一种有效的 Web 性能测试方法及其应用*

赫建营，晏海华，刘　超，金茂忠
（北京航空航天大学 软件工程研究所，北京 100083）

摘　要：针对 Web 应用软件的特征，提出了一种基于目标的性能测试方法，其关注的主要内容包括与 Web 应用相关的负载测试和压力测试两个方面。不但对这两个方面的测试方法进行了全面的分析和探讨，还强调了测试过程管理的重要作用，最后给出了这种方法在 Web 应用性能测试实践中的一个具体应用。

关键词：性能测试；负载测试；压力测试；软件测试
中图法分类号：TP301　　**文献标识码**：A　　**文章编号**：1001-3695(2007)01-0275-03

Effective Web Performance Testing Method and Its Application

HE Jian-ying, YAN Hai-hua, LIU Chao, JIN Mao-zhong
(*Software Engineering Institute, Beihang University, Beijing* 100083, *China*)

Abstract: This paper presents an objective-based performance testing method toward Web-base software applications. Major concentrations of this research are the load testing and stress testing. The paper not only analyses the way to perform load testing and stress testing in detail, but also provides an idea that the software testing process management should also be strongly emphasized. Finally, it gives a case study which utilizes the method proposed in this paper.
Key words: Performance Testing; Load Testing; Stress Testing; Software Testing

目前，随着电子商务和电子政务等 Web 应用的兴起，基于 B/S 结构的软件日益强劲发展，正在成为未来软件模式的趋势。然而，当一个 Web 应用被开发并展现在用户、供应商或合作伙伴的面前时，尤其是即将被部署到实际运行环境之前，用户往往会疑问：这套 Web 应用能否承受大量并发用户的同时访问？系统对用户的请求响应情况如何？在长时间的使用下系统是否运行稳定？系统的整体性能状况如何？如果存在性能瓶颈，那么是什么约束了系统的性能？而这些正是 Web 性能测试解决的问题，如何有效进行 Web 性能测试，目前并没有一个系统和完整的回答。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，即使进行了测试，其关注点也主要放在功能测试上。但是，近年来 Web 性能测试越来越引起重视，成为 Web 系统必不可少的重要测试内容。本文的研究就是基于这种需求，从已进行过的 Web 性能测试实践中总结一套基于目标的 Web 性能测试方法，该方法已在大量的软件测试项目实践中被证明是有效的和可操作的。其具体测试实施方面包括负载测试和压力测试。

1　基于目标的 Web 性能测试方法

1.1　基本概念

一般来说，性能测试包括负载测试和压力测试两个方面，它们是同一过程的两个阶段。本文将这三个概念区别如下：

定义 1　性能测试。为了获取软件处理事务的速度及确定系统性能的瓶颈而进行的测试：①为了检验性能是否符合需求；②为了得到某些性能数据供人们参考[1,2]。

定义 2　负载测试。性能测试的一个阶段，是为了确定在各种级别负载系统的性能而进行的测试。目标是测试当负载逐渐增加时，系统组成部分的相应输出项，如事务处理成功率、事务处理速度、响应时间、CPU 负载、内存使用等来决定系统的性能[1,2]。

定义 3　压力测试。性能测试的一个阶段，是为了确定一个系统的瓶颈或者所能承受的极限性能点而进行的测试，从而获得系统能提供的最大服务级别的测试[5]。

一般来说先进行系统的负载测试，然后确认出各种极限参数后，在此基础上进行压力测试。

1.2　Web 性能测试的目标

Web 性能测试的目标是什么？它不仅是用测试工具去运行一些测试脚本来证明产品是否可以达到性能指标，更关键是要发现产品性能上的缺陷，并解决定位问题，这才是软件性能测试的真正目的。其目标可分为以下两种级别：

(1) 性能测试总体目标。找出 Web 应用系统可能存在的性能瓶颈或者软件缺陷，确认其是否可以达到用户的使用需求。收集测试结果并分析产生缺陷原因，提交总结报告，让软件开发方对 Web 应用进行性能改进。

(2) 性能测试具体目标。此目标又可以分为：①确定 Web 应用系统的总体性能参数，包括所支持的最大并发用户数、事务处理成功率、请求相应的往返延迟等，如表 1 所示；②确定在各个级别的负载及压力测试下服务器输出的具体性能参数，如表 2 所示。

这些测试目标驱动了整个测试过程的进行，因而在 Web 性能测试中起着至为关键的核心作用，因此在软件性能测试之

收稿日期：2005-08-04；修返日期：2005-11-11
基金项目：国家‘863’计划资助项目(2001AA113100)

前一定要有一份《软件性能测试需求规格说明书》,用于定义详细的测试目标,并在表 1、表 2 的空格中输入预期的性能指标,这是检查软件性能是否符合要求的基本依据。

表 1 Web 性能测试总体指标

指标	最大并发用户数	事务处理成功率	请求响应的往返延迟	稳定运行的时间长度
具体数字				

表 2 Web 性能测试基本输出参数

时间指标 / 容量指标	最小响应时间(s)	最大响应时间(s)	90% 响应时间(s)	事务处理速度(个/min)	平均服务器响应时间(s)	CPU 占用率(%)	内存使用率(%)
用户数目							

1.3 基于目标的 Web 性能测试

1.3.1 负载测试方法

负载测试是性能测试中的第一阶段,也是任何 Web 应用开发周期中的一个重要步骤。在构造一个为大量用户服务的 Web 应用,搞清楚最终软件产品所能够承受多大的负载是非常重要的。此外,负载测试还能够暴露出最终会导致服务器崩溃的内存漏洞以及硬件资源的占用情况,通过对测试结果的分析从而决定是否要添置硬件资源和是否要对软件进行整体架构上的变更[3,6]。

负载测试的基本策略就是通过负载测试工具进行脚本的自动录制和回放。因为在实际的测试过程中,要按照实际投入运行的情况,组织成千上万的用户来进行负载测试,无论从哪个方面来看,都是不现实的。一旦发现了问题,不仅需要重复地进行这种耗费巨大的测试,而且问题不容易重现,不能方便地找出性能的瓶颈所在[2]。而使用软件进行自动测试工具就不会存在这种情况。

我们提出一种基于目标的测试方法,其核心思想就是使负载测试的所有执行活动均围绕测试一开始时制定的目标有针对性地进行测试,以减少冗余的测试和无价值的测试。这种方法的基本步骤如下:

(1)确定负载测试目标。在 1.2 节中已经提到过性能测试目标的确定,在此需要强调的是:一个 Web 应用的业务逻辑可能有成百上千种。对每一种业务逻辑进行测试是不现实的也是没有必要的。因此在制定性能测试的目标时一定要选取 Web 应用的核心业务操作,并给出其预期需要达到的指标。事实上,无论 Web 应用怎么千变万化,其基本的业务逻辑无外乎如表 3 所给出的种类。

表 3 Web 应用的基本业务逻辑

业务逻辑	表单操作	页面浏览	上载文件	下载文件	首页登录	特殊业务操作(如生成 Word 文档)
性能指标						

(2)使用负载测试工具录制测试脚本,并编写测试用例。①对于绝大部分 Web 应用来说,负载测试的快速、有效进行还是要求诸于负载测试工具,手工脚本在有些 Web 应用中是有用的,如邮件系统的性能测试,但是同样需要自动脚本的辅助。②目前常用的负载测试工具有:LoadRunner,Pureload,Webload,QAload 等;③测试用例的编写可以 1.2 节中定义出的测试目标为基本划分,每个测试目标编写一份测试用例。脚本可以作为测试用例的一部分附在每份用例后面,这样使得测试更具条理性、针对性和有序性。

(3)执行测试用例,同时使用工具记录在脚本执行过程中 Web 应用服务器输出的各项参数:①通过在一台或几台机器上运行测试脚本,给 Web 应用服务器加载各个级别的负载,模拟成百上千的虚拟用户同时向 Web 服务器发送业务请求。同时记录表 1、表 2 中所列出的各项性能参数。②一定要使用如 Sniffer 等网络监测工具来记录 Web 服务器输出的各项性能参数,很难想象使用秒表记录输出参数会具备任何实际参考价值。③负载的加载应该遵循预热、加压、维持、下降这样一个循序渐进而又逐步下降的阶段性过程,尽量模拟用户的实际使用情况,使得测出的系统输出参数更能真实反映 Web 应用的实际性能。

(4)在测试的执行过程中,若发生系统的异常,即系统崩溃、内存溢出、事务请求无响应和大量链接失效等,则创建软件问题报告,进行软件性能缺陷跟踪。

(5)总结测试结果并分析 Web 应用的性能。对性能测试输出参数进行统计分析和数据挖掘,将发现的问题和错误进行分类。从定性和定量两个方面获取对 Web 应用性能的深刻认识,如数据分布是否合适、负载是否平衡、是否因为数据库连接池设置的限制导致 Web 应用的最大并发用户数上不去、缺陷的产生是否与编程人员的水平和习惯有关、测试过程中是否存在 CPU、内存等资源未被释放造成性能越来越低等,以确定系统的稳定性和优化性能。

1.3.2 压力测试方法

压力测试是性能测试的第二阶段。目标是通过确定一个系统的瓶颈或不能接收的性能点,来获得系统能提供的最大服务级别。了解 Web 应用的这些极限状态是很有价值的,如 Web 应用系统所支持的最大并发用户,在最大并发用户访问的情况下系统能够稳定运行的时间等。此外,在 Web 应用系统中,压力测试是一个必经阶段,只有经过压力测试,达到理想效果,才能经受得起未来正式运行中大量业务、数据处理的考验,如果达不到业务需求的处理能力,必须对系统改进、完善[5]。

压力测试可以被看作是一种极限情况下的负载测试,因此压力测试的进行可以基本上遵循 1.3.1 节中阐述的方法。但是压力测试又有自己的特性,因而它的测试方法与负载测试稍稍不同。压力测试与负载测试不同之处在于两点。

(1)关注的重点不同:①负载测试关注的是在各个级别的用户请求下系统的运行情况及输出参数,其目的是为了模拟用户的实际使用情况,通过系统输出的参数来观察和分析 Web 应用的实际性能。它强调的是确认系统满足了用户正常使用的需求。②压力测试则是一种破坏性测试,给 Web 应用加载远远超过用户实际使用情况的负载。观察系统在此情况下的反应。它强调的是系统在远远超过用户正常使用需求下,系统是否会出现崩溃、内存溢出。

(2)负载测试通常只是加载单一类型的压力(如用户数、页面请求等),而压力测试则是综合的、全面的。其来源基本上可以分为:①时间的压力,如向 Web 应用持续发送请求一个月;②容量的压力,如并发用户的数目、每个用户发出请求的数目;③时间与容量相结合使用造成的综合压力,如1 000个并发用户在一秒钟内同时投递大量的业务请求。

压力测试与负载测试的这些不同，使得它的测试方法除了1.3.1节所述外还应注意以下两点：

(1)预先估算压力测试强度为①在确定测试目标之前应先估算 Web 应用核心业务的压力范围，并以此为根据确定测试用例。②对最大并发用户的压力规模估算应该采用发展的眼光，如某 Web 邮件系统有3 000用户(根据经验数值其最大并发用户在30 人左右)。目前企业正在扩大规模，两年后可能有用户20 000(根据经验数值其最大并发用户在 200 人左右)，那么在进行压力测试时应该给其加载其最大并发用户数应该是 200 个而不是 30 个。③核心业务的压力强度的估算一般采用 80－20 原则，如每个工作日中 80% 的业务在 20% 的时间内完成，亦即每天 80% 的业务在 1.6 小时内完成。④压力测试的总体强度计算方法为 Stress = MaxUser × TotalRequest × 80% / (TotalTime × 20%)。

(2)进行多用例、多场景的综合测试为①在几台测试机上分别运行几个测试脚本，每个脚本模拟若干个并发用户向 Web 服务器请求一种核心业务操作。②循环执行测试脚本，每过一段时间记录 Web 服务器的性能输出参数，并最后记录发出的事务请求总数和成功完成的交易总数。③测试时间长则 1～2 星期，甚至一个月，短则至少也要持续运行脚本 48 小时。

1.4　测试的实施及测试过程管理

高效而有序的软件测试过程管理是软件测试项目实施成功的重要保障，Web 性能测试同样也不例外。那么如何有效地实施 Web 性能测试并对测试过程进行跟踪和有效控制呢？首先取决于人，人是最有价值和最重要的资源，没有一个积极的、团队合作的、高效的测试小组，测试工作就不可能圆满实现[6]。其次依赖于项目管理人员对整个测试流程的有效组织和控制，这就需要项目管理人员需要广泛的技术经验、高度的责任感、极大的耐心和洞察力等。最后需要一个功能齐全、简单易用的测试过程管理工具。

Web 性能测试的一般流程如图 1 所示。一个成功的 Web 性能测试项目管理人员应该很好地完成以下职责：

(1)监督整个测试流程如期完成，检查项目成员测试任务按时完成；

(2)与 Web 应用开发方及委托测试单位进行协调和沟通；

(3)撰写测试相关文档。

此外，测试过程管理工具的有效使用也对整个项目的成功至关重要，很难想象使用纸张或 Excel，Word 文档来管理整个性能测试过程能够有效保证此性能测试的质量。就测试管理工具而言，目前有北京航空航天软件工程研究所的 QESuite 和 Mercury Interactive 公司的 TestDirector 等。

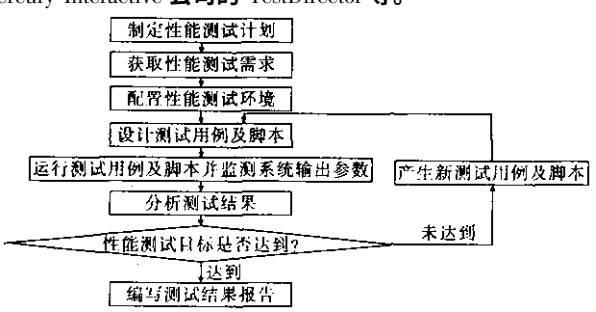

图 1　Web 性能测试的基本流程

2　Web 性能测试方法的具体应用

下面以某电子邮件系统的性能测试为例来简要说明基于目标的 Web 性能测试方法的具体应用。

(1)确定性能测试目标：①总体目标。验证和确认此邮件系统的性能(收发邮件成功率、平均往返延迟、最大并发用户等)能够满足当前3 000用户规模的使用需求，并判断其能否满足未来20 000用户规模的使用需求。对测试结果进行收集并总结，分析系统的性能瓶颈并给出原因。②具体性能测试目标如表 4 和表 5 所示。

表 4　某邮件系统性能测试总体指标

业务逻辑	收发邮件成功率(%)	最大并发用户数	收发邮件平均往返延迟(ms)	用户接收邮件的时间(s)
性能指标	≥99	≥100	≤250	≤6

表 5　某邮件系统性能测试 Web 服务器输出参数指标

时间指标 / 容量指标	最小响应时间(s)	最大响应时间(s)	90% 响应时间(s)	事务处理速度(个/min)	平均服务器响应时间(s)	CPU 占用率(%)	内存使用率(%)
收发邮件数目(M)	≤1	≤1.5	≤1.05	≥800	≤1.25	≤70	≤15
收发邮件大小(N)	≤1	≤1.5	≤1.05	≥800	≤1.25	≤70	≤75
用户数目(T)	≤1	≤1.5	≤1.05	≥800	≤1.25	≤70	≤75

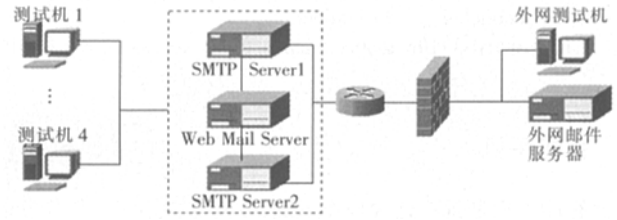

图 2　某邮件系统测试环境配置图

(2)测试方法：①使用开源的性能测试工具 Postal，并辅以手工编写的 Perl 测试脚本；②在内网和外网各自搭建邮件服务器，其测试环境配置如图 2 所示；③压力测试使用三台测试机，每台测试机模拟 50 个并发用户，每个用户发送 40 封 100KB 的邮件。每 20 分钟循环执行一次测试脚本，持续此测试场景 48 小时。

(3)性能测试结果：①系统所支持的最大并发用户为 25 个，主邮件服务器 CPU 最大占用率 100%，平均占用率 74.6%，内存占用率平均 51%；实际邮件发送总数为 219 610 封；实际收到邮件 217 564 封，收发邮件成功率为 99.07%；在测试的 48 小时内系统运行稳定；②性能测试结论；③此邮件系统的系统性能能够较好的满足当前规模用户(3 000 个)的使用要求；④若要支持规模未来 20 000 的用户使用要求，尚需进行软件系统的改进和硬件资源的添置。

3　结束语

Web 应用的多样性决定了 Web 性能测试方法的多样性，但无论其表现，所有的测试活动都一定有其目的性，也只有让性能测试围绕其目标展开才能使投入的测试资源产生最大的效益。本文围绕所提出的基于目标的性能测试而展开，同时强调了测试管理的作用，并给出了一个邮件系统测试的实例。本文提出的性能测试方法已经在许多项目实践中

(下转第 285 页)

（延迟变化量）变化曲线。Jitter 的计算公式如下所示[8]：

$$J=\frac{[\text{rcvtime}(j)-\text{sndtime}(j)]-[\text{rcvtime}(i)-\text{sndtime}(i)]}{j-i},\ j>i \quad (8)$$

其中 i，j 分别是实时流的唯一标志序号。比较两图可以发现，CRFCC 流由于不需要差错重传，比 TCP 流更连续，这也是图 1 中 CRFCC 流变化幅度小于 TCP 的原因。同时，CRFCC 流的 Jitter 多数在 20ms 之间，小于 TCP 流的 50ms，因此 CRFCC 的延迟变化量小于同时竞争的 TCP。

设 m，n 分别是 CRFCC（或 TFRC）流和 TCP 流的数目，m 个 CRFCC（或 TFRC）流的吞吐量分别是 $TH_1^C, TH_2^C, \cdots, TH_m^C$，$n$ 个 TCP 流的吞吐量分别是 $TH_1^T, TH_2^T, \cdots, TH_n^T$，那么 CRFCC（或 TFRC）流与 TCP 流的平均吞吐量为[9]

$$TH_C=\frac{1}{m}\sum_{i=1}^{m}TH_i^C \quad TH_T=\frac{1}{n}\sum_{i=1}^{n}TH_i^T \quad (9)$$

定义 CRFCC（或 TFRC）流与 TCP 流的等价比为

$$ER=TH_C/TH_T \quad (10)$$

两种协议的等价比越接近 1，说明两者公平性越好。在相同的网络环境下，将 TFRC 流和 TCP 流进行仿真试验，通过式（9）和式（10）得到 CRFCC 流对 TCP 流的等价比以及 TFRC 流对 TCP 流的等价比，如图 5 所示。从图 5 不难发现，绝大多数时间内，CRFCC 对 TCP 是公平的、友好的，而且 CRFCC 对 TCP 的公平性好于 TFRC。此外，我们还统计了 180s 内 CRFCC 流和 TFRC 流的丢包率，CRFCC 流的丢包率为 3.18%，TFRC 流为 10.18%，结果表明 CRFCC 对丢包控制更有效。

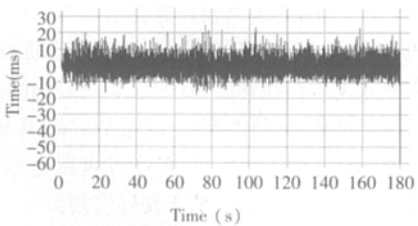

图 3　CRFCC 流的 Jitter 变化

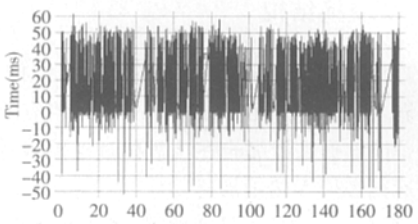

图 4　TCP 流的 Jitter 变化

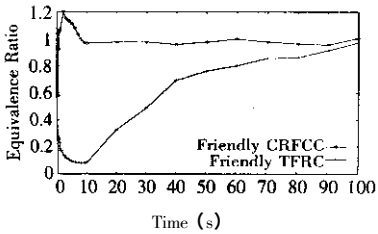

图 5　不同流对 TCP 的等价比变化

4　总结

本文提出一种基于模型的实时流单播拥塞控制算法 CRFCC。在拥塞出现后的速率恢复过程中，算法将前次拥塞时速率反馈值引入该阶段发送速率的调整，使得算法除了具有 TCP 友好性外，还有较好稳定性和较低的丢包率。该算法如何扩展到组播上，还需要作进一步的研究。

参考文献：

[1] Jacobson V. Congestion Avoidance and Control[J]. ACM SIGCOMM Computer Communication Review, 1988, 18(4): 314-329.

[2] Floyd S, *et al*. Equation-based Congestion Control for Unicast Application[EB/OL]. http://www.icir.org/tfrc/tcp-friendly.pdf.

[3] Floyd S, Padhye J, Windmer J. TCP Friendly Rate Control[EB/OL]. http://www.ietf.org/rfc/rfc3448.txt?number=3448.

[4] Floyd S, Fall K. Promoting the Use of End-to-End Congestion Control in the Internet[J]. IEEE/ACM Transactions on Networking, 1999, 7(4): 458.

[5] Padhye J, Firoiu V, Towsley D, *et al*. Modeling TCP Throughput: A Simple Model and Its Empirical Validation[J]. IEEE/ACM Transactions on Networking, 2000, 8(2): 133-145.

[6] http://www.isi.edu/nsnam/ns/[EB/OL].

[7] Schulzrinne H, Casner S. RTP: A Transport Protocol for Real-time Applications[EB/OL]. http://www.ietf.org/rfc/rfc1889.txt?number=1889

[8] http://140.116.72.80/~smallko/ns2/tool.htm[EB/OL].

[9] 董振亚，彭宇行. 一种基于模型的实时媒体流拥塞控制机制[J]. 计算机工程与科学，2004，26(4): 89-90.

作者简介：

谌洪初（1981-），男，硕士研究生，主要研究方向为计算机网络和多媒体通信技术；汪庆（1969-），男，副教授，博士，主要研究方向为图像处理和多媒体信息处理。

（上接第 277 页）得到了良好的应用，但 Web 性能测试模型的总结和深入研究还有待进一步进行，这也是我们下一步的主要工作计划。

参考文献：

[1] Alberto Avritzer, Elaine J Weyuker. The Role of Modeling in the Performance Testing of E-commerce Applications[J]. IEEE Transactions on Software Engineering, 2004, 30(12): 1072-1083.

[2] Subraya B M, Subrahmanya S V. Object Driven Performance Testing of Web Applications[C]. Quality Software. Proceedings of the 1st Asia-Pacific Conference, 2000. 17-26.

[3] 中国软件评测中心测试中心，性能：软件测试的重中之重[J]. 中国计算机用户，2003，31: 42-43.

[4] 林锐. 软件测试—掌握有效测试软件的方法与技术[EB/OL]. www.8848software.com/scmchina/doc/softtest.ppt，2002-11.

[5] 叶新铭，冯晓利. 软件压力测试流程[J]. 内蒙古大学学报（自然科学版），2002，33(1): 107-108.

[6] 高国柱，李红. 银行计算机综合业务系统性能测试与分析[J]. 系统设计技术，2002，(7): 63-64.

作者简介：

赫建营（1980-），男，博士研究生，主要研究方向为软件测试技术、软件工程和知识管理；晏海华（1964-），男，副教授，主要研究方向为软件工程、软件测试技术和面向对象技术；刘超（1958-），男，教授，主要研究方向为软件工程、软件测试技术和面向对象技术；金茂忠（1941-），男，教授，博导，主要研究方向为软件工程和编译技术。