电子商务软件负载能力评估的实例分析

成　燕[1]　江建慧[2,3]　楼俊钢[2,3]
(1. 上海宏迅软件有限公司, 上海　200030; 2. 同济大学嵌入式系统与服务计算教育部重点实验室, 上海　201804;
3. 同济大学计算机科学与技术系, 上海　201804)

摘　要: 负载能力是表征电子商务软件性能的一项重要标准。负载能力的评价方法包括在线系统的数据统计和估算、程序测算、基于测试的评估方法等。其中,基于测试的评估方法越来越受到人们的重视。本文以一个小型电子商务系统为例,采用基于测试的评估方法进行了系统的负载能力分析。结果表明,对于以优化为目标的系统评估,可以在优化后采用逐步细化的负载测试方式得到最佳负载预测数据。同时,在负载能力的评估过程中,需要根据系统的主要功能来选择合理的评估指标。

关键词: 负载测试; 虚拟用户; 并发用户数
中图分类号: TP311

引　言

随着软件行业的逐步成熟,软件的质量成为越来越多开发人员需要考虑的重点,而软件的性能测试在软件质量保证中起着非常重要的作用。通常,软件发布报告中最重要的问题往往不是系统崩溃,也不是系统功能不全,而是系统性能下降或者处理要求达不到生产能力的要求。值得注意的是,尽管软件功能方面的测试越来越广泛,但对性能测试的重视度还远远不够。

软件性能测试,即通过测试评估一个系统或一个组件是否满足特定的性能需求。性能需求是一种在功能需求之上加以强制条件的需求,例如指定功能执行的速度、准确性,或当时的内存使用情况[1]。系统的瓶颈可能存在于系统的应用层、数据库层、操作系统层、网络层中的任意一层或多层。对于系统软件层,由于多数系统采用较为成熟的商品化软件,业界已提出专业的性能评测系统,但对于应用层的性能评估存在较大的空白。

软件性能的评估包括:负载评估、容量评估、压力评估等方面。负载评估是通过逐步增加系统负载,测试系统性能的变化,并最终确定在满足性能指标的情况下,系统所能承受的最大负载量。容量评估指的是对某些系统存储、传输、统计、查询等业务进行的最大数据量的评估。压力评估用以确定在什么负载条件下,系统性能处于失效状态,并以此来获得系统能提供的最大服务级别的测试。负载评估、容量评估、压力评估在不同的测试目标、不同的测试条件下通过不同的方法取得这些指标数据,并采用不同的评估方式进行评估。可能采取的测试方法是从平均的测试条件(如并发量、数据量)开始负载测试,逐步加大负载;从极端的测试条件开始容量测试和压力测试。

对于一个健康的系统,通常存在两种运行状况:一种是在低负荷下健康运行,另一种是高负荷下健康运行。应用服务器过载是导致系统性能下降的根源[2]。所以我们需要考虑的问题是:当系统负荷提高到何种程度时,系统仍能满足应用健康运行的需求?对于一个希望提供 24 ×365 小时不间断服务的大型电子商务系统,客户对系统的性能需求很高[3]。例如,因特网用户需要的是需求及时得到满足,他们不会长时间地等待页面加载或事物的完成。的确,几秒钟的延时都可能导致客户转向其他网站。本文以一个小型的 B2C 电子商务平台为例,介绍系统负载能力的一种综合评估方法。

1　负载能力测试

1.1　基本方法与测试原理

负载能力的评估的典型方法有在线系统的数据统计和估算(方法一)、程序测算(方法二)、基于测试

收稿日期: 2007-04-30
第一作者: 女,1969 年生,在职硕士生
E-mail: yan.cheng@163.com

的评估(方法三)。这三种方法特点的比较见表 1。

表 1　负载能力评估的三种方法比较

Table 1　Comparison of 3 kinds of load ability evaluation methods

比较项	方法一	方法二	方法三
数据真实性	高	低	中
评估数据的准确性	低	中	高
评估的可实施性	低	高	高
评估结果的公平性	高	低	中

对在线系统的数据统计得到的数据非常真实,但是鉴于系统的多变性及其估算的难度,该方法很难实施。程序测算由系统开发者提供专门的程序进行测试,该类程序通常通过调用系统的 API 来模拟对象的创建等,可以取得较为准确的数据。但由于测试程序由开发者自行提供,同时跳过了最终用户界面,测试数据难免有失公允。基于测试的评估借助于商业负载自动测试工具来模拟最终用户的实际操作进行测试。评估数据的准确性和可实施性都很高,而数据的真实性和评估结果的公平性则依赖于测试脚本对于真实操作模拟的准确性。

图 1 给出了一个针对小型的为 Internet 用户提供签订订单的电子商务系统进行测评的原理图。不同于普通的信息浏览网站只关注于页面的响应时间,签订订单类系统更加关注于订单的平均生成数量,该指标同时与并发用户的数量和订单提交等操作的响应时间有关。

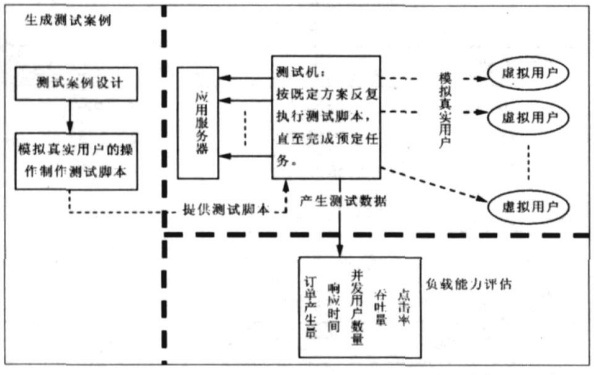

图 1　基于测试的评估原理

Fig. 1　Principle of test-based evaluation

1.2　测试环境配置

对基于因特网的应用系统进行测试需要对客户的环境以及使用网站的方式作一些假设[3]。有多种重要因素影响 Web 应用服务器的性能,包括系统体系结构、服务器的硬件配置、操作系统、Web 服务器软件、数据库性能、网络速度和工作负载等[2]。所以,要求所有负载测试必须在稳定的模型和资源下进行,即保持在相同的硬件环境(包括网络)中进行,同时,操作系统、数据库服务器、J2EE 服务器也全部采用完全相同的版本

服务器端配置: HP RP7410 小型机(2 个 750 MHz CPU/2 GB 内存);操作系统: HP UNIX 11i;数据库: oracle9i、weblogic8.1、应用服务器安装于同一台机器。

客户端配置: PC(奔腾 4 主频 2.8 GHz,2 GB 内存)、操作系统: Windows XP sp2、IE6.0。

测试工具: LoadRunner7.8,安装在客户端上。

网络配置: 局域网 100 Mbps。

1.3　测试基础数据准备

对于一个不断成长的系统,操作系统和数据库的资源利用,是在空间和时间的资源消耗中寻找平衡[4]。所以,在负载测试时,有一个较大的、有效的数据基础非常重要。例如,用户查询、产品查询、创建订单时,对于空表或不同数据量的表的读写操作,系统使用数据库空间和时间的资源必定不同。

以 Oracle 为例,高水标记(High Water Mark, HWM)是一个数据库存储 segment 中已使用和未使用的 block 的分界线,当请求新的空闲块,并且现有空闲列表中的块不能满足要求时,HWM 将移动指向下一个未使用过的块。显然,访问的 block 数量越多,意味着需要消耗更多的资源[4]。

1.4　评估指标

测试得到的数据包括并发用户量、点击率、吞吐量、事务响应时间、订单产生量。根据通常的测试推断,对同一条件下的同一系统,点击率与吞吐量保持成正比,响应时间与并发用户量成反比;在系统稳定运行的情况下,并发用户量与吞吐量成正比,而用户量与订单产生量亦成正比。

2　测试用例生成

负载测试用以测量基于实际的客户行为的系统性能状态。采用测试工具的录制功能创建客户与系统交互的测试脚本,然后采用测试工具模拟并发的功能,模拟多浏览器重复执行该脚本。每个被模拟的浏览器称作一个虚拟用户。需要确保虚拟用户完全仿照真实的用户模式;当响应时间过长时,用户放弃操作。

为了取得公平的评估数据,选取信息量相近,且更加全面地与应用服务器和数据库服务器交互信息的"签订订单"交易作为测试用例,用例中包括用户登录系统、选择货品、填写订单货品信息、送货地址、送货时间、付款方式、签订订单(其中订购2项商品)、查询订单状态、退出系统等步骤。用例的选取综合了系统的主要功能,并涵盖了对主要对象的访问。业界对于负载测试的案例选取都存在不同的侧重点,如文献[2]对一个选课系统选择系统的压力峰值时间段,用户可能进行的所有操作作为测试用例,此类用例更加接近于真实系统。

3 测试方法

测试脚本依照以上过程制作,并且按照用户操作的习惯,在各操作动作之间增加了思考时间。准备以上功能的测试脚本后,首先确定并发用户量、运行时间,按照这个预定的测试场景开始运行,直至时间截止。这样的一轮测试被称作"一次测试"。文献[2]中的测试模拟了系统达到实际压力峰值时的运行状况,此类方式的测试在数据准备阶段需要做较多的调查。本文借鉴了采用程序输入和输出的关系减少测试数量的方法(见文献[5])来进行负载测试。

(1) 以保障系统性能可接受为准则,并结合此类系统的通用需求,初步确定测试采用的并发用户数量为50～200。在负载测试方案中,应考虑用户数增长速度,即"上升速度"问题。用户数量不是一次增到最大值,而是逐渐增加[6]。

(2) 基本要求:测试过程中,每次采用相同的测试环境、基础数据、测试用例、测试时长,得到每次的测试数据。

(3) 第一批测试从50个并发用户开始测试,依照步骤(1)要求得到第一批的第一次测试结果。

(4) 基于可行的方案实施测试,即每次增加50个用户,重复步骤(2),分别得到100、150、200个并发用户的测试结果,第一批测试结束。

(5) 通过比较第一批测试结果,得到第一批负载测试的最佳评估值。

(6) 在第一批测试的最佳值(如130～170个并发用户)附近,再次细化数据,重复步骤(3)～(5),每次增加10个用户进行测试。

(7) 最终得到系统的最佳负载数据。

在每个单次测试中,LoadRunner逐步启动并发用户,各用户对相同的用例进行迭代执行,但每个用户的操作相互孤立,并不完全同步,即一个用户启动后执行用例,第一次迭代结束后,重复执行,直至测试结束。

4 测试结果及分析

4.1 测试结果数据

本系统关注的测试结果数据是并发用户量和订单的平均产生量。本文将以这两项数据为主,先抽取第一批测试数据(并非最佳值)进行对比分析,然后对这一批测试中产生最大平均订单量的一次测试(150并发用户)的数据进行详细的分析。

统计数据表明,各项事务的响应时间与并发用户的增长成正比(见表2),从50～150个用户,服务器接受的平均每秒点击次数和产生的每秒吞吐量没有明显的变化趋势(见表3),所以这个数量级的用户量的增长对于服务器的处理能力没有很大的影响。表4体现了系统最为关注的平均产生的订单数量,这是系统必须保质保量完成的重要任务。

表2 典型事务响应时间

Table 2 The response time of typical transactions

并发用户数量/个	平均响应时间/s		
	登录事务	签订订单事务	登出事务
200	19.654	230.356	0.133
150	3.916	61.608	0.007
100	1.226	41.828	0.003
50	1.15	22.213	0.002

表3 服务器的平均每秒点击次数和吞吐量

Table 3 Average hits per second and throughput of server

并发用户数量/个	平均吞吐量/(字节/s)	平均点击次数/(次/s)
200	135355	8.765
150	391284	24.165
100	369716	22.798
50	464638	28.358

表4 每分钟产生的平均订单数量

Table 4 Average orders per minute

并发用户数量/个	订单数量*/张
200	50.4
150	131.3
100	101.8
50	105.5

*指每分钟的平均数量

考虑150个并发用户所产生的测试数据。测试工具模拟用户的实际操作,每min启动10个用户,

测试启动 15 min 后所有用户都进入操作环境(如图 2 所示)。

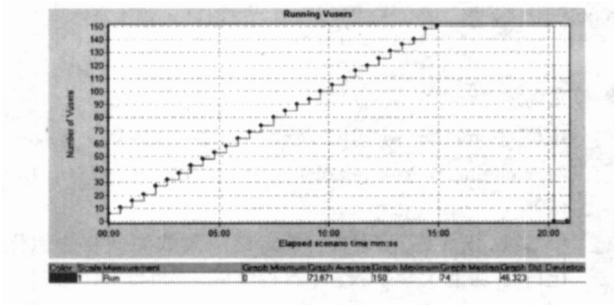

图 2　并发用户数

Fig. 2　The number of concurrent users

从图 3 可以看到,随着测试时间的推移(测试进行了 7～8 min),服务器接受的每秒点击次数达到最高。此时用户量约为 80。大约 11 min 后(此时用户量在 120 左右),点击次数部分下滑,同时基本稳定在一个水平上。

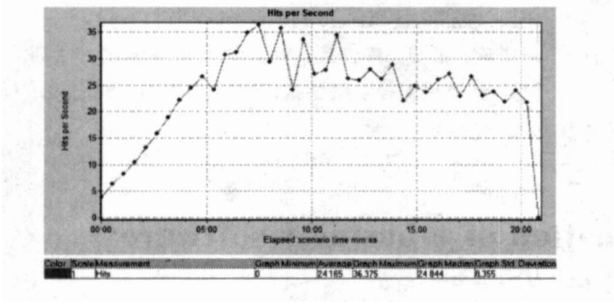

图 3　每秒点击次数

Fig. 3　Hits per second

与每秒点击次数的测试数据的走势接近,测试进行了 7～8 min,服务器的吞吐量达到最高。同样在约 11 min 后,与点击次数走势仍然相近(如图 4 所示)。

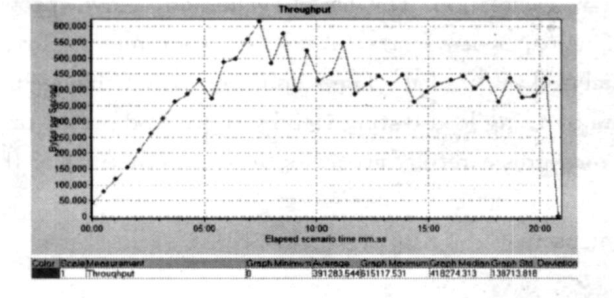

图 4　系统吞吐量

Fig. 4　System throughput

在测试进行了约 11 min 之内,"签订订单"的事务响应时间基本稳定,之后缓慢上升(如图 5 所示)。

图 6 体现了系统产生的订单数量与用户量是密切相关的。当测试进行了约 6 min 后(此时 65 个用户在并发运行),订单量达到最高值,排除服务器的瞬间影响因素,数据基本稳定。

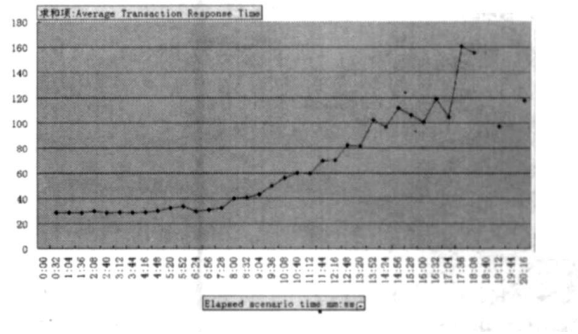

图 5　"签订订单"的事务响应时间

Fig. 5　The response time of transactions

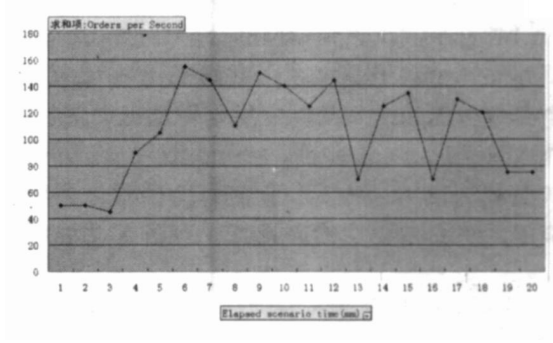

图 6　服务器每分钟产生的订单数

Fig. 6　Orders per minute of the server

4.2　测试数据分析

从图 2～6 可以看到,在第一批测试的最大负载测试中(并发用户 150 个),当用户量达到 80 左右时,点击次数、吞吐量达到最佳值;而当用户量在 65～120 个时,订单的产生量表现出均衡的状态。

从表 4 可以看到,当 50～100 个用户并发访问系统时,系统运行平稳;到 150 个并发用户时,产生的平均订单量达到最大;当继续增加到 200 个并发用户时,系统性能急剧下滑。所以依照第一批测试结果分析,该系统的最佳负载是 150 个并发用户,平均每 min 产生 131.3 张订单。依照以上同样的分析方式,测试 130～170 个并发用户的情况,采用每次增加 10 个用户进行第二批测试,最终得到当并发用户达到 160 时,平均每 min 产生 137.5 张订单。

4.3　其他测试结果

采用其他测试方法,按照选取系统压力峰值情况进行测试,例如选取 200 个并发用户进行测试,将得到这一组单纯的数据;按照随机选取负载的方式进行测试,可以随机选取并发用户数为 100 和 200

这两组数据。

对于选取压力峰值情况得到的数据,显然不能客观地反映系统的最佳负载能力。如果与另一个近似的系统进行比较,由于硬件资源的限制,它们在200个并发用户时得到的负载数据可能类似,但是两者的最佳负载数据可能差异很大。而随机选取负载的方式,可能抽取最差和中等质量的数据,容易低估系统负载能力。

5 结论

对于以优化为目标的系统评估,可以在优化后采用逐步细化的负载测试方式得到最佳负载预测数据,然后看最佳负载数据是否有所提高。在系统调优过程中,重复"负载测试—负载能力评估—系统调优"过程,直至测试达到期望的性能标准。而测试者需要建立一个对于系统在正常情况下的行为基线,这个基线可以在回归测试中用来衡量一个新版本的性能改善程度。

负载测试结果很难仅仅解释为通过或失败,需要根据系统的主要功能来选择合理的评估指标,如一个偏重信息查询的系统更注重于事务的响应时间,而一个偏重订单签订的系统则需要更加关注订单的生产效率。

参考文献:

[1] IEEE Computer society. IEEE standard glossary of software engineering terminology[S]//IEEE Std. 610, 12, 1990.

[2] 姚海波,张保卫,张毅坤. Web站点负载测试与性能优化[J]. 西北大学学报:自然科学版, 2004, 34(增刊): 158-161.

[3] MYERS GJ. 软件测试的艺术[M]. 2版. 王峰, 陈杰,译. 北京: 机械工业出版社, 2006.

[4] 盖国强,冯春培,叶梁,等. Oracle数据库性能优化[M]. 北京: 人民邮电出版社, 2005.

[5] SCHROEDER PJ, KOREL B. Black-box test reduction using input-output analysis[C]//Proc of the International Symposium on Software Testing and Analysis. Oregon, IEEE Computer Society Press. 2000: 173-177.

[6] 卢庆玲, 张威, 宫云战. Web站点负载测试方法研究[J]. 装甲兵工程学院学报, 2003, 17(2): 57-59.

Case study of load ability evaluation of e-business software

CHENG Yan[1] JIANG JianHui[2,3] LOU JunGang[2,3]

(1. Shanghai BMI Asia Software Company Ltd, Shanghai 200030; 2. Key Laboratory of Embedded Systems and Service Computing, Ministry of Education of the People's Republic of China, Tongji University, Shanghai 201804; 3. Department of Computer Science and Technology, Tongji University, Shanghai 201804, China)

Abstract: The loading capacity is an important measure of e-business software. Three typical load-testing methods are data statistic and estimation of online system, program calculation, test-based evaluation. Now, people pay more and more attention to the test-based evaluation. This paper demonstrates this evaluation method by an example a small e-business system, and its load ability is analyzed in detailed. The results show that the evaluation aiming to optimization can get the best forecast load data by using load testing step by step. And on the process of load ability evaluation, we should choose reasonable measures carefully according to main functions of the under-evaluation-system.

Key words: load testing; virtual user; number of concurrent users