Bench4Q:一种面向服务质量的电子商务测试基准设计与实现

段智全[1,2] 张文博[1] 王 伟[1,2]
(中国科学院软件研究所软件工程技术研发中心 北京 100190)[1] (中国科学院研究生院 北京 100039)[2]

摘 要 作为电子商务系统的基础运行平台,应用服务器提供的服务质量是关注焦点,然而目前主流的电子商务测试基准如TPC-W,主要关注性能度量,缺乏针对电子商务服务质量需求的设计,使得其难以准确地评价应用服务器提供的服务质量。提出了一种面向服务质量的电子商务测试基准Bench4Q,它在模拟负载仿真、度量分析等多个方面对TPC-W进行了扩展,并通过对典型的应用服务器的测试展示了Bench4Q的设计特征。

关键词 基准测试,中间件,电子商务,TPC-W

Design and Implementation of Bench4Q: A QoS-oriented E-commerce Benchmark

DUAN Zhi-quan[1,2] ZHANG Wen-bo[1] WANG Wei[1,2]
(Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[1]
(Graduate University of Chinese Academy of Sciences, Beijing 100039, China)[2]

Abstract The ability of application server which is the infrastructure of the E-commerce has become a key matter of people's concerns. TPC-W, as a very popular E-commerce benchmark, mainly focuses on performance of application server and is lack of concerns on the QoS guarantees. We designed a QoS oriented E-commerce benchmark named Bench4Q. This benchmark has made many extensions of load simulation and metrics analysis of TPC-W specification. We showed the characteristic of Bench4Q by running it on representative application server.

Keywords Benchmark, Middleware, E-commerce, TPC-W

1 引言

随着Internet的迅速发展,电子商务企业正面临着前所未有的发展机遇。电子商务应用是对服务质量比较敏感的应用模式,提供给最终用户的服务质量将直接影响企业的商业利益[1]。在电子商务领域,大部分企业会使用已有的成熟的应用服务器技术来部署自己的应用。在众多的可供选择的技术当中,企业总是希望存在一个简单的可类比的方法来帮助他们做出选择。测试基准(Benchmark)则是推动该研究工作的重要手段之一,它对电子商务企业选择应用服务器及对已有应用服务器系统的配置进行评估有较大的指导意义。

然而,传统的评价应用服务器的测试基准都仅仅关注于应用服务器的性能,缺乏对应用服务器所能提供的服务质量的评估能力。例如,由TPC(事务处理性能委员会)所提出的TPC-W[2]是目前最流行的电子商务测试基准,但是其在面对电子商务的服务质量需求时仍然存在以下不足:1)负载生成方式单一,是封闭不可控的,不符合电子商务系统开放式负载的需求;2)TPC-W的主要指标是系统的性能度量,用点击率来衡量,不能反映电子商务更关心的服务质量度量,如以会话等以业务特征为中心的度量。本文针对当前基准测试中的不足,设计了一种面向服务质量的电子商务测试基准Bench4Q,该基准从负载仿真和度量分析等方面进行了扩展,并实现了一个Bench4Q测试工具。通过对典型应用服务器的测试,Bench4Q可以评估电子商务企业所需的服务质量。

本文第2节介绍相关工作;第3节描述Bench4Q的系统模型;第4节详细介绍了Bench4Q测试工具的设计与实现;第5节通过实验来展示Bench4Q的特点;最后总结全文。

2 相关工作

TPC-W的主要模拟一个网上书店系统,通过在一个可控的环境下执行一系列的电子商务典型事务来评价系统。TPC-W提供了一个网上书店常用的功能,包括图书的浏览、查询及订购等。围绕这些功能,TPC-W定义了14种事务交互,分为浏览和订单两大类。浏览类型的交互只涉及数据库的读操作,而订单类的交互涉及到数据库的读操作和写操作。按照浏览和订单类事务的比例,TPC-W将测试分为3种混合模式:购物模式、浏览模式和订单模式[3]。

TPC-W的主要关注点是整个系统的性能度量,它衡量系统支持客户浏览和在网站上处理订单的性能。测试的主要指标是在3种混合模式下的WIPS(每秒处理的Web交互次数)

到稿日期:2010-01-21 返修日期:2010-04-07 本文受国家自然科学基金(No. 90718033),国家重点基础研究发展计划项目(No. 2009CB320704),国家高技术研究发展计划项目(No. 2007AA01Z134,2007AA010301,2009AA04Z153)资助。

段智全 男,硕士生,主要研究方向为网络分布计算、软件工程,E-mail:duanzhiquan07@otcaix.iscas.ac.cn;张文博 男,副研究员,主要研究方向为网络分布计算、软件工程;王 伟 男,博士生,主要研究方向为网络分布计算、软件工程。

及$/WIPS(每WIPS的花费)。WIPS的计算方法是用给定测试间隔完成的总的Web交互次数除以测试时间。同时要求这些事务中至少有90%的WIRT(Web交互响应时间)必须是在测试基准给定的范围之内。测试基准还规定系统得到WIPS所使用的负载模型必须是符合某些规定生成的,例如平均思考时间必须在7s左右。通过这些限制,TPC-W避免了一些系统为了得到更好的峰值性能而进行的一些不合理的优化。

TPC-W只是单纯地用性能指标来评价整个系统的优劣,不能反映电子商务系统的服务质量。比如,TPC-W的负载生成方式是封闭的,是基于模拟用户窗口来生成负载的。在这种生成负载的方式下,后一个会话的产生依赖于前一次会话的完成情况,系统存在着会话间的依赖,使得TPC-W不仅难以模拟动态开放的Internet计算环境,而且使得模拟的压力部分取决于待测系统的表现,从而无法实现真正意义上的负载的完全控制,也无法生成真正意义上的过载。这些类似的设计问题使得TPC-W并不能真实地反映电子商务系统能否满足业务需求。

3 Bench4Q模型

本文针对目前测试基准中的不足,研究并提出了一种面向服务质量的测试基准Bench4Q。Bench4Q对TPC-W的负载仿真和度量分析方面进行了扩展,可供电子商务企业选择应用服务器或者评估其系统配置。Bench4Q以会话为基本的组织方式,通过控制负载仿真的特征评估电子商务系统的服务质量表现,并实现以会话为中心的度量分析,从而全面地评估电子商务系统的服务质量特征。本节从负载模型和度量分析两个方面来详细介绍Bench4Q。

3.1 负载模型

负载模型的生成应该基于真实的并发用户行为,负载模型与真实负载的相似程度直接关系着测试结果的正确性。一个错误的仿真可能会影响测试的结果,从而可能让使用者得到完全相反的结论。

在TPC-W中,系统使用一组EB(仿真客户)来产生负载。在实际系统中,一般使用多线程来实现。每个EB通过模拟用户行为来向待测系统发出一系列请求产生压力。该EB结束一次会话后,开始模拟一个新的用户,开始一轮新的会话。测试系统是以线程作为模拟负载的组织基础。EB收到上一次请求的返回结果后开始下一次请求,在结束上一次会话后才会开始下一次会话。在这种情况下,系统会同时存在请求间的依赖关系和会话间的依赖关系。在这种负载生成方式下,会存在以下弊端:

1)系统无法产生真正的持续过载压力[4]。由于系统产生的下一次会话会依赖于当前会话的完成情况,当待测系统超过其自身处理能力的时候,当前的处理可能正在排队,或者队列已满,错误返回。如果当前任务在排队,则无法生成下一个会话,此时系统的实际压力会下降;如果当前任务错误返回,系统会接着发出下一会话,而下一次会话也会像上一次请求一样错误返回,系统会生成许多错误响应的请求,成为无效的负载。上述两种情况都不是测试真正需要的过载的负载压力。

2)系统无法完全控制生成的负载。由于系统产生的压力会部分取决于待测系统的响应及网络传输速度等因素的影响,当待测系统处理慢或者网络延迟比较大的时候,系统会产生比较少的负载;反之,则会生成比较多的负载。因此,每次待测系统的变化或者测试环境的变化都会引起负载模型所产生的实际负载的变化,甚至不同时间在相同测试条件下进行的测试可能会受到网络等因素的影响,产生不同的测试压力,导致测试结果的不同。在这种情况下,待测系统的不同表现失去了对比的基准,使用户无法对结果进行正确的判断。

不同于TPC-W的封闭式负载模型,Bench4Q采用开放的负载模型,如图1所示。在这种模型下,不是以线程或者进程作为模拟负载的基本单位,而是以用户为中心来组织负载生成。系统的基本生成负载单位是会话。开放模式去除了不必要的会话间依赖,使得系统生成的会话完全相互独立,这样保证了在测试过程中每次生成的会话数量及时间不受待测系统及网络状况等的影响,保持了负载的一致性,在这种情况下得到的会话相关测试结果才具有代表性。

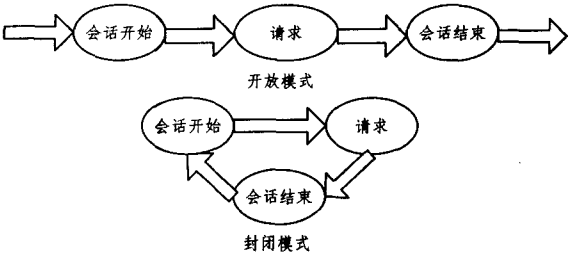

图1 负载仿真的两种模式

TPC-W采用封闭的测试模式,所以无法进行负载规模的动态变化控制。而通过开放模式,则可以实现这种控制。在本文的测试基准中,我们使用负载代理来控制负载规模。通过负载代理实现在测试过程中每个时刻发起会话的控制。每次的并发负载由若干个负载代理来组合生成。

负载代理的控制参数如图2所示。各个参数的具体含义如下:

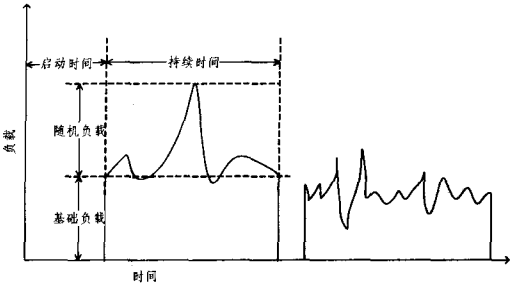

图2 典型的并发负载模型

• 基础负载:表示负载代理在启动时所模拟的并发用户数量。不同于TPC-W中关于基础负载的定义,Bench4Q中基础负载表示的是每秒钟系统发起的会话数,系统发出的会话相互独立。

• 随机负载:随机负载用于在基础负载的基础上增加负载的随机波动性。

• 变化率:表示基础负载每秒钟的变化量。当变化率为正数的时候,基础负载每秒都增加相应数量的并发用户;当变化率为负数的时候,基础负载减少;如果变化率为零,则表示基础负载恒定。

• 启动时间:负载代理的启动时间,该时间是相对于测试

开始时间的偏移值。

• 持续时间：负载代理运行的持续时间。

可以在一次测试过程中定义多个负载代理，通过负载代理的叠加产生复杂的测试场景。通过不同的负载代理组合，可以模拟出多种典型的负载场景[5]，如图 3 所示。

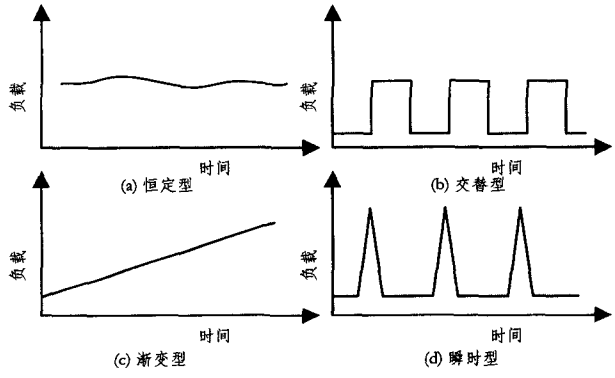

图 3　典型的并发负载场景

恒定型场景：如图 3(a)中所示，负载规模在测试过程中基本维持不变。这是一般测试过程中常用的场景，用于衡量应用服务器在恒定规模负载下的表现。

交替型场景：如图 3(b)中所示，负载规模在低负载和高负载之间交替变化。该测试场景主要用于衡量系统在不同负载压力下系统的适应能力。

渐变型场景：如图 3(c)中所示，负载规模逐渐变化，从较低的负载逐渐变化到一个高负载，或者从一个高负载逐渐减弱到一个低负载。在测试的起始阶段和结束过程中会经常用到该类型场景。

瞬时型场景：如图 3(d)中所示，负载规模平时维持在一个较低的水平，每隔一段时间产生一个瞬时的高负载。该场景用来衡量系统对于突发的大量请求的适应能力。

另外，用户还可以通过定制来实现更符合自己需求的负载类型。具体的方法是：将测试分成不同的阶段，对每个阶段的负载进行建模，然后采用若干个负载代理来模拟。通过对每个负载代理的不同设置来实现压力模型的变化。

在 TPC-W 中，由于封闭的负载生成方式，用户无法对系统在过载情况下的表现做出判断[4]。而在 Bench4Q 中，用户可以通过图 3 中的场景(b)和场景(d)来分别模拟持续的过载和瞬时的过载，通过对系统进行过载的压力测试，可以更好地了解系统的运行情况。

3.2　度量分析

度量分析是测试基准的重要方面。在电子商务领域，一次典型的交易可能包括浏览、搜索、选择、下订单等多个环节。TPC-W 主要关注性能相关指标，使用 WIPS 来衡量系统每秒钟能完成的交互次数。然而，在电子商务企业中，企业的关注点并不是系统能够提供的完成的总操作数，而是系统所能提供的对用户一系列操作连续完成的保障情况，及所能提供的用户体验情况。电子商务中以业务为核心的服务质量需求指标更应该是衡量的重点。在 Bench4Q 中，通过统计测试过程中的会话完成相关情况来体现系统的服务质量。具体指标如下：

• 完成的总会话数：完成的总会话数量体现了系统在测试区间内完成服务的用户数。完成的总会话数量对于电子商务企业来说可能比系统总的吞吐量更有意义。

• 带来商业利益的会话数：这是企业关注的核心。对于能带来商业利益的用户，服务是应该得到优先保证的。

• 失败的会话数及失败原因：失败的会话数是系统未能为用户提供所有服务的数量，通过分析失败会话的原因可能得出系统的瓶颈所在，可以通过对系统的优化来为用户提供更好的服务。

4　Bench4Q 测试工具的设计与实现

基于以上关于 Bench4Q 的描述，我们设计并实现了一个 Bench4Q 工具。Bench4Q 工具的目的是作为 Bench4Q 研究成果的展示，并且为用户提供一个简单易用的工具。用户无需对 Bench4Q 进行深入的了解，只需在图形化的界面上进行简单的配置即可方便地进行测试工作。

目前，Bench4Q 工具发布在国家 863 课题“可信的国家软件资源共享与协同生产环境”成果 Trustie 社区中[6]。同时，Bench4Q 工具也作为一个开源项目发布在国际开源软件组织——开源软件国际联盟(简称 OW2)上[7]。

Bench4Q Tool 的主要设计如图 4 所示。

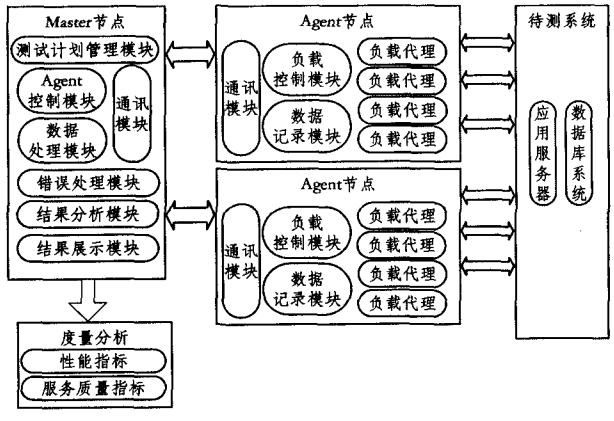

图 4　Bench4Q Tool 架构

系统主要包括待测系统、负载生成器及度量分析器 3 个主要模块。待测系统是一个网上书店的具体实现，用于部署在应用服务器上来模拟典型的电子商务应用；负载生成器按照上节所提出的负载模型来生成以会话为组织的负载压力；度量分析器会在测试过程中记录相关数据，并且在测试结束后搜集整理相关数据。

为了实现大规模并发负载，Bench4Q 设计了一个分布式的负载生成策略。系统使用一个 master 节点来控制多个 agent 节点，可以将 agent 分布在多个不同的客户端电脑上，用于分布式地生成负载。每一个 agent 可以根据配置单独生成负载，agent 节点启动后会定时向 master 节点报告自己的状态信息，并且接受 master 节点的指令进行操作。测试结束后，测试结果统一发送到 master 节点进行统计。

我们使用 Java swing 技术开发了一个图形化的工具，主要功能有 agent 节点的控制、测试计划的配置、测试结果的图形化展示等。系统界面如图 5 所示。通过该界面可以进行测试计划的配置、保存等。也可以通过该界面监控到 agent 节点的状态信息、控制 Agent 节点和测试结果搜集分析展示等。测试结果也可以通过图形化的方式得到展示。

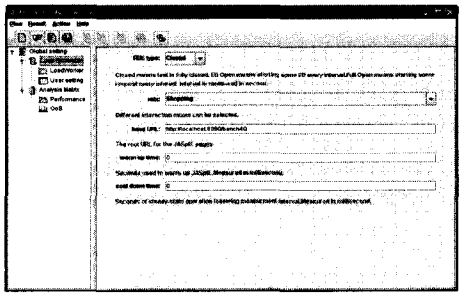

图 5　Bench4Q Tool 基本界面

5　实验

我们使用 Bench4Q 工具对 Tomcat 进行了测试。实验环境包括一台普通客户端 PC、一台应用服务器 PC 和一台数据库服务器 PC，网络连接采用 100Mbps 的以太网交换机。

在实验 1 中，使用封闭和开放两种不同的测试模式。实验结果如图 6 所示。

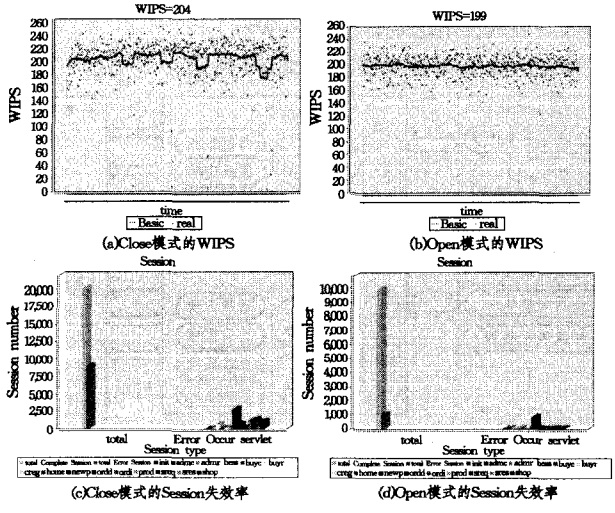

图 6　实验 1

两次测试结果中，WIPS 表现相近，封闭模式 WIPS 为 204，开放模式为 199。但是，在采用封闭的模式时，系统的会话失效率很高，接近 40%。原因如上所述，在过载情况下，可能会出现大量用户请求得不到处理，但模拟的用户会重复发起新请求，得到错误响应，再发起新请求的恶性循环，造成了 WIPS 值的失效。而开放的模式则可以避免这种恶性的循环，会话失效率不到 10%。由此可见，采用封闭的测试模式得到的性能数据是不真实的，不能够反映系统的真实运行情况。

在实验 2 中，采用开放的测试模式进行两次不同的测试。在两次测试中，通过配置不同的配置应用服务器，可以得到不同的 WIPS 值。实验结果如图 7 所示。

从性能角度来看，配置 2 的 WIPS 值达到 199，明显比配置 1 的 WIPS 值(131)要高，说明在配置 2 的情况下，系统能有更大的吞吐量。如果单纯考虑性能，电子商务企业可能会选择配置 2。但是通过对服务质量的考察，会发现在配置 2 下有大量的失效会话，接近 90%。而在配置 1 的情况下，没有失败会话。配置 2 性能的提高是通过损失部分用户的服务质量来得到的。在这种情况下，企业可能得到了更大的吞吐量，其实却不一定能得到更大的利润。从长远来看，可能会影响用户体验，造成用户流失，从而导致更严重的后果。因此，对于电子商务企业来说，仅仅关注系统的性能指标而忽略系统的服务质量保证，是不可取的，必须全方位地对一个系统进行考虑。

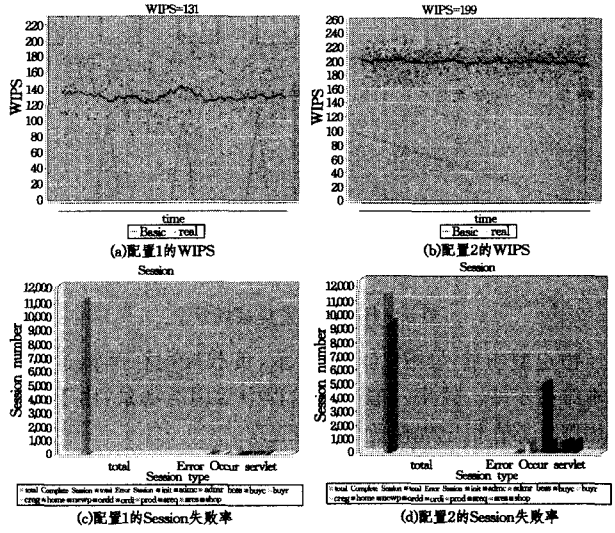

图 7　实验 2

以上两次试验说明，TPC-W 仅关注电子商务系统的性能度量是不够的，而 Bench4Q 通过分析其提供的服务质量可以更准确地分析电子商务系统能否满足业务需求。

结束语　本文提出了一种面向服务质量的电子商务测试基准，其在负载仿真及度量分析等方面对 TPC-W 测试基准进行了扩展，可实现可控的负载仿真以及以会话为中心的度量分析。实验结果表明，与传统的性能测试基准相比，Bench4Q 能够反映电子商务企业更加关注的服务质量特征。

参 考 文 献

[1] Research Z. The Need for Speed II. Zona Market Bulletin, Issue 5[EB/OL]. http://www.keynote.com/downloads/Zona_Need_For_Speed.pdf. 2001

[2] TPC. Transaction Processing Performance Council [EB/OL]. www.tpc.org, 2003

[3] Menascé D A. TPC-W A Benchmark for E-Commerce[J]. IEEE Internet Computing, 2002, 6(3): 83-87

[4] Schroeder B. Web Servers Under Overload-How Scheduling Can Help[J]. ACM Transactions on Internet Technology, 2006, 6(1): 20-52

[5] Andreolini M, Casolari S, Colajanni M. Models and Framework for Supporting Runtime Decisions in Web-based Systems[J]. ACM Transactions on the Web, 2008, 2(3): 17-43

[6] http://www.trustie.net/projects/project/show/Bench4Q

[7] http://forge.ow2.org/projects/jaspte