一种模拟驱动的 Web 应用程序性能测试方法

梁 晟[1] 李明树[1] 梁金能[2] 陈振冲[2]
[1](中国科学院软件研究所 北京 100080)
[2](香港理工大学电子计算学系 香港九龙)
(ls@intec.iscas.ac.cn)

摘 要 性能是 Web 应用程序成功的要素之一,性能测试则是保证这一要素的重要手段.但由于 Internet 及 Web 用户的不确定性,Web 应用程序的性能测试难于传统 Client/Server 的测试.比较了 3 种主要的 Web 性能测试方法;提出了一种简单可行的、通用的方法——模拟驱动的自动负载测试方法.关键的步骤有:根据系统使用方式和客户端各种特征的分布信息来确定测试负载、设计测试用例;利用测试工具开发相应的测试脚本;运行测试用例模拟不同类型用户的典型行为;收集被测程序的性能数据.结合实例详述了该方法,并给出了测试计划的模板

关键词 Web 应用程序;性能测试;负载测试;负载;测试脚本;测试场景
中图法分类号 TP302

A Simulation-Driven Performance Testing Method for Web Applications

LIANG Sheng[1], LI Ming-Shu[1], LIANG Kam-Nang[2], and CHEN Chun-Chung[2]
[1](*Institute of Software, Chinese Academy of Sciences, Beijing* 100080)
[2](*Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong*)

Abstract Performance is a crucial factor to the success of a Web application, and performance testing is an important way of ensuring this quality. However, due to many uncertainties of Internet and Web users, performance testing of Web applications is more difficult than traditional client/server testing. Three performance testing methods for Web are compared. A practical, simulation-driven automated load testing method is presented with an example. The key steps are: ① identify the testing workload and design test cases according to system usage patterns and different client characteristics, ② use suitable tools to develop test scripts, ③ test the application with predefined workload to simulate the behaviors of real users, and ④ collect performance data. A template of the performance testing plan is also provided.

Key words Web application; performance testing; load testing; workload; test script; test scenario

1 引 言

性能是 Web 应用程序成功的一个重要因素,对于用户来说,性能有时比功能更重要.以下是 Zona 一份研究报告[1]的数据统计:页面的下载时间减少 1s,用户的放弃率从 30%下降到 6%~8%;由于性能问题,超过 34%的用户没有从最初访问的网站购买商品,而其中的 21%后来从别的网站购买了商品.据估计,性能问题造成全球商务网站每年损失 43.5 亿美元,占总损失的 15%[2].

因此,要保证 Web 应用程序达到预期的性能,开发过程中必须进行性能测试.只有通过性能测试后才有信心将它投入使用[3].性能测试的目标在于

收稿日期:2002-02-04;修回日期:2003-01-17
基金项目:国家自然科学基金(60273026);国家"八六三"高技术研究发展计划基金(2001AA113191)

通过模拟真实负载，找出性能瓶颈，优化系统性能，从而保证程序在实际运行中提供良好和可靠的性能 然而，由于 Internet 和 Web 用户的不确定性，Web 的性能测试存在以下难点：

（1）负载的不可预知

负载是指要求一个系统处理的过程量和信息量[4]. 在性能测试中，确定负载是一个关键步骤，它决定了测试结果的准确性和可靠性 传统 Client/Server 应用是基于 LAN/WAN 环境，用户群是相对熟悉、可预知的，他们的访问时间和访问方式也是相对可预知的 而 Web 应用是基于开放的 Internet 环境，用户群是高度不可预知、分布广泛、类型各异的，他们的访问时间和访问方式也是不可预知的 因此，确定 Web 应用程序的负载是一项具有挑战性的工作

（2）测试场景设计的困难

测试场景是指性能测试的测试用例，它的可执行形式是测试脚本 测试场景的设计不当会造成测试结果的偏差 例如，假设为每个 Web 事务生成一个脚本，如果简单地设定每个脚本都执行同样长的时间段，那么执行快的事务将会执行得比较频繁，使得所测的吞吐量结果偏大；如果设定每个事务都反复执行一定的次数，那么短脚本执行结束的时候，长脚本还将运行下去，这会导致服务器的负载不均衡，从而歪曲吞吐量结果[5]. 因此必须仔细地设计测试场景以避免这些情况的发生

（3）测试环境和真实环境的差异

Web 应用的运行环境是开放的 Internet，而测试通常是基于 LAN 环境，Internet 上的很多因素往往被忽略了，例如某些公共网络骨干上的交通瓶颈 在这种情况下，测试的结果会与实际用户的体验不同 因此测试时应尽可能使测试环境逼近真实环境，例如，要考虑测试环境中是否要模拟不同的操作系统、不同的浏览器和体系结构以及不同的 IP 地址等等

2 相关工作

针对不同的应用类型和侧重点，目前主要有 3 种 Web 性能测试方法：虚拟用户方法[6,7]、WUS 方法[8]和对象驱动方法[9]. 它们的比较如表 1 所示：

表 1 各种 Web 性能测试方法的比较

比较项	虚拟用户方法	WUS 方法	对象驱动方法
负载的基本单位	真实用户的商务处理	经常被访问的路径	页面组件对象的行为
脚本的形式	（商务处理 1, 2, …, n）	（被访页面 1, 2, …, n）	（对象行为 1, 2, …, n）
获取负载信息的途径	人工收集和分析	挖掘日志文件	—
适合的程序类型	电子商务应用程序	有已发布版本的程序	页面组件类型多的程序
优点	方法直观，有工具支持	科学、精确地确定负载	结构化的测试方法
缺点	负载靠人工收集	负载的确定依赖日志	强调局部性能

（1）虚拟用户方法

通过模拟真实用户的行为来对被测程序（application under test, AUT）施加负载，以测量 AUT 的性能指标值，如事务的响应时间、服务器的吞吐量等. 它以真实用户的“商务处理”（用户为完成一个商业业务而执行的一系列操作）作为负载的基本组成单位，用“虚拟用户”（模拟用户行为的测试脚本）来模拟真实用户. 负载需求如并发虚拟用户数、商务处理的执行频率等通过人工收集和分析系统使用信息来获得. 一些负载测试工具支持该方法，可用较少的硬件资源模拟成千上万个虚拟用户同时访问 AUT，并可模拟来自不同 IP 地址、不同浏览器类型以及不同网络连接方式的请求，同时可实时监视系统性能指标，帮助测试人员分析测试结果

该方法有成熟的工具支持，比较直观，适合电子商务应用程序的测试；但确定负载的信息要依靠人工收集，准确性不高

（2）WUS 方法

基于“网站使用签名（website usage signature, WUS）”的概念来设计测试场景，强调建立真实的负载 WUS 的提出是为了衡量测试负载和真实负载之间的接近程度，它是一系列能全面刻画负载的参数和测量指标的集合，包括每小时浏览的页面数/点击数、平均访问持续时间、每次访问平均浏览的页面数/点击数以及页面请求分布等. 这些参数值可以从日志文件中得到. 同时也包括一些影响负载的客户端变量，如用户对网站的熟悉程度、对延迟的忍耐程度和客户端连接速度等. 经常被访问的路径作为负载的组成单位. 该方法认为，只有当测试中的 WUS 和实际应用中的 WUS 基本相符时，测试才是

有效的

该方法的优点是测试负载来源于网站实际的运行数据, 因此能反映和代表实际的负载情况　缺点是太依赖于日志文件, 不适用于测试新开发的程序.

(3) 对象驱动方法

基本思想是将 AUT 的行为分解成可测试的对象　对象可以是链接、命令按钮、列表框、消息、图像、可下载的文件、音频等　对象定义的粒度取决于应用程序的复杂性　一个 Web 页可以用对象来递归定义, 性能测试的过程也就变为测试每个对象或某些对象的集合, 这些对象的行为作为负载的组成单位.

该方法通过将 AUT 分解为对象, 使测试结构化程度高、可重用性好、结果清晰、适合页面组件类型较丰富、业务复杂的应用程序; 但过于强调局部组件的性能难以反映用户对性能的实际感受.

3　模拟驱动的自动负载测试方法

针对 Web 性能测试中潜在的问题和困难, 在吸收已有方法优点的基础上, 我们全面地考虑了影响负载的各种因素, 提出了一种简单可行的、通用的性能测试方法——模拟驱动的自动负载测试方法　其主要思想是: 根据系统使用方式和客户端各种特征的分布信息来确定负载、设计测试用例, 利用合适的工具开发相应的测试脚本, 来模拟不同类型用户的典型行为, 以测量 AUT 的性能　该方法分为以下阶段: 确定性能需求、编制测试计划、设计测试场景、设置测试环境、开发测试脚本、运行测试场景、分析结果和优化系统以及测试总结　该过程如图 1 所示.

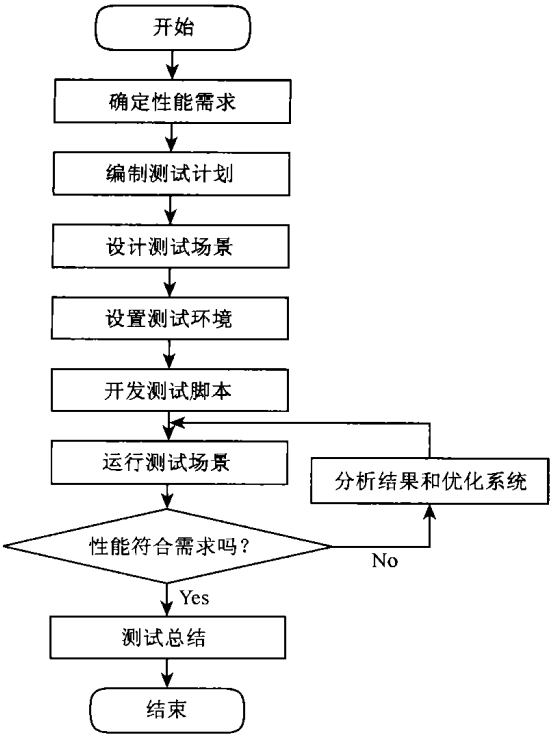

图 1　模拟驱动的自动负载测试流程

为了详细说明该测试方法和过程, 我们结合一个实例——网上书店来加以说明　假设该应用程序有这样一些事务: 注册新用户、登录、浏览、搜索、放入购物车、修改数量和结账.

3.1　确定性能需求

测试之前需要确定性能需求　通常性能需求包括并发用户的数量、可接受的响应时间以及运行环境的描述等[10], 同时系统使用方式信息也应作为需求的一部分. 这里我们借鉴虚拟用户方法和 WUS 方法收集负载的一些途径, 用以下几种形式较完整地表示系统使用方式信息: 事务分布表、事务简表、用户简表、用户在线行为特征分布表和客户端系统特征分布表[7, 8].

(1) 事务分布表: 记录应用程序各个事务在一天中不同时段的吞吐率情况　表 2 是网上书店在一个高峰负载日的事务分布表, 标有下划线的项表示该事务在一天中高峰负载时段的吞吐率, 单位为事务数/小时　性能测试通常要模拟高峰时段的负载.

表 2　网上书店高峰负载日的事务分布表示例

事务	各个时段事务的吞吐率(每小时提交的事务数)						
	4:00~6:00	6:00~8:00	8:00~10:00	10:00~12:00	12:00~14:00	14:00~16:00	16:00~18:00
注册新用户	500	400	800	1200	1000	500	600
登录	800	320	1050	2500	950	550	370
浏览	1000	8000	12000	31200	26000	12000	8000
搜索	200	1600	8800	16800	13500	3000	2000
放入购物车	100	800	1200	3450	2000	3500	2400
修改数量	110	300	570	600	480	510	170
结账	100	230	760	1050	660	240	150

(2) 事务简表: 记录各事务在高峰时段和非高峰时段的吞吐率、所引起的 Web 服务器负载和数据库负载情况、操作失败带来的风险等信息　示例见表 3:

表 3　网上书店的事务简表示例

事务	典型日	高峰日	Web Server 负载	DB 负载	风险
注册新用户	750	1200	中	轻	低
登录	1050	2500	中	轻	低
浏览	15600	31200	重	中	中
搜索	12000	16800	中	重	高
放入购物车	2100	3450	重	中	中
修改数量	200	600	中	中	高
结账	800	1050	重	重	高

(3) 用户简表:记录用户类型、用户数、平均访问持续时间、事务提交率等信息. 示例见表 4.

表 4 用户简表示例(高峰日)

统计项	随意浏览者	新顾客	老顾客
并发用户数/个	600	150	250
平均访问持续时间/min	10	30	20
注册新用户/(次/小时)	1.5	2	—
登录/(次/小时)	2	2	4
浏览/(次/小时)	36	14	30
搜索/(次/小时)	18	—	24
放入购物车/(次/小时)	—	8	9
修改数量/(次/小时)	—	4	—
结账/(次/小时)	—	2	3

(4) 用户在线行为特征分布表:描述用户之间行为的差异,包括用户交互速度、对延迟的忍耐度、对网站的熟悉度等信息. 示例见表 5、表 6:

表 5 用户对延迟的忍耐度分布示例

忍耐度	用户比例/%	极限/s	忍耐度对脚本的影响
低	25	8	如果某操作的响应时间超过了用户的忍耐极限,该操作将被终止
中	50	16	
高	25	24	

表 6 用户对应用程序熟悉度的分布示例

熟悉度	用户比例/%	思考时间系数	熟悉度对脚本的影响
新手	35	2.0	思考时间系数可用来调节测试脚本中用户提交下一个请求前的思考时间
一般	55	1.0	
很熟	10	0.5	

(5) 客户端系统的特征分布表:描述客户端硬件、软件和位置等之间的差异. 表 7、表 8 所示的分别是连接速度和浏览器类型的分布示例:

表 7 客户连接速度的分布示例

连接速度	比例/%	连接速度	比例/%
56Kbps Modem	20	1.544Mbps T1/LAN	5
112Kbps Dual ISDN	5	10Mbps LAN	25
256Kbps DSL/Cable	5	45Mbps T3/LAN	15
512Kbps DSL/Cable	15	其他	10

表 8 客户浏览器类型的分布示例

浏览器类型	比例/%
Netscape	25
Internet Explorer	50
其他	25

以上各种有关 AUT 的使用方式和用户可接受的响应时间、系统的运行环境等信息构成了性能测试的完整需求,它们是制定测试计划和设计测试场景的依据.

3.2 编制测试计划

确定了性能需求之后,要根据该需求和所采用的测试方法编制可行的测试计划,详细描述测试需求、测试目标、测试策略、测试工具、测试进度和测试资源等,作为测试活动的指南. 基于 IEEE 软件测试文档标准(IEEE Std 829—1998)[11],我们为模拟驱动的自动负载测试方法设计了一个测试计划模板(见附录 A),模板的内容可根据实际应用酌情增删.

3.3 设计测试场景

设计测试场景的关键是使其能代表 AUT 在现实中的真实使用情况,通常遵循以下几个原则:

(1) 测试场景中事务类型的组合要反映真实情况;

(2) 测试执行的速度应反映真实用户执行活动的速度;

(3) 对于每种事务都应该有真实的测试数据.

根据以上原则,我们的方法是将设计测试场景分为 3 个步骤:确定要自动执行的事务组合;确定测试负载;定义每个事务的详细步骤.

3.3.1 确定要自动执行的事务组合

这是整个设计过程中最具风险的步骤,在这个步骤中应给出测试中要自动执行的事务清单. 试图执行所有的事务是困难、费时的,经验表明选择执行 80%的事务比较有效,通常在每个程序模块中选择 5~10 个事务. 我们的方法是选择那些执行频率高、给 Web 服务器和数据库服务器带来较重负载或风险代价大的事务. 根据表 3,我们在网上书店应用程序中选择了 5 个事务:浏览、搜索、放入购物车、修改数量和结账作为测试的基础,如表 9 所示,下划线表示选择的事务和原因.

表 9 选择的自动执行的事务组合示例

事务	典型日负载	高峰日负载	Web Server 负载	DB 负载	风险
注册新用户	750	1200	中	轻	低
登录	1050	2500	中	轻	低
浏览	15600	31200	重	中	中
搜索	12000	16800	中	重	高
放入购物车	2100	3450	重	中	中
修改数量	200	600	中	中	高
结账	800	1050	重	重	高

3.3.2 确定测试负载

用户在访问网站时会表现出一定的特征:某段时间里用户的访问数量服从一定分布;每个用户的访问持续一定的时间,在这期间执行某些操作;每个

用户都有自己的行为特征; 不同的客户端系统还有不同的连接速度、不同的软硬件配置等. 所有用户的操作构成了整个应用程序的负载. 模拟驱动的自动测试方法通过模拟用户的行为来模拟 AUT 的真实使用情况. 根据表 3 ~表 8 所示的系统使用方式信息和表 9 所示的事务组合信息, 我们设计了网上书店的一次测试负载, 如表 10、表 11 所示. 它模拟了高峰时段 1000 个用户同时访问的情况, 定义了不同用户特性的分布百分比. 例如, 表 4 表明每位新顾客平均每小时执行 14 次浏览操作, 因此在一次持续 30 min 的访问中需要模拟 7 次该操作(14× 30/60). 表 5 表明忍耐极限为 8 s 的用户占总用户的 25%, 我们假设 3 类用户中具有这种特征的用户分别占总用户的 15%, 5%和 5%, 这样与 25%的总比例相符.

表 **10**　测试负载示例

用户类型	并发用户数	平均访问时间/ min	事务	每次访问执行次数/ 用户
随意浏览者	600	10	浏览	6(36× 10/ 60)
			搜索	3(18× 10/ 60)
新顾客	150	30	浏览	7(14× 30/ 60)
			放入购物车	4(8× 30/ 60)
			修改数量	2(4× 30/ 60)
			结账	1(2× 30/ 60)
老顾客	250	20	浏览	10(30× 20/ 60)
			搜索	8(24× 20/ 60)
			放入购物车	3(9× 20/ 60)
			结账	1(3× 20/ 60)

表 **11**　测试负载示例(续)

用户类型(百分比)	客户特征分布							
	延迟忍耐度		思考时间		连接速度		浏览器类型	
	时间/ s	百分比	时间/ s	百分比	时间/ s	百分比	时间/ s	百分比
随意浏览者	8	15	3	5	56Kbps	20	IE	35
(60)	16	30	6	35	512Kbps	15	Nets.	15
	24	15	12	20	10Mbps	25	其他	10
新顾客	8	5	3	3	256Kbps	5	IE	5
(15)	16	5	6	5	1. 54Mbps	5	Nets.	5
	24	5	12	7	其他	5	其他	5
老顾客	8	5	3	2	112Kbps	5	IE	10
(25)	16	15	6	15	45Mbps	15	Nets.	5
	24	5	12	8	其他	5	其他	10

表 **12**　事务定义示例: 结账

事务步骤		预期结果
1. 点击"去收银台"链接		显示送货地址输入页面
2. 输入送货地址信息		
输入域	输入数据	
姓名	xxx	
送货地址	y 市 z 街 w 号	
Email	xxx@xxx. com	
3. 按下"确认"键		出现选择付款方式页面
4. 输入付款方式信息		
输入域	输入数据	
付款方式	信用卡	出现确认页面
信用卡号	xxxxxx	
持卡人姓名	xxx	
思考时间: 3 秒		
开始计时		
5. 按下"确认"键		显示订单接受信息和订单号
结束计时		

3. 3. 3　定义每个事务的详细步骤

为每个事务详细定义各个步骤, 并为每个步骤确定输入数据, 以便为后面的脚本开发等阶段提供详细的文档. 每个步骤中用户只提交一个请求. 表 12 所示的是结账事务的定义.

上面的例子只提供了一个用户的输入数据. 由于性能测试要模拟很多用户的行为, 因此对于某些输入域(例如信用卡号和持卡人姓名), 需要为不同的用户定义不同的输入数据. 这样做主要有两个原因: 避免数据冲突引起程序错误; 避免请求的页面已经在缓存中. 如果发生数据冲突, 测试将会失败; 如果请求的页面已经存储在缓存中, 服务器端的活动将不能被驱动, 从而导致测试结果不正确.

3. 4　设置测试环境

这个阶段的目的是安装和配置 AUT 以及测试工具, 创建一个虚拟的测试环境. 测试环境可以由系统管理员来设置. 有几点需要注意:

(1) 需要一个专门的测试环境、专门的网络;

(2) 力求测试环境的硬件, 如处理器速度、处理器数量、内存、磁盘、网络配置等接近 AUT 实际运行环境的硬件配置;

(3) 测试数据库的数据量要与实际数据库一致.

3. 5　开发测试脚本

测试脚本是测试场景的可执行形式[12]. 为了执行设计的场景, 必须按照设计文档开发测试脚本. 这个阶段可以分为 3 个步骤: 初步开发、参数化和验证.

首先利用测试工具自动生成初步的测试脚本;

开发人员录制 AUT 实际的执行过程，该执行过程被自动转化为测试脚本. 运行该脚本时，会重现执行过程. 然后对测试脚本进行参数化以模拟大量不同的用户. 对于那些需要参数化的输入域，将录制的值用一个参数变量来代替，并生成一个包含大量数据的参数文件. 例如在网上书店程序中需要参数化的输入域有：用于查询的关键字、购买的书名、付款方式、信用卡号和持卡人姓名等. 在测试正式运行之前还要对脚本进行调试和验证，以解决多用户同时操作的问题，如死锁、数据冲突等.

3.6 运行/分析/优化

运行测试场景、分析测试结果、优化应用程序是一个反复的过程. 运行最好分阶段进行，通常从较轻的负载（如 3～5 个虚拟用户）到中等负载（如 20%的负载）再到重负载（如满负载 100%和超负载 120%）. 运行测试场景的同时，测试人员还要用测试工具来监视各个性能指标，收集性能数据. 这样做的好处是可以有效地调试系统问题，并且通过逐渐增加用户数可以精确地找到性能下降的转折点. 运行中收集到的性能数据可通过测试工具生成一系列直观的图表和报告，供分析人员处理. 他们通过分析可以发现影响性能的瓶颈，从而提出优化方案. 优化之后测试人员要重新运行测试场景以验证优化的有效性.

3.7 测试总结

测试活动结束后，测试人员需要总结测试结果，整理测试报告. 报告内容包括测试数据的汇总、测试中发现的问题、所做的优化工作以及目标的完成情况等.

4 结束语

Web 性能测试的困难主要表现在负载的不可预知、测试场景的设计困难、测试环境和真实环境的差异等方面. 针对这些困难，本文对已有的各种性能测试方法进行了分析和比较，在此基础上提出了一种简单可行、通用的方法——模拟驱动的自动负载测试方法，并结合一个网上书店的实例详细描述了该方法和过程，指出了保证测试成功的关键因素，附录给出了一个测试计划的模板供参考.

参 考 文 献

1 Zona Research Inc. The economic impacts of unacceptable web-site download speed. 1999. http://www.keynote.com/downloads/whitepapers/economic-impact-of-downloadspeed.pdf

2 A Rudolf, B Pirker. E-Business testing: User perceptions and performance issues. In: Proc of the 1st Asia-Pacific Conf on Quality Software. Los Alamitos: IEEE Computer Society Press, 2000. 315～323

3 D A Menasce, Virgilio A F Almedia. Capacity Planning for Web Performance Metrics, Models & Methods. New Jerseyz: Prentice Hall, 1998. 1～12

4 H Q Nguyen. Testing Applications on the Web. New Jersey: John Wiley & Sons, 2001. 12～26

5 M D Anderson. The top 13 mistakes in load testing applications. Software Testing & Quality Engineering. 1999, 1(5): 31～39

6 Mercury Interactive Corp. Load testing to predict Web performance. 2000. http://www-svca.mercuryinteractive.com/pdf/products/whitepapers/load_testing_web_perform.pdf

7 LoadRunner User's Guide (Version 6.5). Sunnyvale, CA: Mercury Interactive Corporation, 2000

8 A Savoia. The science of website load testing 2000. http://www.keynote.com/downloads/whitepapers/science_of_loadtesting.pdf

9 B M Subraya, S V Subrahmanya. Object driven performance testing in Web applications. In: Proc of the 1st Asia-Pacific Conf on Quality Software. Los Alamitos, California: IEEE Computer Society Press, 2000. 17～26

10 D Grossman *et al*. Performance testing a large finance application. IEEE Network, 1996, 13(5): 50～54

11 IEEE Standards Software Engineering. IEEE Std 829-1998, IEEE Standard for Software Test Documentation. 1999

12 M Fewster, D Graham. Software Test Automation. Boston, MA: Addison-Wesley, 1999. 143～167

梁 晟 女，1976 年生，博士研究生，主要研究方向为智能软件工程、语义 Web 技术

李明树 男，1966 年生，研究员，博士生导师，主要研究方向为智能软件工程、Internet/Web 技术及应用、实时系统

梁金能 男，1957 年生，博士，副教授，主要研究方向为软件测试、软件维护、过程工程和质量改进、软件度量、软件外包

陈振冲 男，1959 年生，博士，副教授，主要研究方向为软件工程、数据挖掘、计算智能

附录 A　Web 性能测试计划模板

(1)介　绍

对本测试计划文档做一些介绍. 列出它的目标、背景、范围和参考文献.

(2) 测试需求

列出测试主要的性能需求, 说明要测试什么.

(3) 测试策略

列出推荐的测试策略, 说明怎样测试. 对每种测试类型, 给出测试的详细描述, 主要说明要采用的测试技术和测试结束的条件.

例如某次测试采用负载测试和压力测试两种策略, 其描述见表 A1 和表 A2.

表 A1　测试策略 1: 负载测试

测试目标	测试技术	结束条件	特殊考虑
测量指定的事务在不同负载条件下的响应时间	模拟驱动的自动负载测试方法	成功地执行所有测试场景, 事务的响应时间都符合测试需求	负载测试要在专门的环境中进行, 以便完全控制和准确测试. 测试数据库规模要与实际一致

表 A2　测试策略 2: 压力测试

测试目标	测试技术	结束条件
验证 AUT 在指定的极限条件下能正常运行; 找出限制系统性能的关键资源, 即性能瓶颈	使用负载测试的测试场景, 但是忽略用户的思考时间	成功地执行所有测试场景, 达到或超出指定的系统极限时仍能无错运行

(4) 测试文档

列出测试活动中将要生成的各种文档和报告, 指出提交人、接收人和提交时间.

测试文档包括测试计划、测试用例说明书、测试过程说明书、测试日志、测试事件报告和测试总结报告等.

(5) 测试环境要求

说明测试环境所需的条件. 指出所需的测试工具.

① 硬件: 指出所需硬件的类型、数量和配置需求.

② 软件: 指出所需的软件和配置需求.

③ 测试工具: 列出工具名称、功能描述、供应商、版本等信息.

(6) 人员资源和职责

列出推荐参加性能测试活动的人员, 他们的主要职责、知识和技能. 例见表 A3.

表 A3　人员资源和职责表

人员	职责
测试经理	实施监督管理; 提供技术指导; 获取适当资源; 撰写管理报告; 撰写测试计划
市场部门	提供 AUT 使用方式信息: 用户数、用户特征分布等; 预测将来访问的增长量
测试设计人员	设计测试场景; 评价测试的有效性
测试实现人员	执行测试; 开发测试脚本/执行测试场景; 测试日志; 错误恢复; 记录变化需求
测试系统管理员	设置、管理和维护测试环境; 管理测试系统的访问权
数据库(网络、Web 服务器)管理员	管理和维护数据库(网络、Web 服务器); 分析测试数据, 优化数据库(网络、Web 服务器)

(7) 测试任务和进度

列出测试的各项具体任务, 确定任务间的依赖关系和完成任务所需的特殊技能, 估计所需时间和人员. 例见表 A4.

表 A4　任务进度表

任务	前驱任务	特殊技能	负责人	所需人数	完成日期
(1) 测试计划	确定需求		测试经理	1	
(2) 测试设计	任务(1)		测试设计人员	2	
(3) 设置测试环境	任务(2)		测试系统管理员	1	
(4) 开发测试脚本	任务(3)		测试实现人员	4	
(5) 执行测试场景	任务(4), (6)		测试实现人员	2	
(6) 分析测试结果、优化系统	任务(5)		数据库(网络、Web 服务器)管理员	4	
(7) 总结测试结果	任务(6)		测试实现人员	1	

注: 表中未给出特殊技能和完成日期的事例, 在实际测试计划中请根据具体情况填写.

(8) 风险和偶然事件

指出测试计划的风险假设, 为每种假设制定一个偶然事件的解决方案.

例如, 假如白天硬件问题影响了系统的可用性, 那么测试队伍将在晚上安排测试活动.

(9) 批准

指出此计划必须提交审批的人员, 并留出空间给他们签名.