 web 软件系统是否成功最具有话语权，由于每一个用户的需求感受不一样，所以很难有一个确定的答案。然而系统的响应时间是一个直观的反应，是判断用户的可接受程度的一个重要指标。现如今存在八秒定律，即请求访问一个 web 系统，假如使用者等待一个页面的响应延迟多于了八秒钟，会有百分之七十几的用户放弃等待。当然，在 web 系统性能测试中，还会有许多其他的指标，如：页面请求次数等，这些在测试过程中，应当全面充分地考虑这些指标。

2.2 软件测试国内外现状

目前，国外的软件行业把软件测试过程作为系统开发中的关键过程，软件测试以及变成了非常独立的行业。而且在许多国外的一些大型软件企业中，软件测试人员在开发测试人员中所占的比例非常高，一般都是 70%-80%[9]。在 web 性能测试方面，国外的一些大学机构对其研究的成果很显著，这些大学机构主要包括美国的乔治梅森 George Mason 大学、英国的杜伦 Durham 大学等[10]。自动化测试系统已经变得更加的通用化、智能化、标准化，其中上世纪 90 年代以来，以综合通用的自动化测试为主导，替换某一系列的产品[11]。国内外在性能测试探究方面获得了一定的进展并获得了许多的成就，例如：创新了许多性能测试的模型、测试方法以及测试对策等，例如：印孚瑟斯技术 InfosysTechnologies 公司研究的基于对象的测试模型等。从公开源代码免费的测试工具到商用昂贵的测试工具，现如今许多国外公司机构开发了许多性能测试的工具。一些功能完善的工具，它们的开发量很是庞大，所以这些工具一般都是非常昂贵，例如：Compuware 公司的 QALoad；HP Mercury 的 LoadRunner 测试工具，SilkPerformer 最高级的企业级别负载测试工具，RadView 公司的 WebRunner 性能测试与分析工具。

其中，HP LoadRunner 是一款自动化性能测试产品，它能够预先判断系统行为以及性能，该产品来自于为 HP 进行系统负载压力测试而研制的自动化测试产品，它能够检查系统在实际生成负载时的行为以及性能[12]。在 2006 年 11 月份 HP 收购了 Mercury Interactive。通过模拟出成百上千的虚拟用户，LoadRunner 来执行实际中用户在客户端上的动作，例如：IE 浏览器，使用 HTTP 协议发送请求到 IIE 或者 Apache web 服务器，它可以模拟成千上万的并发用户将应用程序通过严酷的真实用户负载以及实时性监控，而从关键的基础设施组件（web 服务器、数据库服务器等）收集信息来进行详细分析探索特定行为的原因，找出系统的问题缺陷。LoadRunner 能够进行多种协议包负载测试：.NET 记录/回放，数据库，DCOM，GUI 虚拟用户，Java 记录/回放，网络，Oracle 电子商务，远程访问，远程桌面，富互联网应用等。它的功能非常优秀齐全，但是价格也非常的不便宜。

免费测测试工具有微软的 Web Application Stress Tool 工具，OpenSTA 等。其中 OpenSTA 是一款功能丰富的基于 GUI 基准实用程序，它可以执行脚本进行 HTTP 和 HTTPS 衡量性能指标的负载测试，它现如今只能运行于 windows 系统中。它的脚本是一个被称为“sci”语言的记录，这是一种相当简单的编码语言，支持自定义函数、变量的作用域和随机或顺序列表。而且测试脚本录制完成之后，还可以对其进行按照指定语法的编辑，以满足特定性能指标的分析，它的丰富的图形化界面，极大的方便了测试结果的展示分析。OpenSTA 是基于 CORBA 的体系结构，它可以虚拟出代理客户，并应用其脚本语言记录通过该代理的所有 HTTP/HTTPS 的通信量，测试人员通过分析 OpenSTA 中的各个性能收集器中的性能指标数据，分析整个应用软件的性能情况。OpenSTA 最初是由 Cyrano 公司所编写开发，它们原本想在 OpenSTA 中编写商业插入模块和支持非 web 应用程序的性能测试，后来由于资金困难，在 2001 年被 Quotium 接管。它的优点就是支持脚本语言而且其压力测试引擎具有可扩展性，可以进行大规模的压力测试。

目前，国内研究软件性能测试的机构还是寥寥几家，例如：中国软件测评中心等，而且这些机构缺乏商业化的操作，而且这些一般是政府部门主导下的机构进行软件验收的工作，其对于软件测试没有实质性的作用。国内的研究软件测试的大学主要有清华大学、东南大学、武汉大学等，并取得了一些成果，其中东南大学在 web 测试框架方面的研究取得了一些成果[13]。在国内的软件开发公司中，软件测试这一过程一直都不太被重视，大部分测试只是停留在功能测试、单元测试。而且软件测试人员的测试技术和规范化程度普遍都不太高[14]。软件测试人员

的人数所占开发测试技术人员总数的比例只有大概 15%。

从一些文献资料来看，国内对性能测试的研究水平还比较低，而且国内研究开发的测试工具数量很少，这或许与国内的软件开发的发展有一定得紧密联系。在国内的软件行业中，软件测试仍然是在一个发展起步的成长阶段，要想超过国外的研究水平仍有相当长的一段路要走。但是在国内，软件测试还属于新兴的行业，其发展速度还是非常快的，相信随着国内软件行业及知识产权保护的发展，性能测试的研究必定会越来越成熟。

2.3 软件性能测试方法

根据性能测试的目的以及内容的不同，常见性能测试方法能够划分成下面几个种类：

强度测试是指在极端匮乏的系统以及网络资源等条件下，监控分析系统的 CPU、内存、硬盘使用量、网络带宽等变化情况。

负载测试是指在用户正常使用软件的情况下，逐步地地给软件加大负载，观察记载研究系统在各个性能指标方面的改变状况。在达到软件各个性能指标需求的条件下，最后明确软件所能够承载负载量的最大值[15]。

压力测试是指在正常使用软件的情况下，逐渐加大软件负载，观察软件各个性能的改变状况，最后明确软件究竟在哪种负载状况下，软件的运行状态会处于失效状态，并以此确定软件能够提供的最大服务级别[16]。

并发测试表示逐步地加大软件的并发用户数量，能够明确软件的性能缺陷和瓶颈以及不能接受的性能点，该测试结合了压力测试和负载测试。

稳定性测试是指软件在正常运行状态下，给应用加载一定的负载压力，测试软件能够在这种状况下正常稳定运行的时间，从而考察软件各个性能参数指标在此压力环境下的能够保持正常的数值，它主要测试检验软件能够长时间稳定运行的能力[17]。

大数据量测试则用于测试当系统处于大数据量的情形下，观测系统的存储、信息查询、数据传输等业务是否能够有明显的不足或者这些方面性能的下降。

容量测试是指软件在极限值状态下，检验软件应用是否能够正常运行或者出现各种软件故障，从而确定软件在一定的时间段内，能够持续运行的最大负载。

2.4 软件性能测试工具介绍

性能测试软件工具能够被划分为如下四种：负载测试软件、故障定位软件、系统调优软件和资源监控软件。其中主要为负载测试工具，以下文中主要介绍主流的负载测试的工具。

负载测试工具的工作原理主要是通过脚本录制、脚本回放、模拟多个用户同时高并发地访问应用系统，这样能够创造出高负载环境，进而能够得出各性能指标的测试数据并对它们进行记录分析，形成测试结果报告，找出系统的瓶颈为系统调优提供依据。

2.4.1 Compuware QALoad

QALoad：它是美国 Compuware 公司研发的客户/服务器（C/S）模式的测试软件，它能够被用于企业资源配置（ERP）以及电子商务应用的自动化负载测试[18]。它能够模拟成千上万个并发的虚拟用户对系统的关键部分进行负载测试，从而能够发现系统中的问题，为系统优化以及成功部署提供保障。QALoad 能够执行负载测试，直到到达了并发数的最大值；它也能够执行强度测试，直到发现系统所能负担的并发数的极值；可以测试响应时间，直观地衡量用户对系统性能的感受。它主要的用途有如下几个：构建出逼真的负载测试、控制中心控制着全局的负载测试、重复执行发现系统瓶颈、整合的系统资源视图等。它支持的协议非常广泛，包括在通信层的 Winsock、IIOP、WWW、WAP；数据层的 ODBC、MS SQL Server、Oracle、DB2、Sybase 等；应用层的 SAP、Uniface、JAVA 等。

它的体系结构如图 2.4 所示：

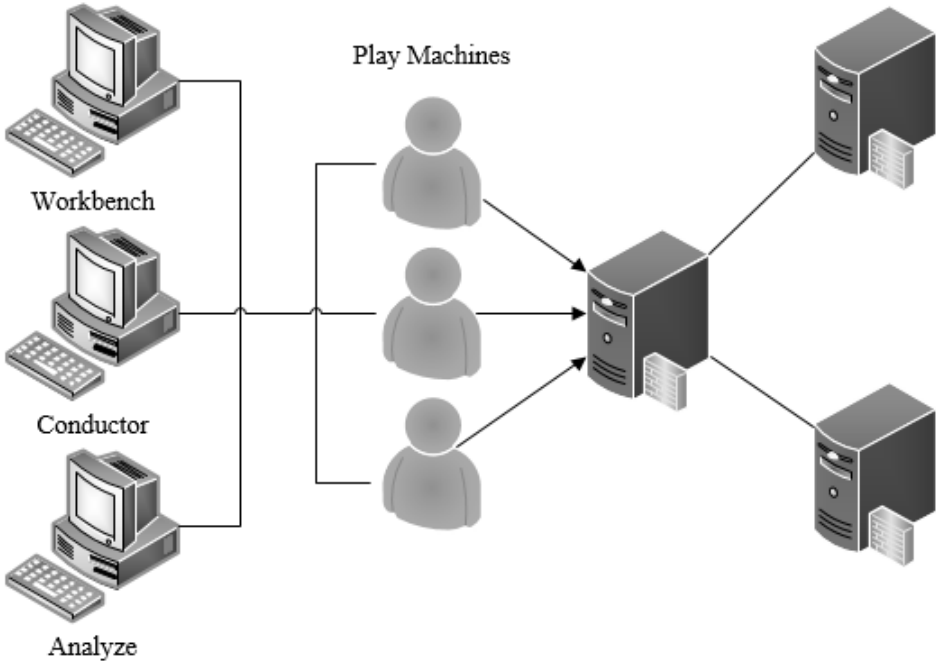


图 2.4 QALoad 体系结构

由图 2.4 的体系结构可知，QALoad 的测试组件由两个部分组成：

QALoad Conductor，它控制所有的测试行为，如：初始化、设置描述文件、生成测试结果，并进行分析生成测试报告。

QALoad Player，创建多个虚拟并发的用户来向服务器发送请求调用。

它的开发组件包括：脚本编辑工具、记录工具、转换工具。

依据可以复用且真实的测试，QALoad 可以完全的检测出系统的性能以及扩展性，它是 QACente 的高性能版本。QACenter 聚集了全部跨企业级别的自动化测试工具，在完整的软件开发周期中，它可以跨越多种平台，自动化地进行测试的任务的执行，QACente 旨在提高软件质量。

2.4.2 SilkPerformer

SilkPerformer 是软件测试行业中最强大以及最容易使用的商业级别负载测试和强度测试工具，它能够检测出系统的性能缺陷并能够调优系统的性能。它使用可视化的脚本生成技术以及能够对拥有成千上万并发用户的应用系统进行性能测试，能够使软件开发公司在应用系统发布部署之前，无需考虑软件规模的大小和复杂程度，对其进行健壮性、稳定性、可扩展性进行全部的性能测试。SilkPerformer 拥有强大的诊断工具和测试报告管理功能，它能够帮助隔离出系统

的故障错误并快速做出处理措施，从而可以从很大的范围内缩短软件测试的周期以及加快软件系统上线的速度。SilkPerformer 允许创建强大现实的负载测试和压力测试用户一系列应用程序环境，包括最新的网络和移动技术。负载测试解决方案主要模拟任何大小的峰值负荷，从而用户没有必要投资负载测试和压力测试的硬件和设置。在发布周期的早期，尽早发现问题能够使开发人员在软件到达最终用户之前纠正问题，减少项目周期时间和开发成本。

SilkPerformer 主要有以下几个功能：

部署多个虚拟用户节点，由单一的中央控制器对它们进行管理并进行实时监控，如图 2.5 所示 SilkPerformer 体系结构；

SilkPerformer 可以将不同的用户进行分组并且将每个组分布到不同机器上，这样可以模拟不同地理位置的空间分布。而且，用户还可以对应每一个类型，不同类型可以使用不同的负载；

SilkPerformer 可以模拟很多种不同类型的网络，支持的网络类型甚至达到了 30 中之多，如：wireless，GPRS，Modem 等。每一种类型，用户都可以进行各自参数的设置；

SilkPerformer 采用 TrueCache 技术，可以模拟不同浏览器的缓存、cookie 管理，这样极大的模拟实际情况的负载，而且 HTTP 请求都是精确的符合实际情况；

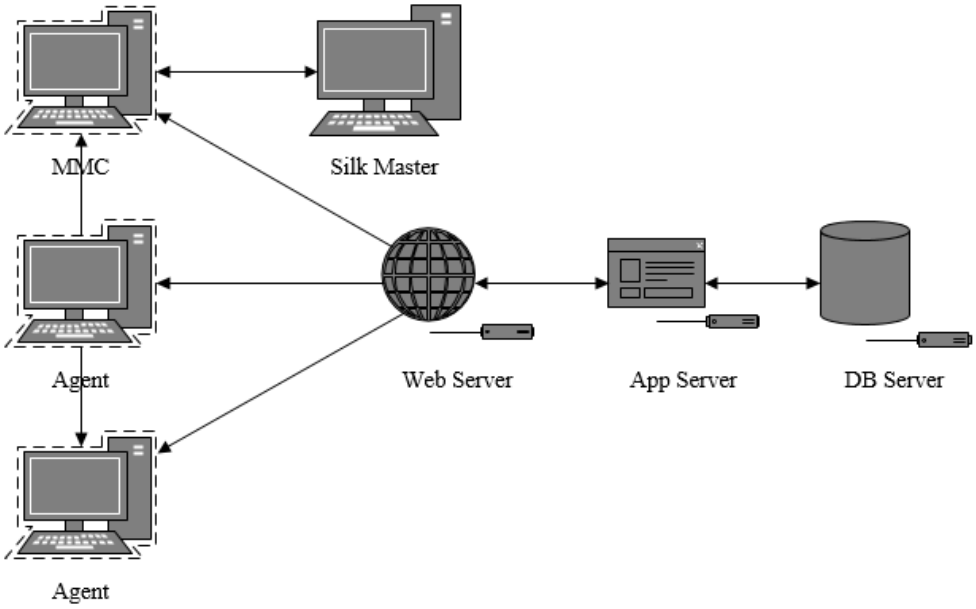


图 2.5 SilkPerformer 体系结构

SilkPerformer 能够模拟不同类型的虚拟用户负载策略，包括：持续增加虚拟

用户数、动态变化虚拟用户数、全天候虚拟用户数等，每一种都可以进行不同参数的设置；

SilkPerformer 支持的协议很广泛，主要支持的协议如下所示：

流媒体协议-Macromedia Flex/AMF、Streaming（MS,Real）等

数据库访问协议-ODBC、Oracle OCI 等

网络协议-HTTP/HTTPS、SMTP/POP、FTP、TCP/IP 等

其他商用的负载测试工具还有 HP LoadRunner、RadView WebRunner 等，通常这些商用的工具由于功能很完善，软件公司花费了巨大的人力资金进行开发，所以这些软件一般都会比较昂贵。市面上还会有一些免费或者开源的测试软件，如：OpenSTA、WAS（Web Application Stress Tool）、Apache JMeter 等。

其中，WAS 是由微软的 web 测试人员开发的，能够特别地执行真实网站的负载测试，该工具能够采用少数的几台计算机设备来模拟很多的使用者对网站进行并发请求的负载。

2.4.3 Apache JMeter

2.4.3.1 JMeter 性能测试工具的体系架构

JMeter 是一个很流行的开源性能测试工具，纯 Java 编写的，它可用于动态以及静态两种系统的性能测试。并且可以模拟系统在不同负载压力情况下系统会有什么样的反应，这种反应带来什么样的结果，以及在不断加压的时候，系统会不会承受的住，找出系统所能承受的极端压力高峰值，这样就可以为系统调优做好基础数据准备的工作。JMeter 主要是使用各种原件来创建测试所要用到的测试用例，也就是新建一个测试计划，这个计划描述了测试执行时的步骤，并且，可以在查看结果树等的图标中直观的显示出来，而执行的结果或脚本则可以保存下来（XML 格式）。在 JMeter 中一个测试计划会包括线程组、控制器、配置元件、定时器、前置处理器、后置处理器、断言、监听器等[19]。

以下将会对这几个部分一一介绍：

线程组就是控制 JMeter 执行测试计划时所使用的线程数量，可以设置每个线程的间隔实际，也可以设置测试要执行的次数，当然，这些线程之间是相互独立的，互相不干扰的执行测试计划。

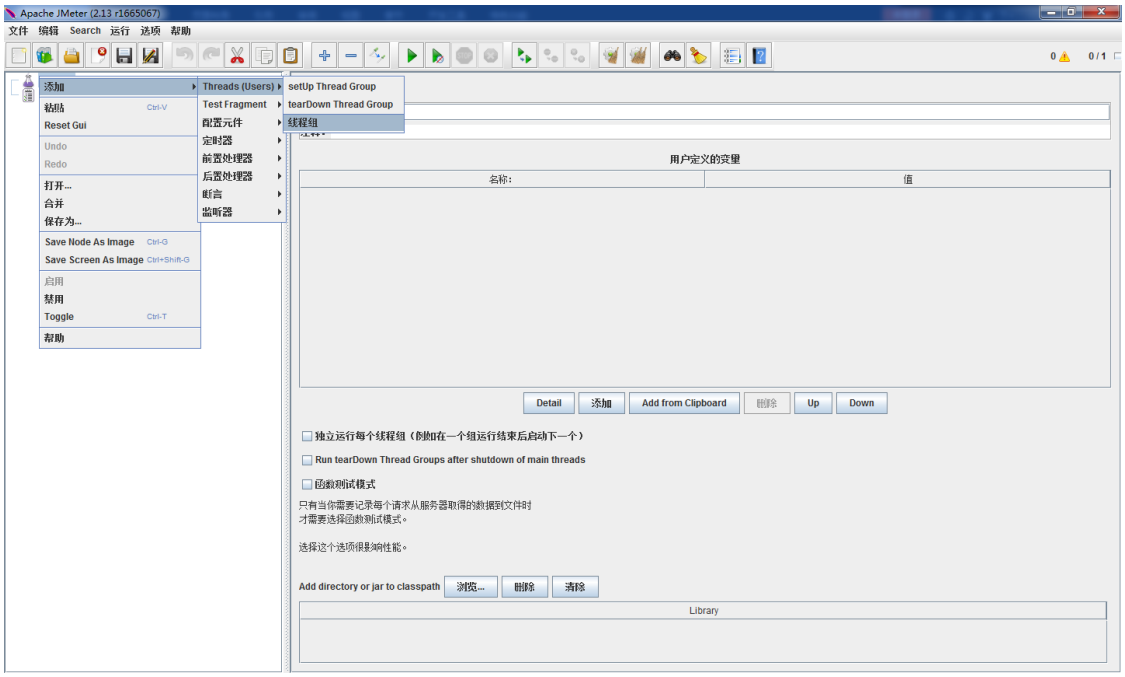


图 2.6 JMeter 操作界面

采样器（Sampler）以及逻辑控制器（Logic Controllers）被称为控制器。

采样器（Sampler）：它是 JMeter 的基础元件，用户能够使用它向服务器提出交互连接，如：HTTP 请求器等，然后采样器会等待服务器的响应，直到响应超时。

逻辑控制器：控制脚本根据用户的逻辑顺序来执行测试，例如控制 JMeter 什么时间提出交互请求。

配置元件（Config Elements）：配置元件是与测试脚本中的设置有着一定关联度，例如：设置用户变量、cookies 设置、HTTP 请求头等，它是与取样器关联在一起工作的，可以定制和修改服务器发送到请求，但是却不能添加请求。

定时器：它保存了与时间有关联的信息。定时器会在连续请求的期间添加延迟时间，如思考时间，这样服务器请求的频率就不会太过频繁，如果不这样，有可能在很短的时间内，服务器接收到了大量的请求，从而导致服务器宕机。定时器是设置在线程组中，如果在一个线程组中存在多个定时器，则延迟的时间是所有定时器中延时时间的总和。

前置处理器：所有采样器在执行之前，都会先执行前置处理器中的动作，例如：对用户参数进行配置设定、重写 URL 等。

后置处理器：所有的采样器在执行完成自己的动作之后，都要执行后置处理

器中的操作进行后置处理，例如：获取响应数据，获取其中的一些数值等。

断言：用于校验从服务器获取的响应数据是否与自己期望的数据一致。

监听器：用于获取 JMeter 在测试过程中的一些测试结果，并能够按照不同的形式予以展示。监听器有不同的种类，包括图像结果监听器、结果树监听器等。其中图像结果监听器可以以图表的形式展现响应时间等信息。结果树监听器则用 HTML 或者 XML 形式来展现取样器的请求和响应。

而 JMeter 源码的基本框架如图 2.7 所示，打开 bin 文件里的 jmeter.bat 就可以运行 JMeter 了。

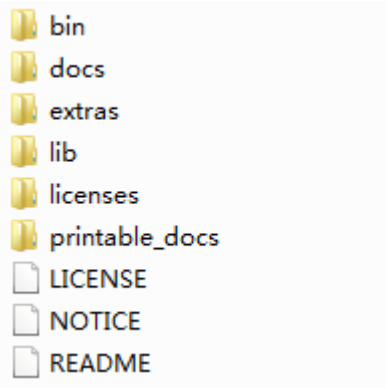


图 2.7 JMeter 软件源码

其中 bin 文件夹包含启动 JMeter 所必需的.bat、.cmd 以及.sh 等文件，还包含了 ApacheJMeter.jar 文件和一些属性文件；lib 目录包含了 JMeter 所需的所有 jar 文件，其中的 ext 文件夹包含了 JMeter 的核心 jar 包，junit 文件夹存放 JUnit 测试脚本，用户可以把扩展功能所使用的 jar 包直接放置在 lib 目录下；docs 文件夹下包含了 JMeter 的 Java Docs，printable_docs 的 usermanual 文件夹下包含了 JMeter 用户使用手册的.html 和 pdf 文件；estras 目录中包含了对构建工具 Ant 支持的文件，用户可以使用 Ant 来实现测试自动化，如：批量脚本、报表的生成等。

2.4.3.2 JMeter 工作原理及执行流程

JMeter 的工作原理其实并不复杂，简单的来说，它可以被用来当作 Web 服务器与浏览器这两者中间的代理网关，如此就可以很容易的并且准确的获取浏览器提出的请求和 Web 服务器给出的响应，如此 JMeter 便很容易地生成操作者想得到的性能测试脚本。之后，JMeter 就可以通过线程组来模拟真实用户对 Web 服务器的访问压力，如图 2.8 所示。

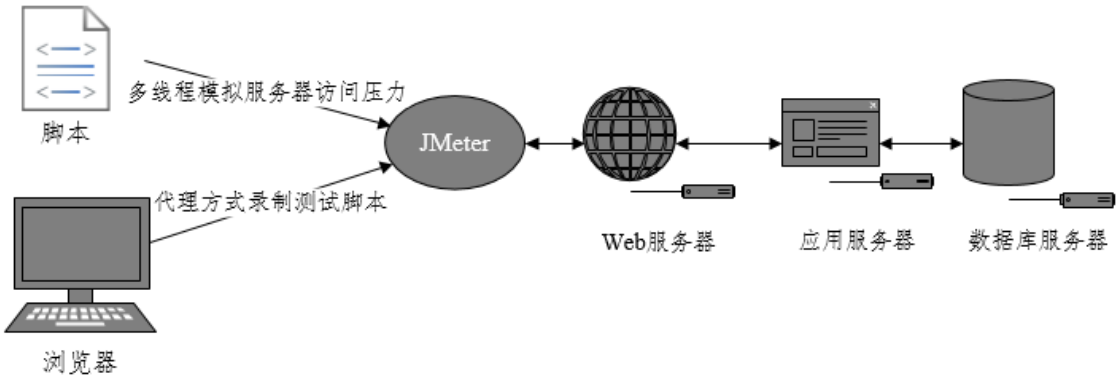


图 2.8 JMeter 工作原理

JMeter 可以按照如下几种模式进行启动，GUI 图像界面模式、服务器模式、帮助模式、版本模式、独立模式、命令模式，一般用户最常用的是图形界面模式，它能够提供给用户一个容易使用操作、具有良好交互性的体验。在执行分布式系统测试时，一般会采用服务器模式。一份齐全的测试计划一般都会包含如上所述的单一甚至许多线程组、采样器、逻辑控制器、定时器、监听器、断言、配置元件。

当点击\bin\jmeter.bat 后，就会打开如图 2.6 所示的 JMeter 界面，下文对 JMeter 的基本操作进行介绍[20]：

①添加/删除测试元件：为测试计划添加测试元件，只需要在“测试计划”图标上，右键，即可在打开的目录树上选择需要的元件，如图 2.9 所示

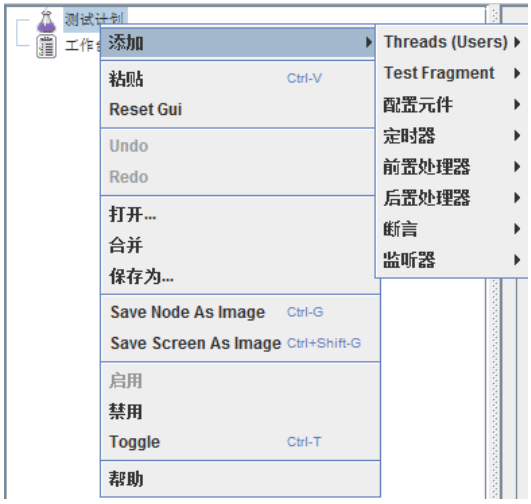


图 2.9 JMeter 添加元件

在添加好的元件上面点击右键，在弹出的菜单栏，选取“删除”既可以移除

该元件；

②加载/保存测试元件：右键已经添加好的测试元件，选择“合并”按钮，然后从对话框中中选择外部文件，这样就可以将测试元件添加进来；假如选取“另存为”选择项，那么可以以文件的形式保存组件，以后的操作就可以仍然使用该元件；

③配置测试元件：点击已添加的测试元件，在界面右侧用户可以设定元件的参数；

④保存测试计划：选择“文件”栏下的“保存测试计划”或者按下快捷键 ctrl+s；

⑤运行测试计划：选择“运行”菜单下的“启动”按钮，可以运行测试计划，界面右上角的正方形内的数值代表已经激活的线程数/总线程数；

⑥终止测试：能够执行停止或者关闭两个指令来终止测试；

⑦错误记录：JMeter 可以讲警告或者错误记录在 jmeter.log 文件中。

JMeter 的测试脚本正常启动后，就可以按照.jmx 脚本中以 XML 格式描述流程进行性能测试；测试结束之后，能够在监控器中查看测试的结果。JMeter 脚本的执行流程如图 2.10 所示[21]：

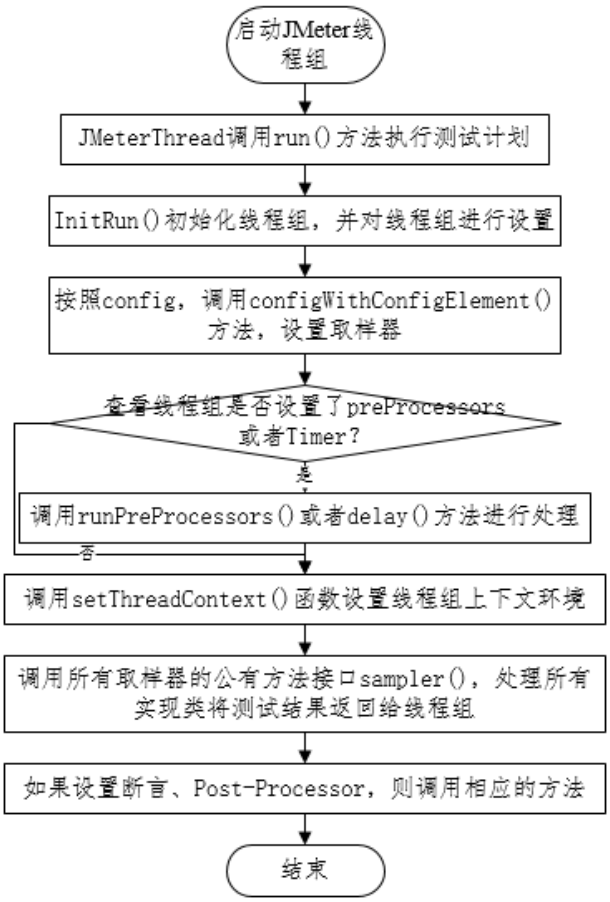


图 2.10 JMeter 脚本运行流程

2.5 本章小结

本章简要描述了系统性能测试方面的内容，主要包括软件性能测试的概念、软件性能测试的目的、软件性能测试的过程、软件测试国内外现状分析、软件性能测试方法以及软件性能测试相关工具的介绍，这样就可以对软件性能测试有个基础的了解。

第3章 基于 web 的医疗保险信息管理系统的性能测试的需求分析

3.1 基于 web 的医疗保险信息管理系统概述

3.1.1 web 系统概述

web 软件系统是一种通过 web 进行交互访问的应用，其最突出的优点是用户只需要通过浏览器既可以进行访问系统，而不需要再安装另外的软件，其采用的是浏览器/服务器（B/S）模式[22]。web 应用的系统结构如图 3.1 所示，组成部分通常会有客户端、web 服务器、数据库服务器三个部分，其中客户端用于展示服务器返回的数据，web 服务器用于处理客户端发送的请求，数据库服务器用于存储以及管理整个系统中的数据。

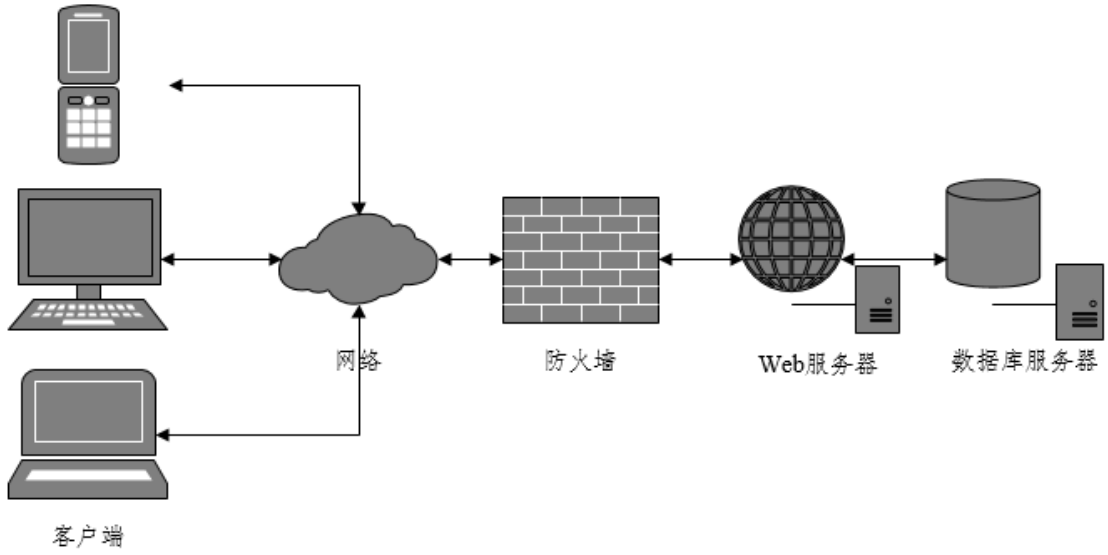


图 3.1 web 系统结构

由于每一层次之间的依赖关系比较紧密且复杂，每一层以及之间极有可能会发生故障，这样使得 web 系统非常需要进行软件测试尤其是性能测试。web 应用系统和传统软件系统比较，它具备了分布式、并发性、异构性、多用户等特点，这几个特征给 web 软件测试带来了一些不同寻常需求和挑战。这些特点主要表现

在以下场景中，用户的多样性，不同的用户所使用的硬件环境、网络状况、操作系统、浏览器都不太一样，这样需要对 web 应用的适用性进行测试；网页不断的多媒体化，使得应用系统的测试对象具有多样性；应用系统具有大量的用户，同一时间对 web 系统并发请求的数量会非常巨大，这对系统的实时性、响应时间等性能提出了很大的考验，这就有必要对它们执行性能测试。

web 系统的测试相较于传统软件测试，它更加的复杂很多，传统软件测试技巧策略已经不再能够胜任 web 系统的测试需要，这对软件性能测试带来了独特的需求，所以需要一套独特的测试方法。

3.1.2 基于 web 的医疗保险信息管理系统概述

随着社会的不断进步，人们的生活水平持续的改善，则开始愈加密切地关心自身的健康，随之而来的是人们对有关医疗的服务设施的需求也更加高涨，医疗条件的不断健全，医院它们的业务也有了快速的增加，它们的操作流程变得更加的多样繁杂，使得传统纯手工的管理方式已经不再能够满足医院和药店的需求，这开始变成它们业务发展的制约因素。在简单的纯人工管理的方式中，因为病人的信息繁多复杂，致使医院查找信息变的异常的有障碍，居民的医疗保险信息和对账系统经常会发生各种异常问题而且这些错误不能跟踪追溯，手工的记账方式更会轻易出现遗漏和错误而且手工方式的收款也是无法规避过失，所以怎么样优化原有的服务质量，提升业务水平，这已经变成了急需改善处理的难题。由以上分析可知医疗保险信息管理系统完全能够使得这些迫切的难题得到解决。

医疗保险信息系统（MIIS: Medical insurance information management system）是为了使病人获得基本的医疗保险服务，又能给医院和药店提供快速便捷的信息交互功能。该系统所牵涉的机构十分地广泛，例如：保险公司、医院、药品零售店、银行等金融机构、财税单位等，只有各种机构单位的通力合作，才使资源得到充分合理的利用，通过与定点医疗机构、定点零售药店建立网络连接，来实现投保人在就诊和购药时，使用医保卡就能实时支付的功能。最终创建网络互联的信息管理系统，使得医疗保险业务的开展更加的信息化。本系统是医疗保险信息管理系统中的一个子项目系统。在本系统中要实现的系统模块是投保人医疗保险信息系统，主要包括投保人的信息、投保药品、药店、和投保人理赔参数的设置等，达到一个提取信息的功能。另外，因为基于 Web 的应用系统，没有必要在用户的计算机上安装其他软件，用户只需要通过浏览器就能实现与之交互，这很大

程度上减轻了应用系统升级部署的代价以及给用户使用所造成的影响，所以 Web 系统被广泛的使用。目前，很多企业的核心业务系统均是 Web 应用，用户可以享用更丰富的功能和体验，所以，在这里研究软件测试的结合项目是基于 web 的投保人医疗保险信息系统。

3.2 系统性能测试的瓶颈与需求

本文以实习阶段参与的测试项目为基础，即上文介绍的医疗保险信息系统，该系统拥有许多复杂的结构模块，由于篇幅原因，介绍整个系统各个模块的性能测试显得不切实际，也没有这个必要，所以下文选取了该系统中的定价（Setup-Pricing）模块子系统作为研究的对象。该模块有许多的操作步骤流程，如：搜索、新建、删除等，性能测试中需要涉及到这些全部的流程，以全面测试该子系统的性能表现。在性能的各个指标中，选取了响应时间、吞吐量以及系统硬件指标参数作为研究方向，通过 JMeter 测试工具和其他系统命令查看各参数的变化情况，确定系统的瓶颈缺陷，能够可以为了系统调优做参考。在下文中会详细讨论各个测试，如：并发瓶颈测试、内存泄露测试等等，以及各个测试的需求和测试用例，在此不再赘述。

3.3 本章小结

本章节简要概述了基于 web 的医疗保险信息系统以及分析了系统性能测试的瓶颈，并对此系统提出相应的性能测试需求。

第4章 性能测试的实施

4.1 性能测试工具的选择

在进行软件性能测试之前，选择一款适合的性能测试工具尤为重要，工具的选择直接影响了软件测试的质量以至于软件的成功上线。由上文描述分析可知，现如今在软件测试行业，用于性能测试的软件工具有很多，市面上的测试软件不下一百种之多。总的来说，商业化的软件功能完备，稳定性、性能方面都比较好，适用性也比较广，但是需要花费很多的学习成本，而且一般这些软件都比较昂贵。随着开源代码/项目的兴起，也有许多开源的性能测试工具存在，其中 Apache JMeter 就是一看比较优秀的工具。

基于 web 的医疗保险信息管理系统的性能测试与分析的之所以选择 JMeter 这个性能测试工具，本文主要考虑以下几个方面：

（1）JMeter 工具经济划算

对于本系统的性能测试选择的工具要在经济方面评价其是否合理，成本-效益分析要估计出系统性能测试过程中的花费并与收益衡量比较。由于 JMeter 是开源软件，测试工具成本可忽略不计，并且它能够避免手工测试所带来的不仅消耗时间而且枯燥单调的缺点，而且需要投入大量的人力资源的问题，使用 JMeter 能达到预期的效益，所以本系统进行性能测试选择 JMeter 这个工具，在经济上是可行的。

（2）JMeter 工具可用性和易用性

在这里要考虑 JMeter 这个性能测试工具是不是支持所要测试的系统。一般要考虑该系统是基于什么协议，用的是什么编程技术，是基于什么样的平台等等。本系统是基于 HTTP 协议，采用 JAVA 编程技术，基于 windows 平台，而且是 B/S 系统。这样看来，这个 JMeter 工具在本系统中是可以使用的。并且，JMeter 性能测试工具操作便捷，对于测试人员来说，在较短的时间内就可以进行熟练的操作。JMeter 性能测试工具除了以上优点外，还具有非常友好的显示界面，并且能够对服务器端的相关应用程序进行脚本的定制录制，然后进行测试。

基于以上分析结果，本系统性能测试工具选择 JMeter，能够很好的满足本系

统的性能测试要求。

4.2 web 系统性能测试平台

Apache JMeter 是 Apache 基金会的开发人员使用 Java 语言所研发的开源性能测试软件，它能够虚拟出数量很大的压力负载对服务器以及网络进行强度测试分析它们的性能。JMeter 不仅能够测试静态资源还可以测试动态资源，而且还可以通过创建带有断言的脚本进行回归测试。它的工具包包含四个部分，分别为：负载发生器、用户运行器、资源生成器、报表生成器。JMeter 工具在性能测试方面有着许多优点：添加测试用例，不用依赖于界面，只要服务启动起来，就能设置传递参数；通过 Badboy 录制生成测试脚本，不需要编程，而且脚本的维护也很方便；完全不需要和前台页面请求交互，仅仅只要求与后台进行交流沟通；断言功能，可以验证代码结果是否与期望值相符合；使用参数化以及内置函数来添加修改测试数据。

关于 JMeter 测试工具的具体体系架构和工作原理，在上文中已经介绍过，此处不再赘述。

4.3 性能测试的实施

4.3.1 测试方案制定

4.3.1.1 测试目标

系统性能测试的主要目标为：

①验证系统性能情况能否满足验收标准；

②识别并确保相关性能的缺陷在系统部署之前能够解决。

4.3.1.2 性能验收标准

性能的指标总是有两个标准。第一个是性能标准（包括要求和目标）；第二个是参与完成标准。在下面的章节中，本文采用两种类型的标准来解释定价子系统（Setup-Pricing）性能测试工作。当所有的性能标准或任何一个参与完成的标准得到满足，那么就可以说已经达到了性能的要求。

性能标准可以具体被分为被测系统的性能要求和目标。优先的性能测试工作的结果是验证系统满足目前所有这些目标和需求。下文具体说明这些指标：

①性能测试要求

性能要求既是只有满足这些条件，应用系统才能上线，成为一个生产系统。目前，系统的性能测试没有设定特殊的性能需求。所以本文也就不再进行描述。

②性能测试目标

目标既是期望应用程序能够达到的标准，它不同于前文所说的要求。测试将会一直继续下去，直到满足了所有的目标或者时间/金钱已经耗尽时，也已经达到要求。

一般的目标是以下三个方面：

①评估定价（Setup-Pricing）子系统的响应时间；

②评估定价子系统的吞吐量；

③评估定价子系统的性能。

针对定价子系统，并没有特别目标的测试需求，所以性能测试工作只是用来评估系统的性能。

4.3.1.3 参与完成标准

因为一些性能测试团队无法控制的情况，性能测试的需求以及目标都不能达到时，性能测试将会在以下情况下被认为是有效的：

①所有阻止应用程序实现性能指标的瓶颈都确定在团队的控制之外；

②达到了预先设定的结束日期；

③尽管一些性能要求或者目标没有实现，但是团队一致认为应用程序执行效率是可以接受的。

4.3.2 测试环境设计

本节描述了定价子系统的生产环境和测试环境，采用不同的基础设备，通过搭建与外部网络隔离的测试环境，来衡量所开发系统的性能。

4.3.2.1 开发环境

定价（Setup-Pricing）子系统包含两个服务器实例：数据库服务器以及 web 服务器。目前，生产环境还没有确定，开发环境可能作为未来构建生产环境的一个重要参考。所以测试一个与开发环境及其相似的测试环境能够部分预测开发环境和潜在的生产环境的性能。

开发环境的详细信息如下所示：

a）20*CPUs（PowerPC_Power7），CPU 时钟频率：3300MHz，内存：16GB；

b）操作系统：AIX 7.1.1.16 TL00；

c）WebSphere Application Server - ND, 7.0；

d）DB2 版本：10.1。

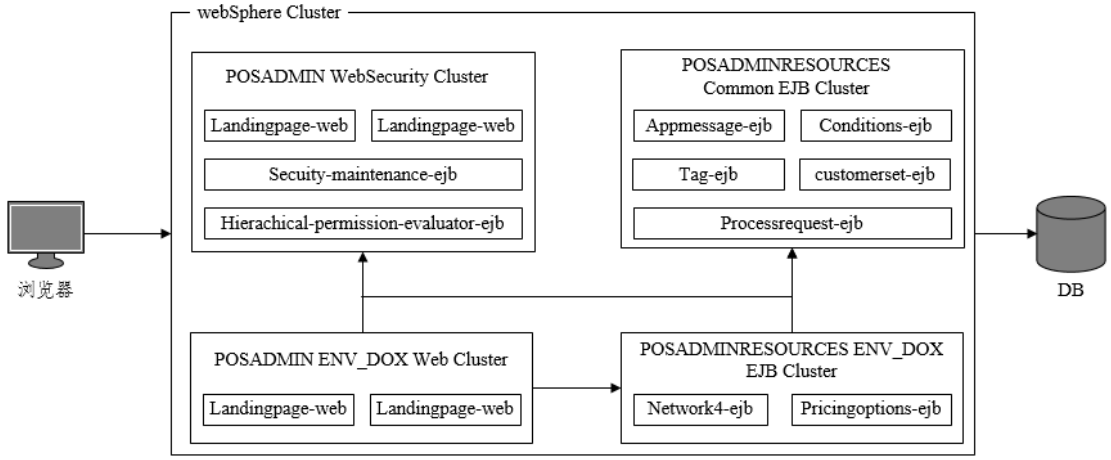


图 4.1 定价子系统开发环境的逻辑视图

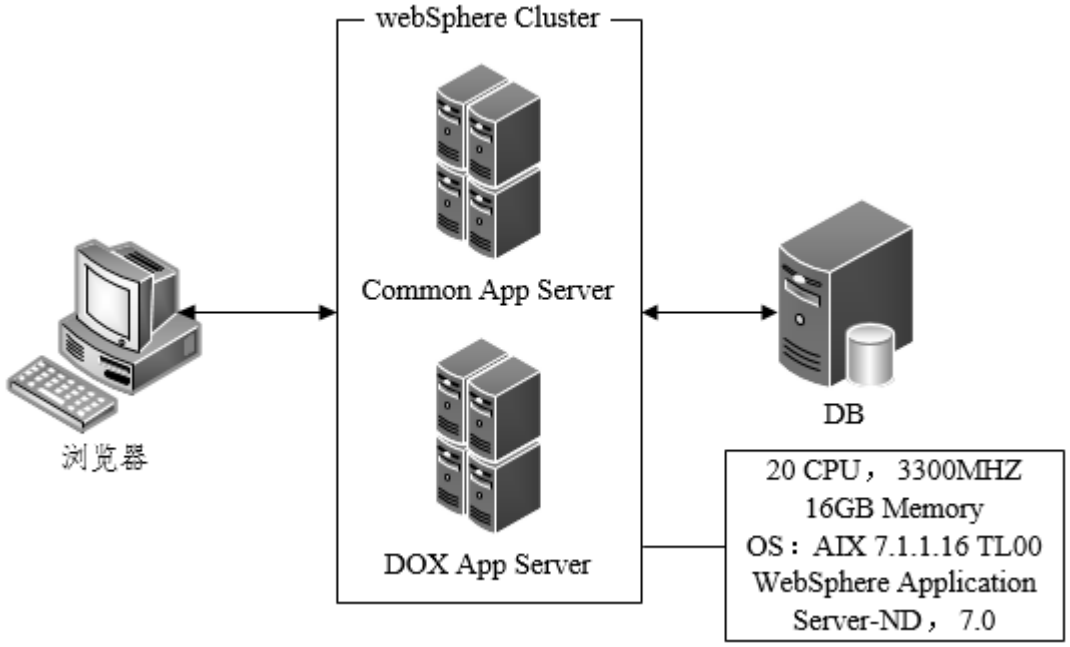


图 4.2 定价子系统开发环境的物理视图

4.3.2.2 测试环境

本节的重点是定价子系统测试环境的搭建。图 4.3 和图 4.4 分别显示了测试环境的逻辑视图和物理视图，由此可见测试环境与开发环境存在一定的差异。所以，从这种开发环境中搜集的性能测试数据与真正的开发环境和生产环境所产生的测试数据没有绝对的联系，然而这些数据可以预测定价应用的性能。

测试环境与开发环境主要的不同点如下所示：

①客户端，测试环境中的 web 服务器和数据库服务器都是部署在公司内部网络中，而不是项目组内部网络的生产环境中；

②在测试环境中，所有的 web 应用程序和 EJB 都是部署在单个机器上的而不是生产环境中的集群上。

测试环境的具体信息如下所示：

a）Intel(R) Xeon(R) CPU E5645 2.40GHz，内存：16GB；

b）Red Hat 4.1.2-48（ Linux 2.6.18-194.el5）；

c）WebSphere Application Server - ND, 7.0；

d）DB2 版本：10.1。

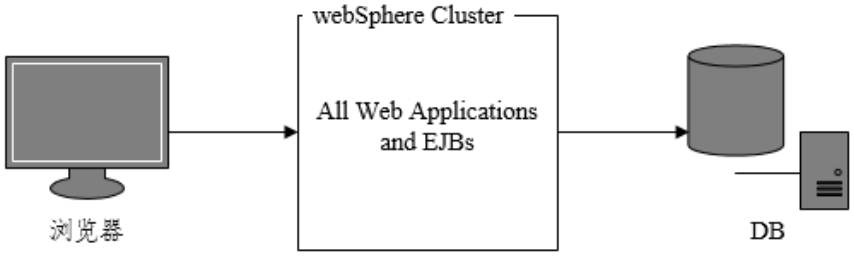


图 4.3 定价子系统测试环境逻辑视图

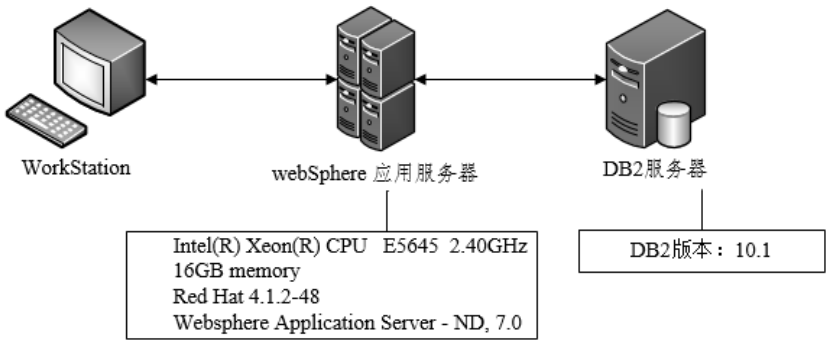


图 4.4 定价子系统测试环境的物理视图

4.3.3 测试场景设计

4.3.3.1 定价子系统数据源

本节详细描述了被测系统的数据管理，性能测试所需的数据可以分为两种类型：

①测试中需要参数化的动态数据；

②目标系统中的数据，通常包括数据库或者文件服务器中的数据。

其中，动态数据通常被描述在特定目标的工作负载特征部分。

经验表明，系统数据库中的数据量通常会很大程度地影响系统的性能。所以，所以本文需要基于规划、营销、估计，来模拟这个定价子系统潜在的数据环境。为了模拟定价子系统的数据量，需要按照定价子系统数据库设计，来生成插入数据。下文将会介绍，生成不同数据量来检查对系统性能的不同影响。详细的数据量将会在测试期间有着启发式的设定。

主要的表结构如表 4.1 所示：

表 4.1 COMPONENT Table

表名			T08		描述	Pricing Option Component			
	主键	外键	逻辑表名	物理表名	描述	大小	NULL	外键	描述
1	O		PRICING_ COMPONENT_ID	PRICING_ COMPONENT_ID	DECIMAL	7	X		
2			LAST_MNT_TS	LAST_MNT_TS	TIMESTAMP	26	X		
3			LAST_MNT_OPID	LAST_MNT_OPID	CHAR	8	X		
4			CUSTOMER_SET_ID	CUSTOMER_SET_I D	DECIMAL	5	X		
5			EVENT_ID	EVENT_ID	DECIMAL	13	X		TRACK USER OPERATI ON
6			PRICING_ COMPONENT_SHT_NM	PRICING_ COMPONENT_SHT _NM	CHAR	10	X		
7			COVERAGE_ CMPONT_STATUS_CD	COVERAGE_ CMPONT_STATUS _CD	CHAR	1	X		
8			COMPONENT_TYPE_ID	COMPONENT_TYP E_ID	DECIMAL	3	X		

表 4.2 EFFECTIVE DATE RANGE Table

表名			S95		描述	Date Range			
	主键	外键	逻辑表名	物理表名	描述	大小	NULL	外键	描述
1	O		PRICING_ COMPONENT_ ID	PRICING_ COMPONENT_ ID	DECIMAL	7	X	T08. PRICING_ COMPONENT_ID	
2	O		PRICING_ COMPONENT_ DTL_ID	PRICING_ COMPONENT_ DTL_ID	DECIMAL	7	X		
3			PRICING_ COMPONENT_ EFFV_DT	PRICING_ COMPONENT_ EFFV_DT	DATE	10	X		
4			PRICING_ COMPONENT_ TERMTN_DT	PRICING_ COMPONENT_ TERMTN_DT	DATE	10	X		

表 4.3 SECTION COST DETERMINATION Table

表名			partition		描述	Pricing Option Component			
	主键	外键	逻辑表名	物理表名	描述	大小	NULL	外键	描述
1	O		PRICING_ COMPONENT_ DTL_ID	PRICING_ COMPONENT_ DTL_ID	DECIMAL	7	X	S95. PRICING_ COMPONENT_ DTL_ID	
2	O		PRICING_ COMPONENT_ID	PRICING_ COMPONENT_ID	DECIMAL	7	X	T08. PRICING_ COMPONENT_ID	
3	O		COST_ DETERMINATION_ ID	COST_ DETERMINATION_ ID	DECIMAL	7	X		
4		O	CONDITION_ID	CONDITION_ID	DECIMAL	9		O42.CONDITION_ID	
5		O	CONDITION_ SEQUENCE_ID	CONDITION_ SEQUENCE_ID	DECIMAL	3		O42. CONDITION_ SEQUENCE_ID	
			CONDITION_MET_CD	CONDITION_MET_ CD	CHAR	1			

表 4.4 SECTION PAYMENT Table

表名			S99		描述	PAYMENT STATISFACTION			
	主键	外键	逻辑表名	物理表名	描述	大小	NULL	FK	描述
1	O		PRICING_ COMPONENT _DTL_ID	PRICING_ COMPONENT_DTL _ID	DECIMAL	7	X	S95. PRICING_COM PONENT_DTL_I D	
2	O		PRICING_CO MPONENT_I D	PRICING_COMPON ENT_ID	DECIMAL	7	X	T08. PRICING_COM PONENT_ID	
3	O		SATISFACTIO N_OPTION_I D	SATISFACTION_OP TION_ID	DECIMAL	7	X		
4		O	CONDITION_ ID	CONDITION_ID	DECIMAL	9		O42.CONDITIO N_ID	
5		O	CONDITION_ SEQUENCE_I D	CONDITION_SEQU ENCE_ID	DECIMAL	3		O42. CONDITION_SE QUENCE_ID	
6			CONDITION_ MET_CD	CONDITION_MET_ CD	CHAR	1			
7			PRIORITY_ID	PRIORITY_ID	DECIMAL	3			

表 4.5 STATISFACTION OPTION Table

表名			S98		描述	STATISFACTION OPTION			
	主键	外键	逻辑表名	物理表名	描述	大小	NULL	外键	描述
1	O		PRICING_COM PONENT_DTL_ ID	PRICING_COMPONE NT_DTL_ID	DECI MAL	7	X	S95. PRICING_COM PONENT_DTL_I D	

2	O		PRICING_COMPONENT_ID	PRICING_COMPONENT_ID	DECIMAL	7	X	T08. PRICING_COMPONENT_ID	
3	O		SATISFACTION_OPTION_ID	SATISFACTION_OPTION_ID	DECIMAL	7	X	S99. SATISFACTION_OPTION_ID	
4			SATISFACTION_OPTION_TYPE_CD	SATISFACTION_OPTION_TYPE_CD	CHAR	2	X		
5			SATISFACTION_OPTION_NM	SATISFACTION_OPTION_NM	CHAR	1	X		
6			ORDER_OF_PRECEDENCE_CT	ORDER_OF_PRECEDENCE_CT	DECIMAL	3			

以上五张表都是与定价相关的表，用户网页登陆的性能将会与这些表有很大的关联。这五张表是定价子系统的主要的表，相比于其他的表，它们会有很高的读写频率。五张表的 E-R 图如图 4.5 所示：

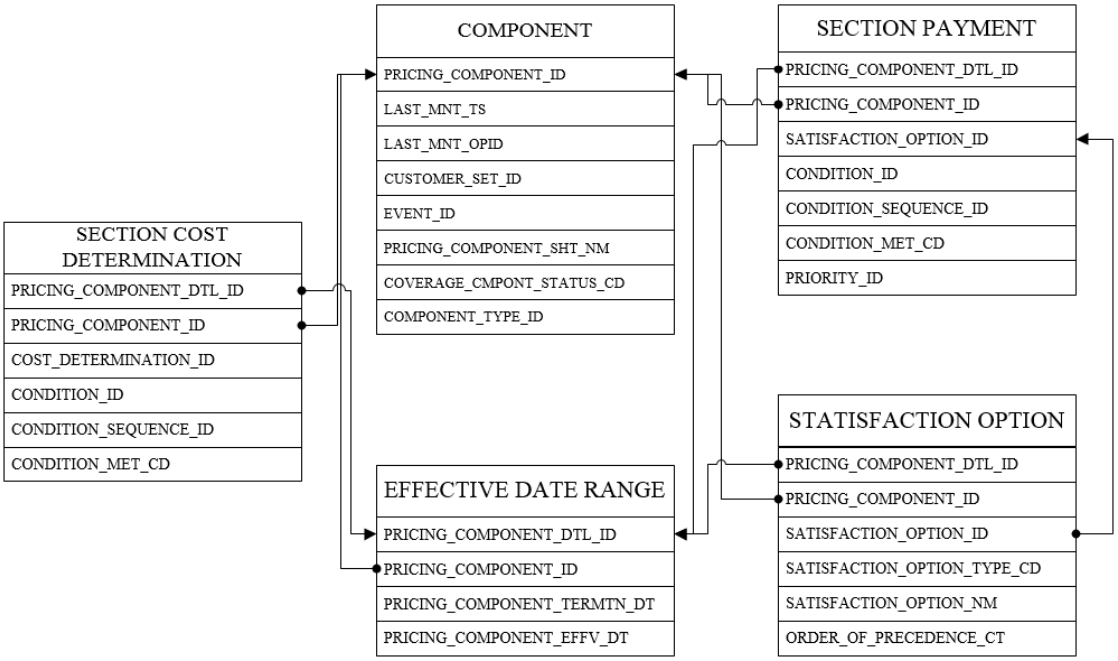


图 4.5 定价子系统表之间 E-R 图

4.3.3.2 响应时间评估

响应时间是一个 web 系统的一个关键指标。定价子系统是一个基于 B/S 的医疗价格管理系统的一个应用程序或者说是一部分。

该子系统性能测试中响应时间的目标如下：

①在不同负载下评估平均响应时间取决于场景的重要性。有三个级别的访问频率水平：高、中、低。

a）高水平：50 个峰值用户连接的访问概率为 70%

b）中等水平：20 个峰值用户连接的访问概率为 20%

c）低水平：10 个峰值用户连接的访问概率为 10%

②用于搜集平均响应时间的页面的最小测试次数必须大于 20。

（1）理论

请求的响应时间是用户向服务器发送一个 HTTP 请求和服务器返回的 HTTP 响应之间的时间间隔。这个过程包括域名解析，建立 HTTP 连接和数据传输。如图 4.6 简要描述了一个 HTTP 请求的过程。计算响应时间可约表示如下：

请求的响应时间=T1+T2+T3+T4

图 4.6 请求的响应时间

web 页面的响应时间是发送第一个 HTTP 请求的开始和收到最后一个 HTTP 请求的结束之间的时间间隔。

以下指标可以用来全面显示响应时间：

算术平均：一组数值的算术平均数；

百分比：一个变量的值，指的是观测值的一定百分比，这个指数将被这样使用，如：“95%的响应时间小于 100ms”；

中位数：数量位于中间的一组值；

正常的值：一组值中，出现最频繁的值标；

准偏差：显示有变化程度或与平均水平的“分散”程度，它用来测量给定数据集的稳定性。

理论上，可以从不同的角度得到的响应时间序列和上面列出的计算统计信息，分析测试结果。额外的指标应该从服务器收集，以帮助确定负载级别，系统瓶颈，系统状态和运行的测试：CPU 的占用、RAM 的利用、SWAP 利用、IO 速度、网络速度。这些统计数据，例如平均、百分数、中位数、正常值和标准偏差，可以用来显示系统动态的不同方面。

（2）工作负载特征

以下部分将描述工作负载模型用于评估定价系统的性能。工作负载分布的估

计由团队完成。因为缺乏系统日志，所以不能得到真正的分布信息。

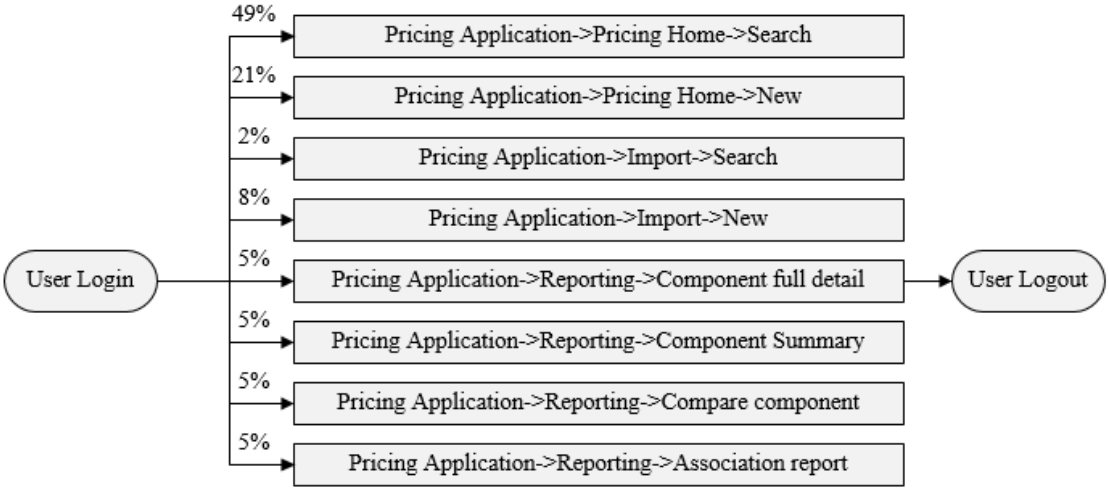


图 4.7 定价系统工作负载模型概况

系统中有八个主要的业务场景：

①定价应用程序→“定价”主页→搜索

②定价应用程序→“定价”主页→新建

③定价应用程序→输入→搜索

④定价应用程序→输入→新建

⑤定价应用程序→报告→组件的全部细节

⑥定价应用程序→报告→组件的概况

⑦定价应用程序→报告→比较组件

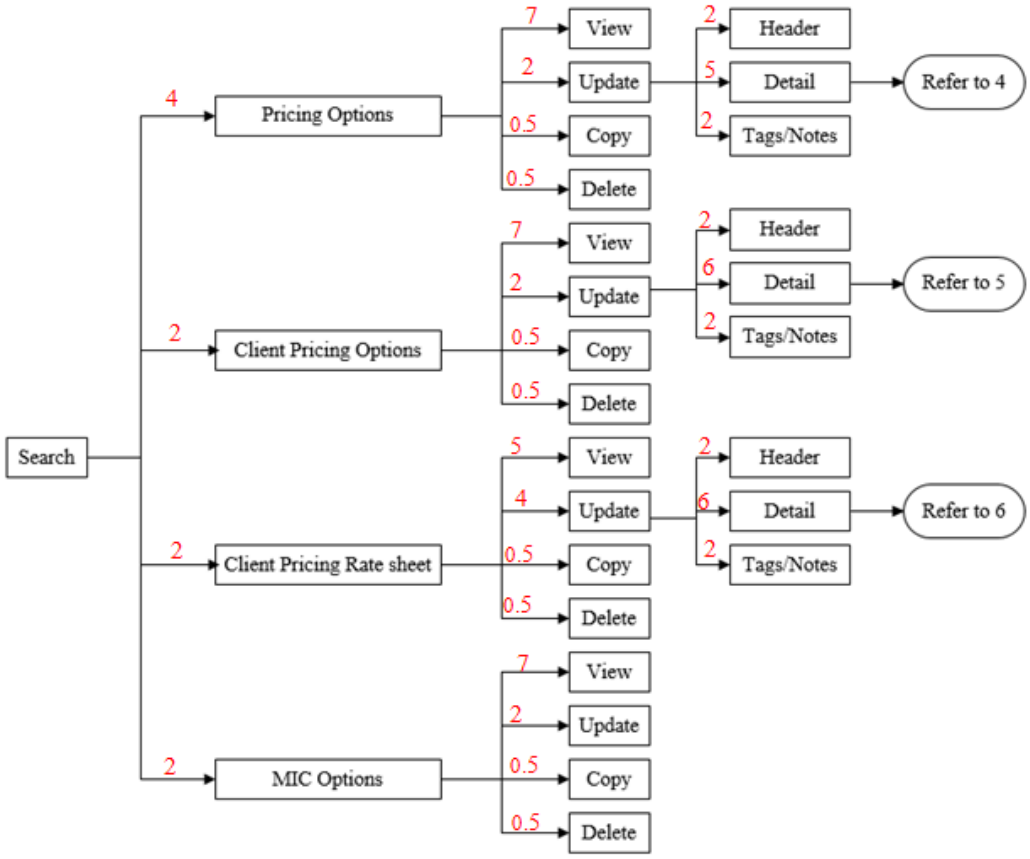
⑧定价应用程序→报告→关联报告

（3）解决方案

本节将描述被推荐的定价系统性能测试方法。根据工作负载分布和工作流，本文将所有可能的路径转换成连续的过程，每个过程将与某一个概率有关。一个过程中的并行虚拟用户数量将为：VUs*P，其中 VU 表示虚拟用户的总数模拟过程中，P 代表一个特定过程的可能性。在不同的 VU 下，随着并发用户的增加，测试该系统来提取出响应时间的趋势（请求和 web 页面）。

（4）测试用例

在本节中将详细描述定价子系统的所有测试用例。由于文章长短的限制，本文将不描述参数化的表。如图 4.8 至图 4.12 包含所有在性能测试中涉及的将被执行的顺序工作流。


Search
4
Pricing Options
2
Client Pricing Options
2
Client Pricing Rate sheet
2
MIC Options
7
View
2
Update
0.5
Copy
0.5
Delete
2
Header
5
Detail
2
Tags/Notes
Refer to 4
Refer to 5
Refer to 6
6
5
4


Refer to 6
Detail Tab
Client Pricing Rates Section
8
Update Default/Override
1
Insert Override
1
Delete Override

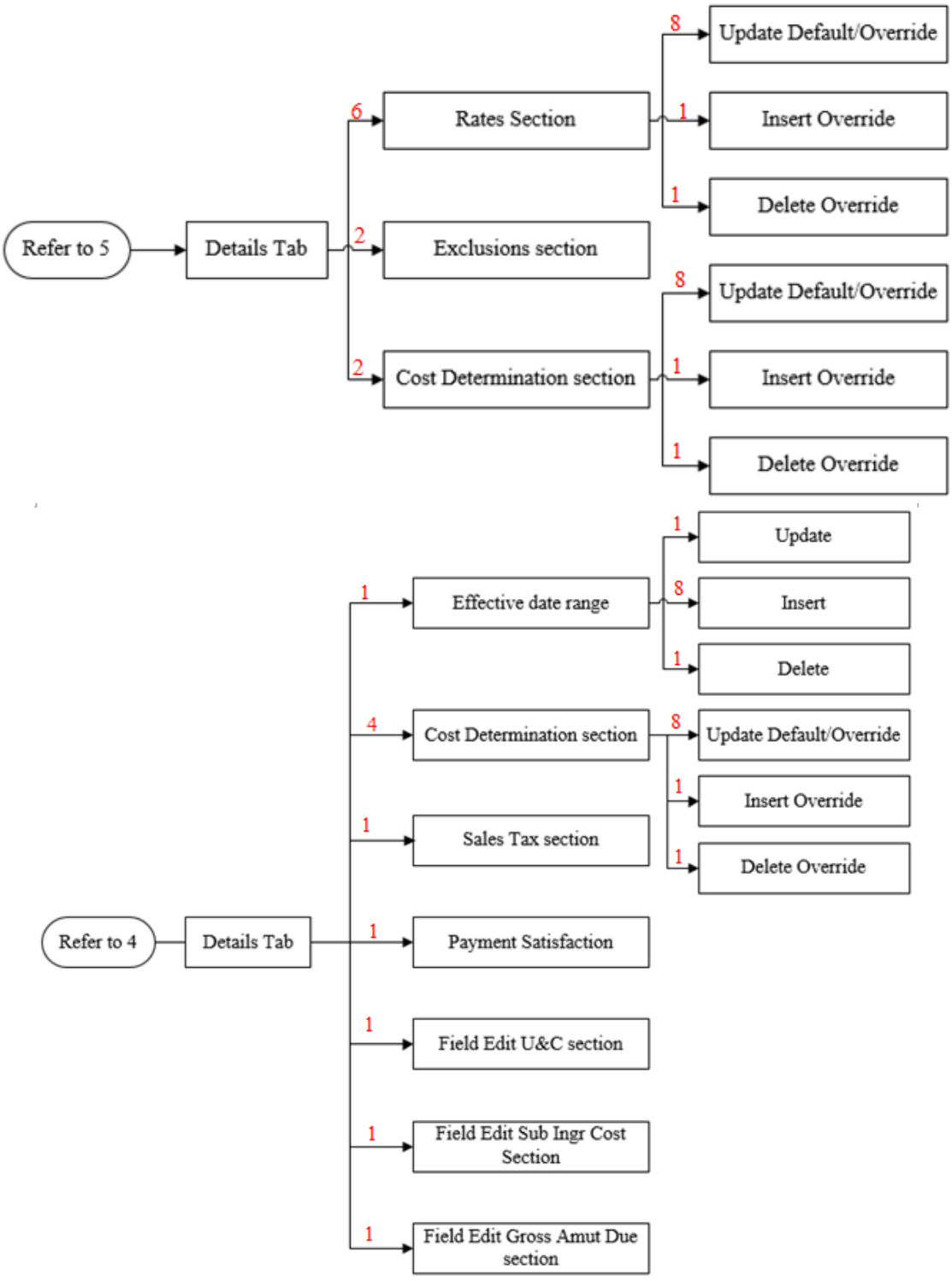


图 4.8 定价主页搜索程序的测试用例

子工作流：定价应用系统→定价主页→搜索

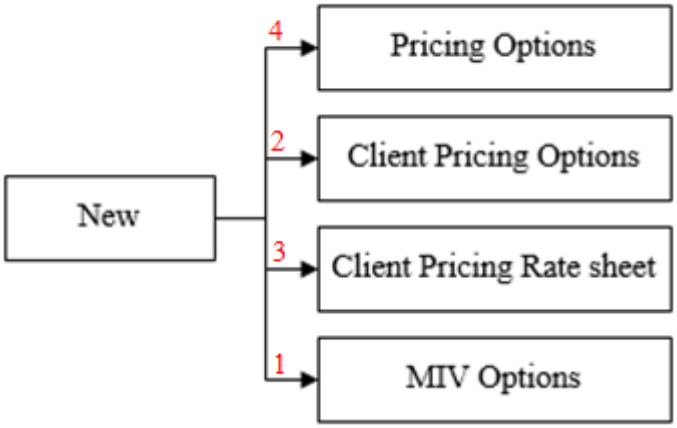


图 4.9 定价系统主页新建测试用例

子工作流程：定价应用→定价主页→新建

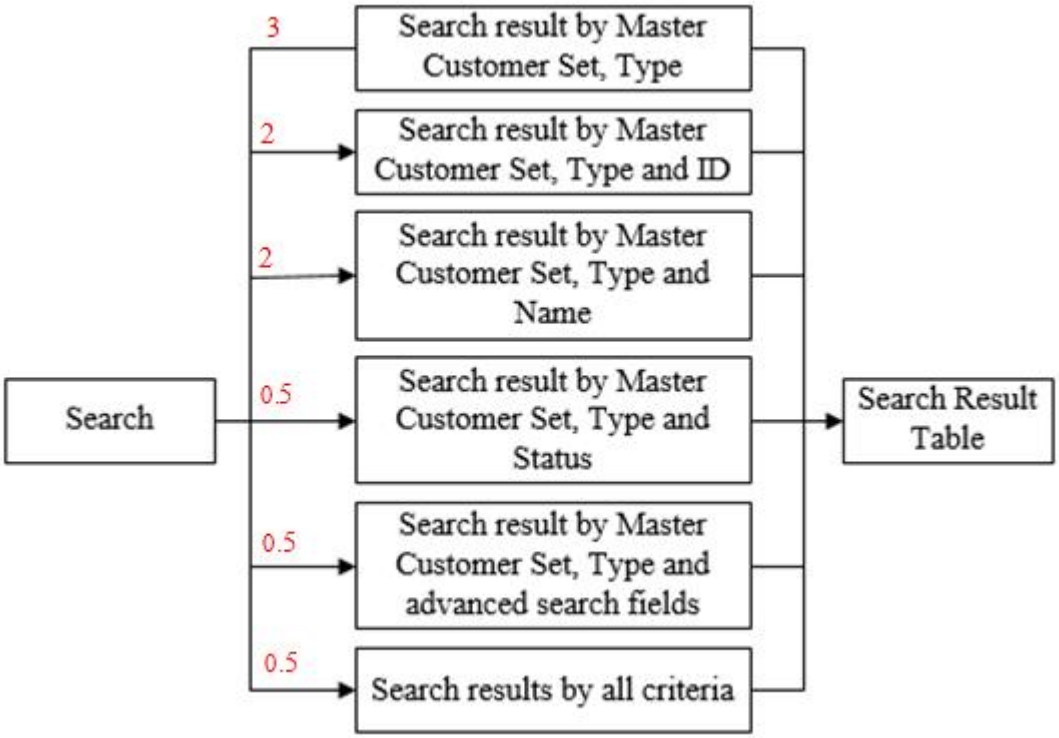


图 4.10 含有搜索的输入程序的测试用例

子工作流：定价应用系统→输入→搜索

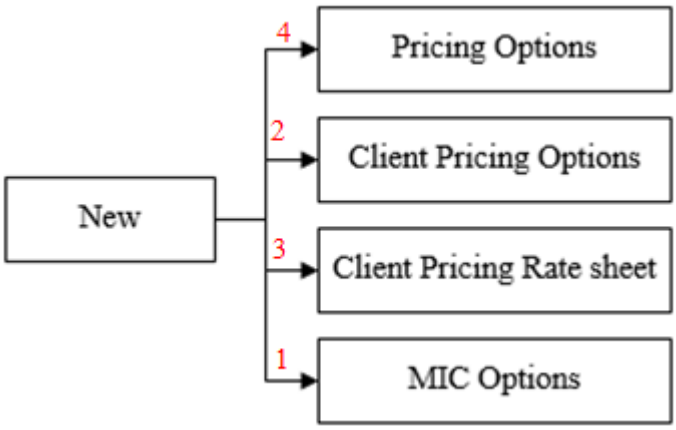


图 4.11 含有新建的输入程序的测试用例

子工作流：定价应用系统→输入→新建

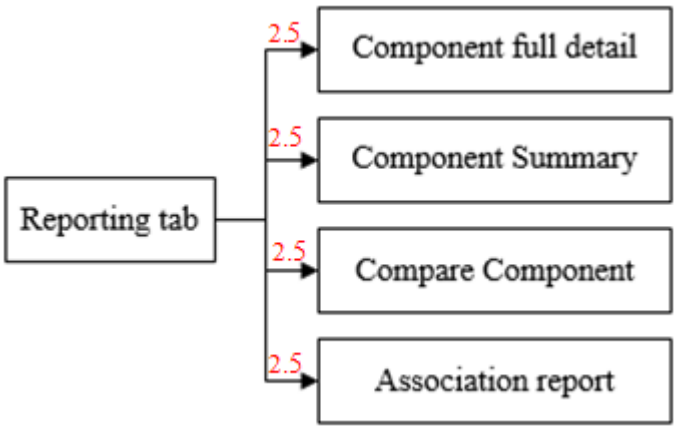


图 4.12 报告程序的测试用例

子工作流：定价应用程序→报告

4.3.3.3 吞吐量评估

系统吞吐量是在一个单位时间内，由服务器处理的工作负载单元的总数，它是系统效率的一个重要指标。许多因素可能会影响系统吞吐量，如负载、软件架构、硬件和系统配置。

（1）理论

吞吐量是指在一个单位时间内，业务系统提供的服务数量。吞吐量通常是系统及其组件处理传输数据请求的能力的总体评价。web 应用程序系统的吞吐量通常以每秒请求数目或页面数目来衡量。理论上，可以得到请求序列以及能够计算

出时间序列内窗口的吞吐量。额外的指标应该从服务器收集，以帮助确定负载级别，系统瓶颈，系统状态和运行的测试，如：CPU 的占用、RAM 的利用、SWAP 利用、IO 速度、网络速度。所有这些数据被整理在一起可用来分析系统的性能。

（2）工作负载特征

因为在评估测试系统响应时间时，吞吐量就可以计算出来，因此，工作负载特性与上文是相同的部分。

（3）解决方案

本节将描述在此定价系统中进行测量吞吐量的推荐方法。根据工作负载分布和工作流，本文将所有可能的路径转换成连续的过程，每个过程将与某一个概率有关。一个过程中的并行虚拟用户数量将为：VUs*P，其中 VU 表示虚拟用户的总数模拟过程中，P 代表一个特定过程的可能性。本文将在不同的负载级别下测量系统吞吐量来预测系统的处理能力。

（4）测试用例

与上文相同，此处不再赘述。

4.3.4 测试场景执行

定价应用程序的性能测试从 2014-09-23 开始一直持续到了 2014-11-23。其具体执行情况参见第五章。

4.3.5 测试结果

测试结果分析参见第五章。

4.4 本章小结

主要讲性能测试的实施，其中包括性能测试工具的选择、平台的选择、方案的制定、环境的设计、场景的设计、场景的执行等。

第5章 性能测试的结果及分析

5.1 测试执行概况

定价应用程序的性能测试开始从 2014-09-23 到 2014-11-23。这个测试基本上是为了评估定价应用程序的性能，以及帮助团队找到潜在的性能相关的瓶颈和错误。本文将采用启发式方法，应用 JMeter 来执行测试用例来评估关键场景的响应时间和吞吐量。JMeter 的总结报告是用于记录的最大、平均、最小响应。性能测试工作可以彻底揭示定价的应用程序的质量，在生产之前减少潜在风险。以下部分将简要描述重要测试场景，具体为：测试的策略、目的、结果、监测数据和结论。从中得出结论，基于测试数据、监测数据，对分析和调优进行分析。

5.2 执行结果

基于应用程序，按照不一样的类别区分测试策略，每个类别有各自的目标和不一样的输出。

5.2.1 执行摘要

5.2.1.1 重要的观测结果

在本节中将给出重要的统计数据和观测。下文会对详细测试数据描述。以下是关键的观察：

（1）饼图（如图 5.1）显示所有测试网页或动作的响应时间的排名比例。我们模拟 1vu、5vu、10vu、15vu 和 20vu 分别访问 Argus 系统。下一线程会在上一线程开始 8 秒钟之后才开始执行。各虚拟用户各执行 10 次测试。每个操作之间的思考时间是 0.3 秒，响应时间一般分为五种类型，如下：优秀 Excellent（<1s）、好 Good（1s-2s），可接受 Acceptable（2s-5s），持平 Fair（5s-10s）、差 Poor（>10）。

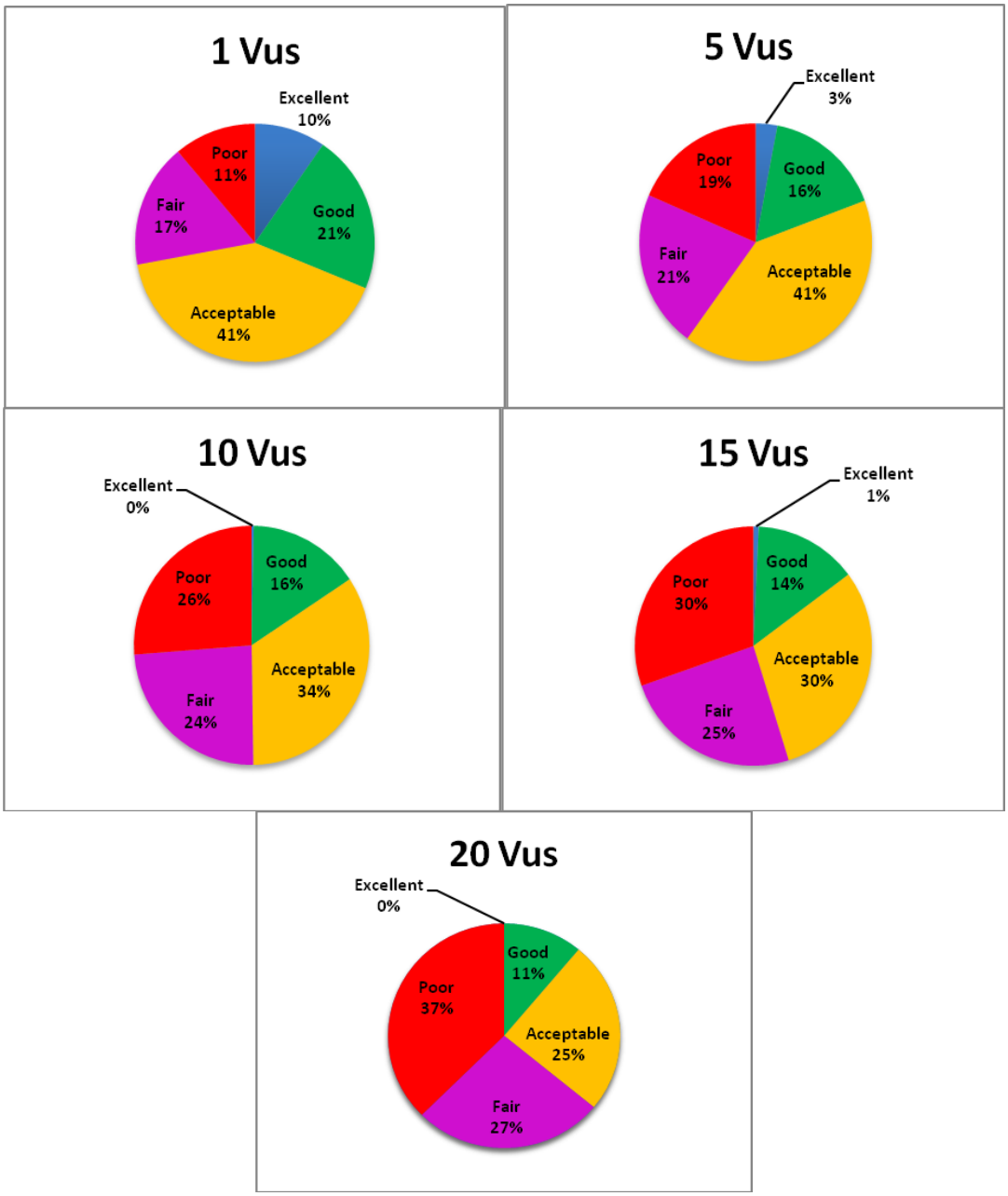


图 5.1 不同的虚拟用户数情况下，响应时间比例

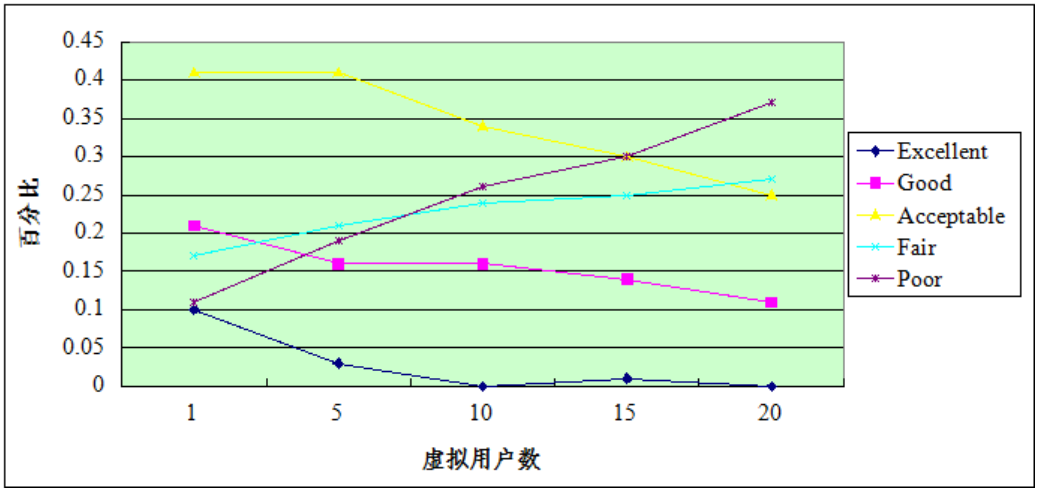


图 5.2 虚拟用户数-响应时间比例趋势图

由图 5.2 可以看到，差 Poor 与持平 Fair 的比例所占量最大，而优秀 Excellent 的比例所占量最小，总是小于 5%。必须指出，当虚拟用户超过 10 时，差和持平总是超过 50%，也需要注意，可接受 Acceptable 的比例总是两倍于好 Good 的比例。

（2）在不同的测试负载下运行所有的测试用例后，发现系统总是花费太多的时间与数据库进行交互。例如：当点击保存、编辑或细节按钮时，响应时间总是太高于其他动作。

（3）测试在 20 个虚拟用户情况下，CPU 的使用使用情况，如图 5.3 所示：

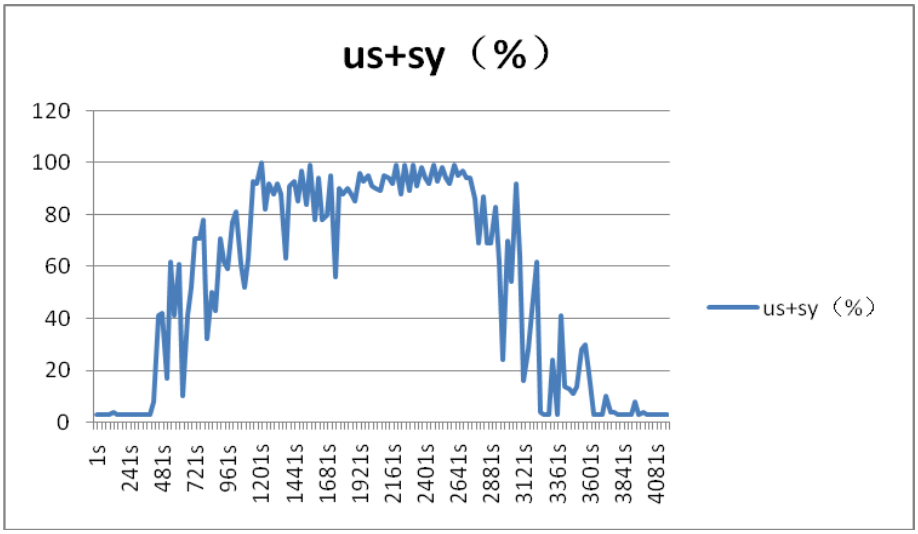


图 5.3 20 个虚拟用户情况下，CPU 的使用率

us（user CPU）：为用户任务服务的 CPU 数量。

sy（system CPU）：为系统任务服务的 CPU 数量的百分比。

需要注意，所有的 CPU 度量均表示为百分数。在 20 虚拟用户访问定价模块子系统的情况下，CPU 值（us + sy）总和达到了 90%。由于测试环境通常也是开发环境，其他一些项目将会产生许多处理流程与该模块一起运行，这个测试的准确性会受到这些因素的影响。

（4）子系统的平均响应时间，如图 5.4 所示：

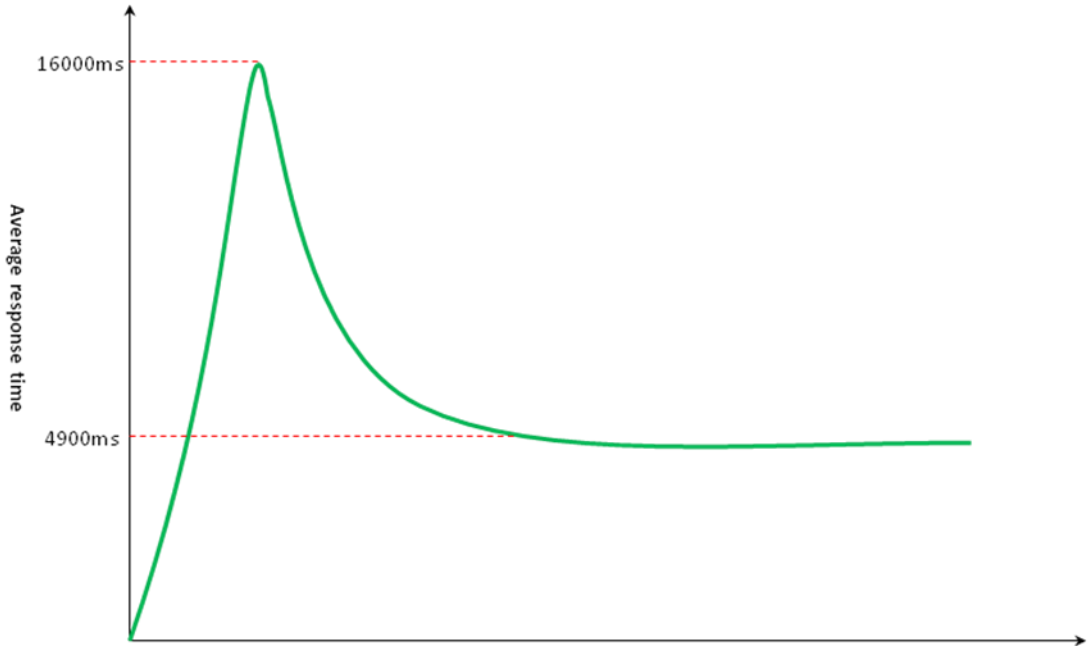


图 5.4 系统平均响应时间

由 JMeter 记录的响应时间应该是服务器响应时间（Ts）和网络响应时间（Tn）的总和。每 10 秒持续地 ping 系统服务器，得出，系统平均响应时间是 292ms。上面的曲线图显示，在开始时间内，平均响应时间增长迅速增长并在很短的时间到达最高点。然后该走势渐渐减少，保持在 4900ms 的稳定值。有三个方面会让系统出现上述现象：

①在初始阶段，包含性能不足，这个在未来得到优化。例如，它总是花费太多的时间进行数据库查询；

②系统是使用缓存机制，它不是一个性能不足；

③固有现象是由于系统架构设计引起的。例如，系统不会建立应用程序服务器与数据库的连接直到收到第一个请求，在随后的阶段该连接不会被释放。在与系统架构、开发人员和最终用户确认之前，该现象不能被定义为一个性能不足。

5.2.1.2 重要的结论

由以上分析可以得出以下的结论：

（1）从这些饼图可以清楚地得出结论，当虚拟用户超过 10 时，整个系统性能不太好；

（2）系统总是花费太多的时间与数据库进行交互，这个是由高 CPU 使用率引起的。

（3）最耗时的前十大性能动作行为，如表 5.1 所示：

表 5.1 最耗时的前十个行为动作

最耗时的前十大行为动作	平均相应时间（ms）
New Client Pricing Rate Sheet/Pricing Options	65818
Pricing Home-Search-Client Pricing Options/MIC Options	64424
Pricing Home-Update Tags/Notes	64169
Pricing Application-Import-Search	60572
Pricing Home- Search-Client Pricing Rate sheet-Update-Detail	59673
Pricing Application-Import-New	49133
New a Priority to the table-Network Rate section-New a Priority to the table	42095
Pricing Home-Update Tags/Notes tab-search	36998
Pricing Home-Update Tags/Notes tab-Detail	33540
Create- Client Pricing Rate Sheet-click-next1-button	28682

5.2.2 并发瓶颈测试

5.2.2.1 并发瓶颈测试策略

在线程组，分别为不同用户设置参数值，为每个用户设置主见增加的足够时间来完成登录过程，并设置循环次数。对于每一个测试计划，进行多次来提高准确率。设置用例包括：确定所有可能的操作来测试系统的总体性能。如：web 容器池大小=100，模拟 1vu，20vu，50vu，100vu 和 200vu 分别访问系统。所有线程将在 5 秒（如果改时间太小，在测试结果，发现并发压力发生在登录过程）。各虚拟用户应该各执行 1 次测试。每个操作之间的思考时间是 0.3 秒。在该用例中，只测试 POC 段的定价搜索功能的性能（BS：运行所有操作需要超过 1PC）。

users=1，loop=1

users=20，loop=1

users=50，loop=1

users=100，loop=1

users=200，loop=1

5.2.2.2 并发瓶颈测试目标

该线程组模拟不同用户对服务器执行一个特定的测试用例。目标是该程序在不同的各个并发量时的性能。用于在高并发的情况下发现系统的瓶颈，包括：CPU 压力，内存压力，EJB/WEB 线程池压力，DB2 连接池压力。

5.2.2.3 并发瓶颈测试结果

（1）平均时间

表 5.2 并发瓶颈测试响应时间

测试时间	2014.11.14		
用例	Pricing Option-View		
web 池大小	100		
JMeter 位置	Local/ZDEV		
用户数	循环次数	ramp-up（秒）	平均时间（秒）
1	1	0	2.7
20	1	5	17
50	1	5	20
100	1	5	74
200	1	5	180

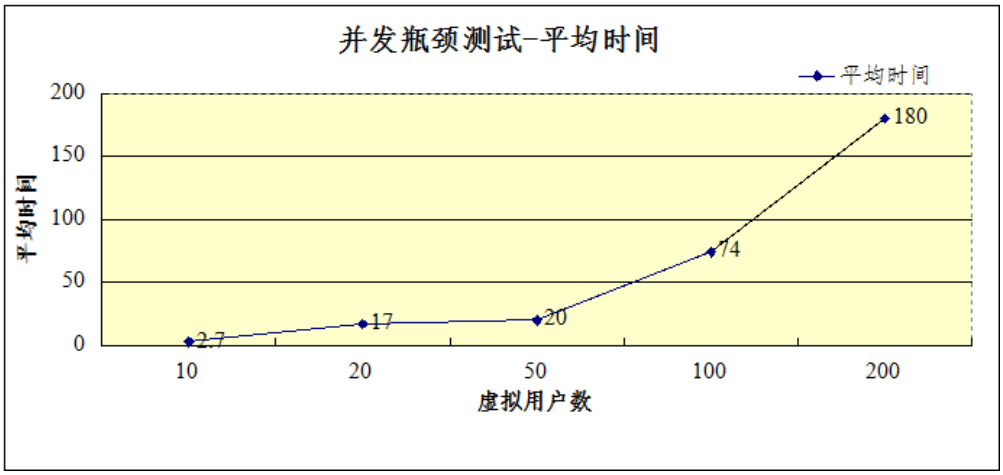


图 5.5 虚拟用户数与平均响应时间变化图

（2）web 服务器测试症状

系统状态监控指标如下所示：

①线程池使用；②JVM 堆使用情况；③内存；④Zdev。

通过以下两个监控工具进行观测以上的各个参数。

①WebSphere PMI

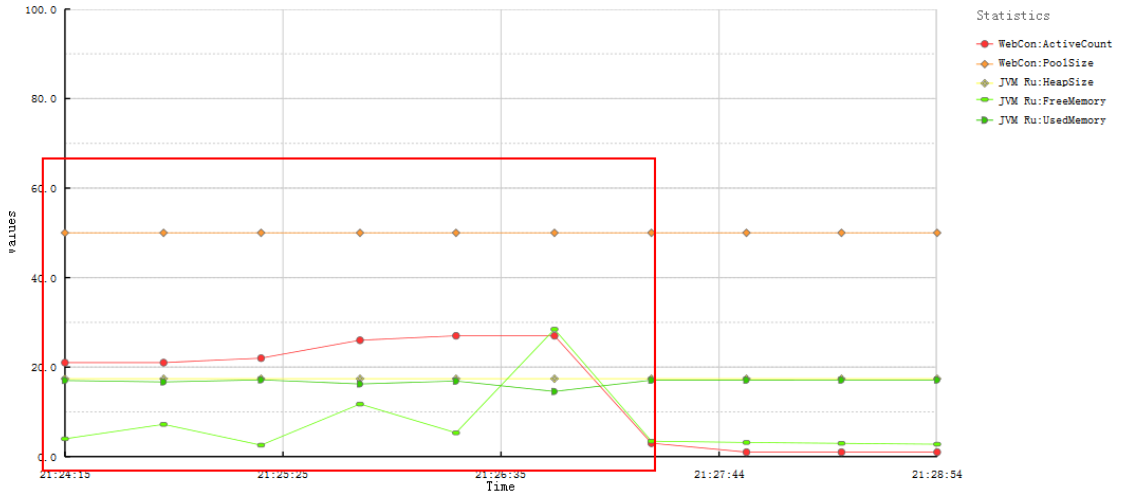


图 5.6 并发瓶颈测试 web 服务器 WebSphere PMI，用户数=20

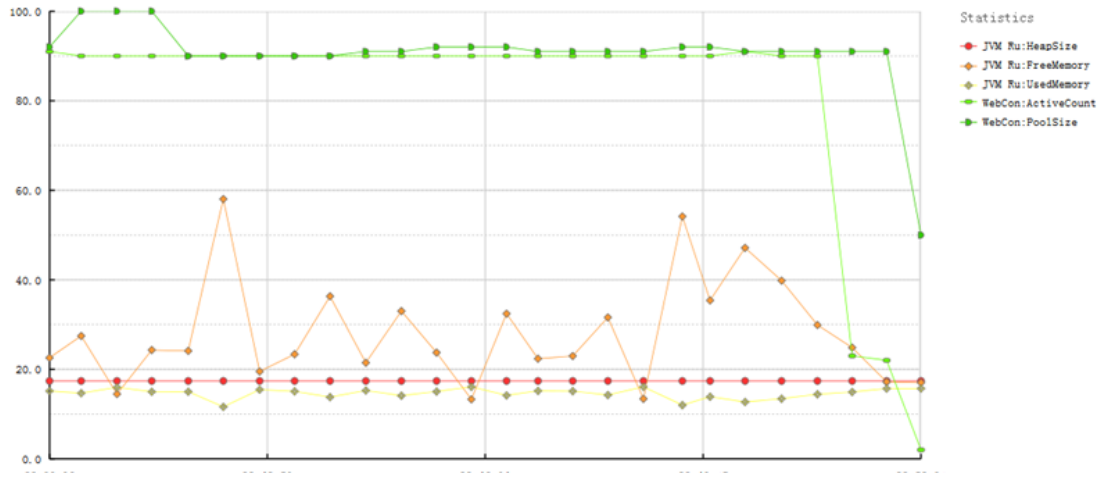


图 5.7 并发瓶颈测试 web 服务器 WebSphere PMI，用户数=100

在图 5.6，用户数为 20，在测试运行期间活动线程大概是 20~24，这意味着线程池中的大多数线程不忙。当 JVM 的内存使用量达到 JVM 堆大小时，许多内存是空闲的，没有发现内存问题。图 5.7，用户数为 100，在测试正在运行期间，活动线程数大概是 90~100，这意味着大多数的 JVM 线程正在忙。当 JVM 的内存使用量达到 JVM 堆大小时，许多内存是空闲的，也没有发现内存问题。

②Vmstat 命令

vmstat 1 1000 | (while read -r line; do echo "$(date +'%Y-%m-%d %H:%M:%S'): $line"; done) > /usr/IBM/WebSphere/AppServer/profiles/AppSrv/jvm-agents/systemPerformenceDuringTest.txt

在 PuTTY 中，输入以上命令，每一秒钟观测系统状态 1000 次，观测数据都被记录在名称为 systemPerformenceDuringTest 的文件中。

```
21:24:57: r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa   pc    ec
21:24:57: 17  0 2945257 510084   0   0   0   0    0   0 126 224627 5048 84 13  3  0  0.69 138.0
21:24:58: 7  0 2945175 510246   0   0   0   0    0   0  30 30012 2928 32 33 35  0  0.62 124.9
21:24:59: 16  0 2944340 511084   0   0   0   0    0   0   5 9564 2138 22 14 64  0  0.28  56.0
21:25:00: 6  0 2944337 511043   0   0   0   0    0   0 108 128988 7299 49  8 43  0  0.34  67.3
21:25:01: 14  0 2944336 510893   0   0   0   0    0   0  96 81306 15996 86  8  6  0  0.51 102.3
21:25:02: 13  0 2944336 510880   0   0   0   0    0   0  42 24002 2269 92  4  4  0  0.50 100.7
21:25:03: 14  0 2944336 510867   0   0   0   0    0   0  44 607830 42140 86 13  1  0  0.51 102.5
21:25:04: 14  0 2944336 510844   0   0   0   0    0   0  28 1383658 13441 71 23  6  0  0.56 112.4
21:25:05: 18  0 2944336 510815   0   0   0   0    0   0  57 822847 28236 87 12  1  0  0.69 138.6
21:25:06: 8  0 2944336 510769   0   0   0   0    0   0  46 1325824 24750 82 16  2  0  0.78 156.9
21:25:07: 8  0 2944336 510743   0   0   0   0    0   0   5 1224701 270428 72 26  2  0  0.69 138.3
21:25:08: 28  0 2944336 510743   0   0   0   0    0   0   0 1096320 140686 73 23  4  0  0.62 125.0
21:25:09: 20  0 2944336 510743   0   0   0   0    0   0   0 2275 2915 98  1  0  0  0.67 134.9
21:25:10: 20  0 2944336 510743   0   0   0   0    0   0   0 2205 2711 99  1  0  0  0.58 116.6
21:25:11: 21  0 2944336 510703   0   0   0   0    0   0  32 117284 3295 90  5  5  0  0.60 120.5
21:25:12: 7  0 2944336 510290   0   0   0   0    0   0 178 37154 2900 88  5  7  0  0.68 136.0
21:25:13: 13  0 2944336 510094   0   0   0   0    0   0 114 45984 2836 90  5  5  0  0.68 136.9
21:25:14: 3  0 2944336 510076   0   0   0   0    0   0  36 364800 14679 56 10 33  0  0.66 131.3
21:25:15: 14  0 2944336 510071   0   0   0   0    0   0  23 698506 26225 86 13  1  0  0.68 136.8
21:25:16: 14  0 2944336 510051   0   0   0   0    0   0  64 901779 11642 87 13  0  0  0.68 136.5
21:25:17: kthr    memory             page              faults           cpu
```

图 5.8 并发瓶颈测试 web 服务器 Vmstat，用户数=20，测试中

在图 5.8 中，users=20，r<20，大多数时候 CPU 没有压力。pi/po 值总是 0，没有发生页面换入/换出现象，这意味着没有内存压力。

```
03:50:28: ----- ----------- ---------------------- ------------ ----------------------
03:50:28: r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa   pc    ec
03:50:28: 8  0 2820520 652759   0   0   0   0    0   0 186 78327 4820 38  9 53  0  0.34  68.7
03:50:29: 9  0 2820538 652701   0   0   0   0    0   0  96 369975 30143 60 15 25  0  0.45  89.0
03:50:30: 8  0 2820536 652674   0   0   0   0    0   0 116 706615 89599 67 23 11  0  0.58 115.2
03:50:31: 1  0 2820536 652638   0   0   0   0    0   0 118 443334 4225 73 11 16  0  0.68 135.2
03:50:32: 8  0 2820555 655137   0   0   0   0    0   0 163 67698 2657 49 10 41  0  0.43  86.8
03:50:33: 10  0 2820556 655054   0   0   0   0    0   0 173 59419 2356 56 10 34  0  0.49  97.9
03:50:34: 4  0 2820556 655053   0   0   0   0    0   0  48 3350 2098  7  4 89  0  0.09  18.7
03:50:35: 1  0 2820556 655031   0   0   0   0    0   0 104 3315 2248 14  4 82  0  0.15  29.2
03:50:36: 1  0 2820556 655010   0   0   0   0    0   0  81 28481 5289 21  7 72  0  0.23  45.0
03:50:37: 17  0 2820556 654967   0   0   0   0    0   0 195 81277 10517 31  9 60  0  0.30  60.6
03:50:38: 1  0 2820558 654925   0   0   0   0    0   0 237 25024 2447 34  6 59  0  0.31  61.3
03:50:39: 14  0 2820558 654890   0   0   0   0    0   0  93 3854 2243 12  6 83  0  0.15  29.3
03:50:40: 1  0 2820558 654853   0   0   0   0    0   0  40 10069 2124  7  5 88  0  0.10  20.3
03:50:41: 5  0 2820558 654812   0   0   0   0    0   0  95 3454 2291 31  5 64  0  0.27  54.3
03:50:42: 0  0 2820572 654693   0   0   0   0    0   0 230 13146 2349 53  7 40  0  0.48  95.1
03:50:43: 1  0 2820570 654609   0   0   0   0    0   0 165 25259 2296 16  6 78  0  0.18  36.2
03:50:44: 1  0 2820562 654528   0   0   0   0    0   0 352 68649 2624 33  8 59  0  0.33  66.0
03:50:45: 0  0 2820564 654485   0   0   0   0    0   0  92 30942 2214 29  6 65  0  0.29  58.7
03:50:46: 0  0 2820561 654412   0   0   0   0    0   0 380 72312 2490 22  8 71  0  0.24  47.0
03:50:47: 10  0 2820622 654133   0   0   0   0    0   0 574 506958 65775 56 16 27  0  0.69 138.4
```

图 5.9 并发瓶颈测试 Vmstat，用户数=100，测试开始

从图 5.9 可知，最初，r 值非常小（<<20），这意味着没有 CPU 压力。

```
03:51:08:  r  b   avm    fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa    pc    ec
03:51:08: 92  0 2822214 653096   0   0   0   0    0   0 101 234055 114225 86 14  0  0  0.63 126.7
03:51:09: 51  0 2822471 652534   0   0   0   0    0   0 136 486399 77477 85 14  1  0  0.69 138.9
03:51:10: 68  0 2822839 652166   0   0   0   0    0   0  25 1141357 245095 70 30  0  0  0.54 108.4
03:51:11: 94  0 2822984 651629   0   0   0   0    0   0 100 348650 171626 84 15  0  0  0.65 131.0
03:51:12: 74  0 2822984 651230   0   0   0   0    0   0 162 181169 41236 90 10  1  0  0.69 138.6
03:51:13: 42  0 2822984 650829   0   0   0   0    0   0 171 399257 85014 87 13  0  0  0.69 138.2
03:51:14: 60  0 2823001 650408   0   0   0   0    0   0 117 676296 222559 79 21  0  0  0.68 135.8
03:51:15: 62  0 2823001 650408   0   0   0   0    0   0  22 1100659 577945 66 33  1  0  0.71 141.6
03:51:16: 54  0 2823001 650383   0   0   0   0    0   0  18 1439555 591642 61 39  0  0  0.69 138.8
03:51:17: 91  0 2823003 652428   0   0   0   0    0   0 116 320646 211666 84 16  0  0  0.69 138.3
03:51:18: 71  0 2823003 652017   0   0   0   0    0   0 131 682271 75230 84 16  0  0  0.68 135.5
03:51:19: 69  0 2823020 651646   0   0   0   0    0   0 123 496063 151449 84 15  1  0  0.70 139.1
03:51:20: 81  0 2823020 651470   0   0   0   0    0   0 103 928481 354167 76 24  0  0  0.75 150.7
03:51:21: 58  0 2823020 651467   0   0   0   0    0   0   5 1627360 668660 55 45  0  0  0.69 137.6
03:51:22: 49  0 2823020 651463   0   0   0   0    0   0   4 1486447 607159 60 39  0  0  0.69 139.0
03:51:23: 47  0 2823020 651459   0   0   0   0    0   0   2 1450156 555614 58 42  0  0  0.64 127.0
03:51:24: 38  0 2823020 651444   0   0   0   0    0   0  11 1251162 697578 61 39  0  0  0.69 138.5
03:51:25: 87  0 2823020 651443   0   0   0   0    0   0  26 1185334 501914 69 31  0  0  0.70 140.5
03:51:26: 76  0 2823020 651435   0   0   0   0    0   0  21 894690 657512 66 34  0  0  0.69 137.5
03:51:27: 82  0 2823036 651416   0   0   0   0    0   0  29 1062698 765561 63 37  0  0  0.74 147.7
03:51:28: kthr    memory             page            faults           cpu
```

图 5.10 并发瓶颈测试 web 服务器 Vmstat，用户数=100，测试中

如图 5.10 所示，在测试期间，r 值增长非常大（>>20），保持了很长一段时间，这意味着发生了 CPU 压力。pi/po 总是 0，没有发生页面换入/换出，这意味着没有内存压力。在这段时间里，Zdev 反应非常缓慢。完成测试 10 分钟后，Zdev 响应恢复正常。在一些测试，Zdev 由于太慢而宕掉，但停止测试后，发现 Zdev 反应也恢复正常。

（3）EJB 服务器测试症状

下面是被监控的症状：

①对象远程代理线程池使用；②JVM 堆使用；③内存使用；④数据库连接池使用。

通过以下两个监控工具获取以上两个指标的数据。

①WebSphere PMI

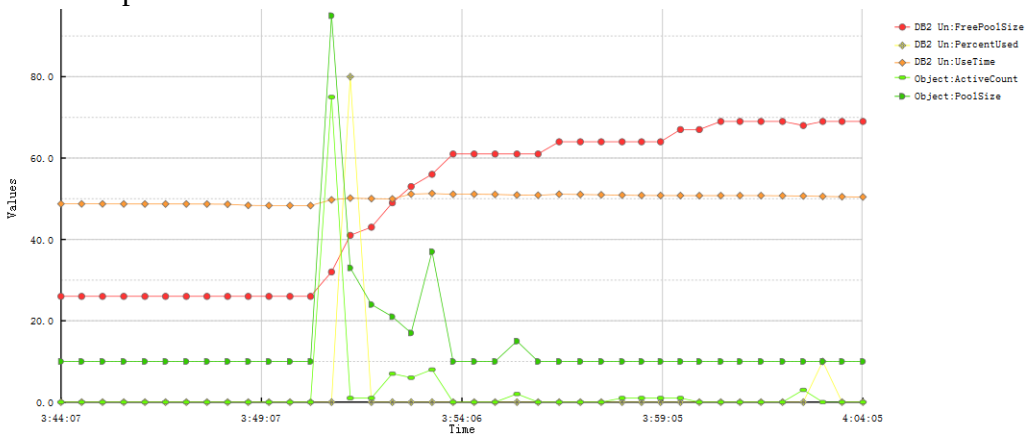


图 5.11 并发瓶颈测试 EJB 服务器，用户数=100

如图 5.11 所示，在一开始，EJB ORB 活跃计数，发生了近 95EJB 远程调用，

然后快速下降并保持稳定在约 10。这意味着大部分时候，很少发生 EJB 远程调用。DB2 的连接池使用率也很非常缓慢。可以看到，此场景中，EJB 站点和 DB2 连接没有压力。

②Vmstat 命令

```
03:50:14: ----- ----------- ------------------------ ------------ -----------------------
03:50:14:  r  b   avm   fre  re  pi  po  fr  sr  cy  in   sy  cs us sy id wa   pc    ec
03:50:14:  3  0 3358269 1440185   0   0   0   0   0   0    2 1294 1210  3  1 96  0  0.04   7.7
03:50:15:  3  0 3358275 1440178   0   0   0   0   0   0    2 4162 1231  1  2 96  0  0.03   6.6
03:50:16:  3  0 3358275 1440178   0   0   0   0   0   0    3 1733 1365  1  2 98  0  0.02   4.6
03:50:17:  3  0 3358275 1440178   0   0   0   0   0   0    0 1554 1362  1  1 98  0  0.02   4.3
03:50:18:  3  0 3358275 1440178   0   0   0   0   0   0    0 1554 1368  1  1 98  0  0.02   4.1
03:50:19:  1  0 3358275 1440178   0   0   0   0   0   0    3 1740 1368  1  2 97  0  0.03   5.4
03:50:20: 31  0 3358275 1440178   0   0   0   0   0   0  175 293771 15063 43 10 48  0  0.35  70.8
03:50:21:  0  0 3358275 1440178   0   0   0   0   0   0    4 353364 83393 16 10 74  0  0.16  31.8
03:50:22:  1  0 3358275 1440178   0   0   0   0   0   0    3 1668 1195  1  2 98  0  0.02   4.6
03:50:23:  0  0 3358275 1440177   0   0   0   0   0   0  326 16883 1534  8  3 89  0  0.09  18.8
03:50:24:  3  0 3358275 1440177   0   0   0   0   0   0  676 799755 3366 60 22 17  0  0.51 101.9
```

图 5.12 并发瓶颈测试 EJB 服务器，用户数=100，测试前

如图 5.12 所示，r 值非常小。ORB 池大小=10，ORB 活跃数=0，没有 EJB 远程调用发生。DB2 空闲池大小=25，DB2 使用率=0，没有 DB2 操作。

```
03:50:54:  r  b   avm   fre  re  pi  po  fr  sr  cy  in   sy  cs us sy id wa   pc    ec
03:50:54: 33  0 3360484 1437928   0   0   0   0   0   0  710 612418 26712 81 17  2  0  0.49  98.3
03:50:55: 11  0 3360484 1437925   0   0   0   0   0   0  445 835040 91704 74 23  3  0  0.50  99.2
03:50:56: 27  0 3360804 1437603   0   0   0   0   0   0  453 675305 31944 80 18  2  0  0.49  98.5
03:50:57: 17  0 3360948 1437441   0   0   0   0   0   0  535 839973 75335 73 22  5  0  0.51 102.1
03:50:58: 51  0 3361076 1437279   0   0   0   0   0   0  293 1369522 234916 61 37  2  0  0.49  98.4
03:50:59: 41  0 3361076 1437213   0   0   0   0   0   0  864 503226 11119 82 15  3  0  0.49  98.7
03:51:00: 38  0 3361076 1437209   0   0   0   0   0   0  398 633736 42309 81 17  2  0  0.49  98.9
03:51:01: 16  0 3361076 1437141   0   0   0   0   0   0  502 530756 5896 82 14  4  0  0.51 102.0
03:51:02:  9  0 3361076 1437141   0   0   0   0   0   0  569 706612 13559 67 21 12  0  0.49  98.4
03:51:03: 12  0 3361076 1437141   0   0   0   0   0   0  129 128651 1783 69  6 25  0  0.52 103.5
03:51:04: 10  0 3362612 1435605   0   0   0   0   0   0    0 2343 1520 61  3 36  0  0.50 100.2
```

图 5.13 并发瓶颈测试 EJB 服务器，用户数=100，测试开始

如图 5.13 所示，一开始，ORB 活跃数迅速出现峰值，此阶段 ORB 池大小=100，ORB 活跃数=75，说明 EJB 端线程池增加至 100，EJB 被调用，活跃的线程有 75 个，此时 DB2 使用百分比为 80%，说明有数据库操作。DB2 空闲池大小为 40，说明连接池仍然有剩余。

```
03:53:40:  1  0 3363949 1435195   0   0   0   0   0   0  119 2626 1608 16  3 81  0  0.15  30.0
03:53:41:  8  0 3363949 1435194   0   0   0   0   0   0  360 125811 15401 77  9 14  0  0.51 102.3
03:53:42: 16  0 3363950 1435185   0   0   0   0   0   0  233 489123 134165 77 18  5  0  0.50  99.8
03:53:43:  3  0 3363950 1435184   0   0   0   0   0   0  186 23301 1761 63  5 32  0  0.51 101.1
03:53:44:  3  0 3363950 1435184   0   0   0   0   0   0   72 2631 1495 21  3 75  0  0.20  39.9
03:53:45:  2  0 3363950 1435183   0   0   0   0   0   0  290 2348 1609 42 13 45  0  0.43  86.1
03:53:46:  1  0 3363950 1435183   0   0   0   0   0   0   33 2185 1456  4  2 94  0  0.05  10.2
03:53:47:  9  0 3363950 1435183   0   0   0   0   0   0   16 13217 1410  3  2 95  0  0.05  10.8
03:53:48:  1  0 3363950 1435183   0   0   0   0   0   0   23 1825 1370  3  2 96  0  0.04   7.8
03:53:49:  1  0 3363950 1435183   0   0   0   0   0   0   17 1979 1437  2  2 96  0  0.04   7.7
03:53:50:  1  0 3363950 1435183   0   0   0   0   0   0   29 1991 1538  4  2 94  0  0.05   9.4
03:53:51:  1  0 3363950 1435183   0   0   0   0   0   0   15 1805 1555  2  2 96  0  0.04   7.8
03:53:52:  1  0 3363950 1435182   0   0   0   0   0   0   21 1993 1462  3  2 95  0  0.05   9.3
03:53:53:  2  0 3363950 1435182   0   0   0   0   0   0  114 3390 2003 36  3 62  0  0.28  55.6
```

图 5.14 并发瓶颈测试 EJB 服务器，用户数=100，测试中

通过图 5.14，可以看出，开始测试之后两分钟：DB2 使用率=0，ORB 活跃数为 7，之后长时间接近于 0，偶尔 DB2 使用率为 10，ORB 活跃数为 4，说明 EJB 操作频率非常低，DB2 操作频率更低，这和监测到 EJB 端 CPU 压力下降的现象是相符的。

5.2.2.4 结论与调优

当虚拟用户个数达到 20 时，系统搜索功能不能运行良好。为了测试系统的总体性能，设置用例，包括所有操作，这些操作有一定的可能性。

用例，如：用户 VUs=100，循环 Loop=1。

Web 服务器启动没有压力。在一个时间点，Web 服务器开始有压力，r 值不断增加这意味着越来越多的线程正在等待 CPU，CPU 使用率几乎达到 90%。处理请求的时间增加。但是系统还没有死掉。内存使用量在正常水平。一开始，EJB 服务器压力。原因是，Web 服务器达到了瓶颈，EJB 服务器的压力减少。

5.2.3 内存泄露测试

5.2.3.1 内存泄露测试策略

为用户设置适当的值（不高并发量以至于不影响系统性能），设置更多次的循环，设置用例包括所有具有一定可能性的操作。循环多次以降低错误，提高准确性。例如：Loop>1000

5.2.3.2 内存泄露测试目标

遍历主要应用场景来找出可以产生内存泄漏的任何的原因。

5.2.3.3 内存泄露测试结果

（1）平均时间

表 5.3 内存泄露测试的平均时间

时间	用例	用户数	Web 池大小	Loop	ramp-up（秒）	JMeter 位置	平均时间（秒）
11/15	所有操作	14	100	1000	200	Local/ZDEV	7
10/29	所有操作	7	50	100	0	Local/ZDEV	2.4
11/16	所有操作	15	100	10000	500	Local/ZDEV	1.62

11/18	所有操作	50	100	5000	500	Local/ZDEV	n/a
11/19	所有操作	50	100	5000	500	Local/ZDEV	5

在 11/19 用例中，4 电脑测试运行了 6 小时。但是在每个电脑，JMeter 每半个小时杀死一个线程。随着时间，它会导致平均时间下降。在 11/18 用例中，Zdev 因为太慢而宕掉。但是测试终止，Zdev 的响应就又变成了正常的状态。同样的结论也已经从之前测试得出过，所以可以说，没有发生内存泄漏情况。

（2）web 服务器测试症状

监视状态线程池使用、JVM 堆使用、内存、Zdev，随着时间，发现 Web 服务器压力不上升，内存使用处于正常水平。这样看来，此子系统没有内存链接问题。

（3）EJB 服务器测试症状

通过监视 EJB 服务器的状态，如：对象远程代理（ORB）线程池使用、JVM 堆使用、内存使用、数据库连接池使用，没有发现压力。

5.2.3.4 结论与调优

此定价系统没有在内存泄漏方面的缺陷或者说是问题。

5.2.4 功能性能测试

5.2.4.1 功能性能测试策略

测试用例包括某些操作，如：搜索、更新、删除、质量、批量变化。

5.2.4.2 功能性能测试目标

对于一个操作的性能展开相关的测试，它关注特定步骤或缓慢的一步。它有两个用例需要测试：①系统中主要的操作场景，测试平均响应时间，大多数的主要操作测试完成之后，发现最慢的操作和步骤。它是优化系统的的一个方向。②可能有特殊的要求的一些操作场景，如：有更多的入口或有更多的日期范围，这个测试是寻找缓慢操作或步骤一个方法。

5.2.4.3 功能性能测试结果

（1）平均时间

每一个操作的平均时间是 2~3 秒钟。

（2）web 服务器测试症状

没有发现压力。

（3）EJB 服务器测试症状

没有发现压力。

5.2.5 实际生产环境性能

5.2.5.1 实际生产环境中性能测试的策略

设置生产环境中实际用户量的 10 倍的数量和循环次数也设置 10 倍。

5.2.5.2 实际生产环境中性能测试目标

对系统在时间生产环境下的性能展开测试。

5.2.5.3 实际生产环境中性能测试的结果

（1）平均时间

每一个操作的平均时间是 2~3 秒钟。

（2）web 服务器测试症状

没有发现压力。

（3）EJB 服务器测试症状

没有发现压力。

5.2.5.4 结论与调优

在生产环境中，定价应用程序没有性能问题。

5.2.6 排除网络传输的系统性能

5.2.6.1 测试策略

在应用程序 web 服务器部署主机上运行 JMeter，在这种情况下 DO2-pricing 网站部署在 232 主机上。

5.2.6.2 测试目标

测试在实际生产环境中系统的性能情况。

5.2.6.3 测试结果

（1）平均时间

表 5.4 排除网络传输的性能测试的平均时间

日期	用例	用户数	Web Pool Size	Loop	ramp-up（秒）	JMeter Location	平均时间
12/10	View Pricing option	1	100	10	0	Local/ZDEV	3.4
12/10	View Pricing option	1	100	10	0	ZDEV/ZDEV	1.4

5.2.6.4 结论与调优

在一个响应实践中，网络传输花费了一半的时间。真正的生产不会有太多的网络传输，它将会比 HT 站点的性能更好。

5.3 本章小结

对系统进行并发瓶颈、内存泄漏、功能性能、实际生产环境性能、排除网络传输的性能等的测试，并从以上几个方面进行结果分析并提出调优方案。

第6章 总结与展望

6.1 总结

本文在对软件性能测试概括描述以及基于web的医疗保险信息管理系统概述的基础上，展开了软件性能测试瓶颈与需求的分析，对这个系统软件的性能测试展开相关的深入研究，进行相应的性能方面的测试，然后对通过测试得出的结果展开分析。

本文主要通过基于web的医疗保险信息管理系统的定价子系统的功能和业务的角度进行需求分析，对本系统有个全面的了解，设定本次性能测试的目标。并根据本系统定价子系统的数据源和工作负载特征设计测试用例。在这里选择JMeter作为此医疗保险信息管理系统的测试工具进行性能测试，性能测试的实施主要包括性能测试工具的选择、相关平台的选择、相关方案的制定、环境的设计、场景的设计、相关场景的执行等。主要对系统进行并发瓶颈、内存泄漏、功能性能、实际生产环境性能、排除网络传输的性能等的测试。

本次软件性能测试主要是为了找出此基于web的医疗保险信息管理系统中有可能存在的性能问题和有可能存在的瓶颈，本次测试是需要结合这个基于web的医疗保险信息管理系统的特征，不只是使用JMeter等的相关工具，运行相关的脚本，来看软件的性能究竟如何，最主要的是得出软件在性能方面的BUG，找到问题具体出现的地方。

6.2 展望

软件性能测试对软件质量的保障有着很重要的作用，我国的软件测试行业对性能测试的研究水平还比较低，而且研究开发的用来进行测试的有关工具的数量很少，这或许与国内的软件开发的发展有一定得紧密联系。国内的软件测试还处于成长时期，若想超过国外的发展水平还需要较长时期的努力。但是在国内，软件测试还属于新兴的行业，其发展速度还是非常快的，相信随着国内软件行业及知识产权保护的发展，性能测试的研究必定会越来越成熟。

参考文献

[1]赫建营,晏海华,刘超,金茂忠.一种有效的 Web 性能测试方法及其应用[J].计算机应用研究,2007(01):275-277+285.

[2]宫云战.软件测试[M].北京:国防工业出版社.2006.

[3]景宏磊,林丁报.软件性能测试的基本概念与一般过程[J].科技资讯,2011(19):22.

[4]蔡立志,杨根兴.软件系统性能测试方法初探[J].信息技术与标准化,2005,07:44-50.

[5]温艳冬.软件性能测试需求的获取方法综述[J].软件工程师,2010(Z1):124-127.

[6]韩明军.软件性能测试过程[J].信息技术与标准化,2007(11):41-43.

[7]韩庆良.软件性能测试过程研究与应用[D].硕士学位论文,山东大学,2007.

[8]赵丽莉,金学军.软件性能测试面面观[J].软件工程师,2006(11):40-42.

[9]郁莲.软件测试方法与实践[M].北京:清华大学出版社,2008.

[10]李健.基于 Web 应用系统的性能测试技术研究[D].硕士学位论文,西安电子科技大学,2010.

[11]易敏捷.软件测试国内外发展现状及趋势研究[J].电脑知识与技术,2013(26):6020-6022.

[12]桑圣洪,胡飞.性能测试工具 LoadRunner 的工作机理及关键技术研究[J].科学技术与工程.2007(06):1019-1020.

[13]Xu Lei, XuBaowen. Research on the Analysis and Measurement for Testing Results of Web Applications[J].Second International Workshop on Web Computing in Cyberworlds, 2005, 559~565.

[14]孙莹,王华伟.软件测试中存在的问题及对策[J].软件导刊,2015(01):50-51.

[15]杨金凤,孟岩.软件性能测试概述[J].电脑知识与技术,2011(34):8886-8888.

[16]陈志皑.软件性能测试初探[J].广东科技,2012(21):185-186.

[17]王爽,李瑞路.软件性能测试研究[J].现代商贸工业,2009(12):296-297.

[18]朱怡雯,钱超,林勇.软件性能测试工具综述[J].中国金融电脑,2009(07):79-82.

[19]佘青.利用 Apache Jmeter 进行 Web 性能测试的研究[J].智能计算机与应用,2012(02):55-57.

[20]温素剑.零成本实现 Web 性能测试-基于 Apache JMeter[M].北京:电子工业出

版社,2012:27-29.

[21]江新.基于 JMeter 的 MS Web 应用系统的性能测试研究[D] .硕士学位论文,南京航空航天大学,2011.

[22]兰景英,王永恒.Web 系统性能测试研究[J].计算机技术与发展,2008(11):90-93.

作者简历

教育经历：

2013 年 9 月~现在　浙江大学　软件工程金融数据分析 硕士研究生

2009 年 9 月~2013 年 6 月　许昌学院　电子商务　本科学士

2006 年 9 月~2009 年 6 月　郸城一高东校区

2004 年 9 月~2006 年 6 月　郸城县又铭中学

工作经历：

2014 年 9 月~现在　恒天 Argus 项目组　软件测试

2014 年 5 月~2014 年 9 月　恒天小微金融项目组　软件测试

攻读学位期间发表的论文和完成的工作简历：

无

致谢

在论文的完成过程中出现了一些令人困惑的问题或难关，是老师和同学们的帮助使得我顺利度过。在这里，尤其要感谢我的导师李善平教授，李老师严谨的治学态度和科学的工作方法深深的影响着我。另外，才振功老师也给我提供了无私的指导和帮助，提出了许多宝贵的意见，在此向两位老师表示最衷心的感谢！

在研究生学习期间，要感谢杭诚方教授、刘二腾老师以及其他学科的老师，是他们的谆谆教诲和无私的付出，让我在研究生学习阶段就可以为论文的写作上打下良好的基础。

还要感谢和我一起学习的同学们，你们积极向上、踏实勤学的精神鼓舞着我，并且在学习期间给予我无数的帮助。

最后，感谢我的家人和朋友，是他们的支持，让我在学习的道路上大踏步的前进！

本文作者的研究水平有限，所以，文中难免存在这样那样的问题，希望各位老师和同学批评指正。

署名

于浙江大学软件学院

当前日期