第3章　性能测试

学习目标

- 性能测试的基本概念
- 性能测试的流程
- Performance Tester 应用
- 测试结果分析

性能测试在软件的质量保证中起着重要的作用，它包括的测试内容丰富多样。中国软件评测中心将性能测试概括为三个方面：应用在客户端性能的测试、应用在网络上性能的测试和应用在服务器端性能的测试。通常情况下，三方面有效、合理的结合，可以达到对系统性能全面的分析和瓶颈的预测。

3.1　性能测试概述

近年来，用户对软件整体表现的要求越来越高，不仅限于对功能基本的验证上，人们对软件在未来实际运行环境下的性能表现也越来越重视。比如，人们可能需要了解当大量用户同时访问 Web 网站时，请求的响应时间有多长，能否满足其需求？性能测试已经成为测试工作中不可忽视的重要内容，成为发现软件系统性能瓶颈的有效手段。

3.1.1　性能测试的定义

性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试，目的是验证软件系统是否能够达到用户要求的性能指标，同时发现软件系统中存在的性能瓶颈，最后起到优化系统的目的。具体包括以下几个方面：

(1)评估系统的能力：测试中得到的负荷和响应时间数据可以用于验证软件系统能力是否

符合设计要求。

(2)识别系统中的瓶颈:当系统负荷被增加到极限水平并继续增加时,可以通过性能指标的变化情况检测系统的瓶颈,从而帮助系统设计者修复系统的瓶颈。

(3)系统调优:重复运行测试,验证调整系统的活动得到了预期的结果,从而改进性能。

(4)验证稳定性与可靠性:在一定的负载压力下执行测试一定时间,是评估系统稳定性和可靠性是否满足要求的有效方法。

虽然性能测试是一项很复杂、专业性强的工作,但是由于企业 IT 系统的重要性,只有保证其性能的稳定,企业才能对外提供优质服务,因此性能测试越来越得到企业的重视。

目前,典型的企业 IT 系统的架构如图 3-1 所示。

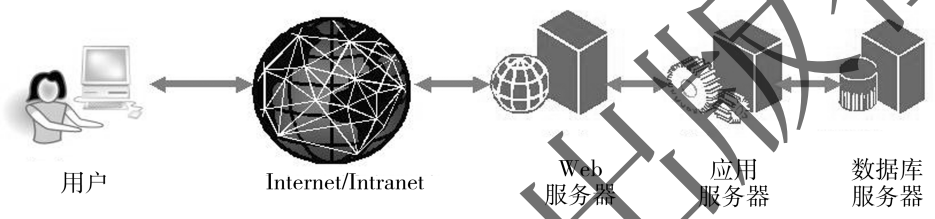


图 3-1　典型的 IT 系统架构

这类系统由客户端、网络、防火墙、负载均衡器、Web 服务器、应用服务器、数据库等环节组成。根据木桶原理,即木桶所能装的水的量取决于最短的那块木板,整个系统的性能要得到提高,每个环节的性能都需要优化。在这样的 IT 系统中,每个环节的都是一个很复杂的子系统,对其调优都是一门专门的技能,例如 Oracle 数据库的调优就需要专门的技能和经验。对于整个 IT 系统的调优,其复杂程度更是急剧增加。因此 IT 系统性能测试调优是一个复杂的项目,需要拥有各种专门技能的专家组成小组来完成。

3.1.2　进行性能测试的原因

为什么要进行性能测试呢?很显然,有了失败的教训,才会让我们重视性能测试。2008 年 8 月,北京奥运会让国人在家门口享受到了体育大餐,但并非所有人都能有幸拿到奥运会门票现场观看比赛。组委会开通的网上售票系统在第一个小时内就达到了 800 万次的访问量,每秒钟从网上提交的门票申请超过 20 万张;票务呼叫中心热线的呼入量超过了 380 万人次。由于瞬间访问量过大,超过了自身设计容量,导致系统瘫痪。

究其原因,系统设计者对系统的访问量估计不足,没有进行充分的性能测试。因此,正确地实施性能测试是保证系统上线后稳定运行的重要手段。

3.1.3　性能测试与功能测试的关系

性能测试和功能测试是测试工作中两个不同的方面,只是在关注的内容上有差异而已。但它们的最终目的都是为了提高软件的质量,以更好的满足用户的需求。软件的“功能”指的是在一般条件下,软件系统能够为用户做什么以及能够满足用户什么样的需求。例如一个网上银

行，用户期望这个网站能够提供转账、汇款、缴费等功能，那么只有这些功能都正确实现了，用户才认为满足了他们的功能需求。从“性能”的角度来说，用户也是有期望需求。例如，服务器需要能够及时处理大量用户的同时访问请求；服务器程序不能出现死机情况；不能让用户等待很久才能打开想要的页面。数据库必须能够支持大量数据的存储以实现对大量的银行业务数据的保存。

因此，软件系统“能不能工作”是一个基本要求，而能够“又快又好”地工作才是用户追求的目标。“好”体现在降低用户硬件资源成本，减少用户硬件方面的支出上；“快”体现在系统反应速度上，用户在进行了某个操作后能够很快得到系统的响应。这些“好、快”体现在软件性能上。也就是说，“性能”就是在空间和时间资源有限的条件下系统的工作情况。

综上，功能关注的是软件“能做什么”的问题；性能关注的是软件所完成的工作“做得如何”的问题。显然，软件性能的实现是建立在功能实现的基础之上的，只有“能做”才能考虑“做得如何”。

在了解了功能和性能的区别之后，再理解功能测试和性能测试就很容易了。功能测试主要针对于软件功能开展测试，常常会依据需求规格说明书；性能测试则主要针对系统性能进行检测，通常会依据性能方面的一些指标或需求进行测试。

3.1.4　性能测试的自动化

性能测试可以通过“手工”和“自动化”两种测试手段实现。与手工性能测试相比，自动化性能测试具有很多优势。

假设要测试一个 Web 系统的性能，验证其是否能支持 500 个用户并发访问，如果采用手工测试的手段，可以按如下步骤实施。

(1)准备足够的资源，500 个用户，每人一台计算机以进行操作支持。

(2)准备一名调度人员以指挥 500 名用户的同步测试。每个参与测试的用户都必须集中注意力，当调度员发出开始测试的指令后进行理论上的并发操作。

(3)在 500 个参与测试的用户“同时”执行操作后，对每台计算机上的测试数据和服务器中的测试数据进行收集和整理。

(4)在缺陷修复后，要开展回归测试，即需要重复执行(1)～(3)直到满足性能需求为止。

很显然，手工测试需要的人力很多，如果要测试 5000 个用户的并发操作，如何寻找这么多的测试人员就是个棘手的问题。其次，支持理论上的并发访问并不是真正想要的性能，而是需要真正意义上的并发访问。再次，回归测试往往需要在相同的场景下进行，在手工测试中，不可能再现上一次的场景。因此，性能测试的手工测试是无法实施的。

借助于自动化测试工具，可以解决以上问题。如图 3-2 所示，应用自动化测试工具进行性能测试的实施方法如下：

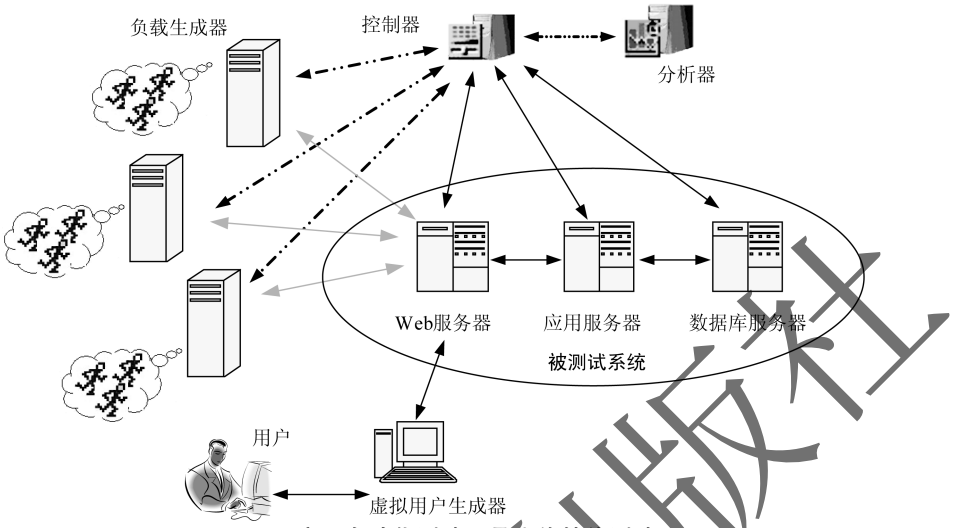


图 3-2　应用自动化测试工具实施性能测试

(1)准备好性能测试工具,如 Rational Performance Tester、LoadRunner、Robot 等。配置负载生成器及计算机,不需要召集 500 个用户和 500 台计算机来协助测试。自动化测试工具能够模拟出 500 个用户的并发访问。与手工测试相比,节省了大量的硬件资源和人力资源。

(2)无须协调人员的统一调度,自动化工具可以自动控制虚拟用户的运行与同步,实现严格意义上的并发操作。

(3)测试完成后,测试工具将自动收集测试数据并分析测试结果。

(4)测试完成并修复缺陷后,要开展回归测试。此时,只需要重新运行上一次测试操作的脚本,即可重现测试场景。

自动化性能测试工具能够帮助测试人员模拟出很多真实复杂的业务场景,能够让系统持续运行数天,能够捕获很多难以捕获的结果,这些都是手工测试所不能完成的。

3.2　性能测试的分类

性能测试设计范围比较广,相关的名称和方法繁多,而且容易混淆,如一般性能测试、负载测试、压力测试、大数据量测试、配置测试、疲劳测试、稳定性测试等,他们均属于性能测试范畴。读者只需要理解它们的关注点,不必严格区分,真正在实施性能测试的时候,一般会综合使用多种方法开展。

(1)一般性能测试主要是验证软件在正常环境和系统条件下,即不施加任何压力情况下重复使用系统,验证其是否能够满足性能指标要求,如响应时间、系统资源占有率等。例如让 1 个人或 10 个人并发访问网上银行,观察系统运行情况。基于假定的前提条件,此时的系统运行一定非常正常,响应时间非常快,系统资源占有量也非常小,这时可记录下各项指标平均值。

通常一般性能测试会在进行负载测试、压力测试等之前进行,作为性能基准测试。通过其

测试得到的数据作为后面各项测试的基准值，即后面测试获得的数据可同其进行对比。

(2)负载测试是确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况，例如吞吐量、响应时间、CPU 负载、内存使用等，来确定系统的性能，直到系统性能出现“拐点”，即某个性能指标达到了预先定义的指标阈值。负载测试是一个分析软件应用程序和支撑架构、模拟真实环境的使用，从而来确定能够接受的性能过程。能够帮助人们了解系统的处理能力，即在某些目标下系统所能承受的负载量极限值，为系统性能调优提供有力依据。

(3)压力测试主要是在“模拟系统已处于极限负载下或某指标已经处于饱和状态”情况下，继续给系统增大负载或运行时间，观察系统性能表现，验证系统是否出现内存泄露、系统宕机等严重异常，有助于进行系统稳定性的验证及性能瓶颈的确定。

(4)大数据量测试包含两层意思，既可指在某些容器(如数据库、存储设备等)中有较大数量的数据记录情况下对系统进行的测试，也可指进行并发或某些操作时创建大量数据来动态的开展测试。大数据量主要是指使用大批量数据对系统产生压力或影响，同时验证系统各项指标运行是否正常。这种测试有助于进行系统可扩展性的验证及性能瓶颈的确定。

(5)配置测试主要是在不同的软硬件配置环境下，进行测试以找到系统各项资源的最优分配原则的一种测试。通常，可以通过“正交试验法”来设计测试用例，从而筛选出一定的软硬件配置组合。配置测试有助于找到最优的配置组合，确定由数据库设置或服务器硬件等造成的性能瓶颈。

(6)疲劳测试是采用系统稳定运行情况下能够支持的最大并发用户数，持续执行一段时间业务，通过综合分析交易执行指标和资源监控指标来确定系统处理最大工作量强度性能的过程。疲劳强度测试可以采用自动化工具进行测试，也可以手工编写程序测试，其中后者占的比例较大。

(7)稳定性测试主要强调的是连续运行被测系统，检查系统运行时的稳定程度。通常采用错误发生的平均时间间隔来衡量系统的稳定性，该值越大，系统的稳定性越强。稳定性测试可以帮助测试者找到一些严重的系统问题，如死机、内存泄露或系统崩溃等。

3.3　性能测试术语

性能测试涉及比较多的术语，如吞吐量、点击率、响应时间、每秒事务数等，它们在性能测试中具有非常重要的作用。

3.3.1　虚拟用户

在测试环境中，一些自动化测试工具在物理计算机上使用虚拟用户来模拟真实用户。虚拟用户以一种可重复、可预测的方式模拟典型用户的实际业务操作，对系统施加压力。虚拟用户的出现和使用能够大大减少人力物力的投入。

3.3.2 并发及并发用户数

在性能测试中,经常能够见到“系统允许 500 个用户并发访问”或者“系统支持 500 个用户并发进行登录操作”等描述,这些都属于并发操作。仔细思考一下,就会发现这两句话所描述的并发操作存在着实质性区别,是两种不同类型的并发。

第一种“并发”含义更为广泛,对于用户访问系统所进行的业务操作没有限制。500 个用户既可以同时执行同一操作,也可以同时执行不同的操作,只要达到“同时”完成即可,这类“并发”更加真实地模拟了用户对系统的实际使用情况;而后一种“并发”限制更严格,明确对于某项操作进行了并发访问要求,即 500 个用户同时完成了相同的登录操作。

可见,“并发”强调的是大量用户的“同时性”操作,该操作需要对服务器产生影响,而具体进行的操作是否相同,则需要结合实际情况进行分析和模拟。

与“并发”直接相关的一个概念是“并发用户数”。它指的是在同一时刻同时进行了对服务器产生影响的操作的用户数量。在前面的例子中,500 个用户同时进行了操作,都向服务器发送了请求,对服务器产生了影响,因此并发用户数就是 500。

需要注意的是,有些情况下读者可能会对“并发用户数”产生误解。如“系统可有 1000 个使用用户”、“系统在运行高峰时,最多支持 800 个用户同时在线”,这两句话描述的数字都不是“并发用户数”,主要是因为以下原因:

(1)“系统可有 1000 个使用用户”表明的是可以使用该系统的全部用户数量为 1000.这 1000 个用户并非都在使用系统,或者说都在对服务器产生影响。因此,这里的 1000 仅能成为系统用户数而不是“并发用户数”。

(2)“系统允许 800 个用户同时在线”表明的是系统最多运行 800 个用户登录系统。这也不是说 800 个用户同时都对服务器产生了影响,例如,有的用户登录系统后,在系统中打开某个网页进行查看,这个“查看”操作在当前没有向服务器发送请求,不会对服务器产生压力,只是在最初用户单击某个链接进入该页面的时刻,向服务器发送了请求,对服务器产生了压力,而后续停在该页面查看不会和服务器再有交互,故不会对服务器产生压力。查看页面的用户属于在线状态,不应算作“并发用户”。因此,800 应称为“在线用户数”,而不是“并发用户数”。

总之:“并发用户数”一定是多个用户同时进行了相同或不同的操作,对服务器产生了影响。它与“系统用户数”、“在线用户数”存在差异。在测试工作中,读者经常要用到的概念是“并发用户数”,这才是真正对服务器产生压力的数值。

既然“并发用户数”对性能测试非常重要,那么如何计算“并发用户数”呢? 通常,可以依据“服务器历史日志分析”获取该数值,或者通过经验进行推导得到。前一种方法获取该值更加准确,经验推导则比较主观,在没有其他有效数据可以参考的情况下,往往根据该公式估算:系统用户数×(10%~30%)。

3.3.3 响应时间

“响应时间”包含两层含义,一是“请求响应时间”,另一个是“事务响应时间”。以图 3-1 为

例，对响应时间相关的知识进行学习。

该图所示为用户请求和响应的过程示意图。用户在客户端进行操作，如单击超链接、单击按钮后，将向服务器发送请求，经服务器处理后，服务器向客户端做出响应。

用户在客户端进行的一次操作，如单击超链接，可能会导致客户端向服务器发送一次或多次请求，如请求 HTML 页面、请求图片、CSS 框架等内容。具体请求次数依据页面不同会有差别。

通过以上讲解，可以得出“请求响应时间”和“事务响应时间”两个概念的区别。

(1)“请求响应时间”为对请求做出响应所需要的时间，即从客户端发出请求到得到响应中间所经历的时间，单位通常为“秒”或“毫秒”。请求时间包括两部分，即“网络响应时间”和“服务器端响应时间”。对性能测试结果进行分析时，若发现请求响应时间过长，则需要分析究竟是网络的原因还是服务器端的原因。

(2)“事务响应时间”与事务有关。事务是一个操作或操作序列，可将所关注的一系列操作定义为一个事务，以使相关人员对这一系列操作的各项指标进行整体度量。“事务响应时间”是指完成该事务的操作所经历的时间。例如，“网上银行转账操作涉及的事务”。每个操作可能设计一次或多次请求，因此，“事务响应时间”中会包含一个或多个“请求响应时间”。

3.3.4　吞吐量与吞吐率

吞吐量(Throughput)与吞吐率是性能测试中常见的性能指标。吞吐量即在单次业务中，客户端与服务器端进行的数据交互总量。通常，该参数受服务器性能和网络性能的影响。

吞吐量除以传输时间，即得到吞吐率。吞吐率是衡量服务器性能和网络性能的重要指标之一。吞吐率一般可用“请求数/秒、页面数/秒、访问人数/天、业务数/小时、字节/天”等单位来衡量。

3.3.5　每秒事务数

每秒事务数(Transaction Per Second，TPS)是指每秒系统能够处理的交易或事务的数量。在性能测试中，经常能够遇到类似“取款业务成功率达到 1000 次/秒”这样的描述，这就是 TPS。TPS 是重要的性能测试指标，能够衡量系统处理能力，比如业务成功率等。

3.3.6　点击率

点击率(Hits Per Second，HPS)是指用户每秒向 Web 服务器提交的 HTTP 请求数。点击率越大，表明对服务器产生的压力也越大。但是，点击率的大小并不能衡量系统的性能高低，因为他并没有代表单击产生的影响。点击率是 LoadRunner、RPT 中常用的术语。

3.3.7　性能计数器

性能计数器(Performance Counter)是一系列用于描述各类服务器或操作系统性能的指

标，在进行资源监控和系统瓶颈分析中起着重要作用。例如在 Windows 任务管理器中使用的 CPU 使用率、进程时间等都是常见的性能计数器，在后面一节中，我们会重点介绍常用的性能计数器。

3.3.8 资源利用率

资源利用率反应对系统(如 Web 服务器、数据库服务器、操作系统等)的各种资源的使用情况，例如 CPU 利用率、内存占有率、磁盘利用率等。资源利用率是性能测试中分析瓶颈、发现问题从而改善性能的主要参数之一。为了方便进行分析，通常用“资源的实际使用量/总的资源数量”的形式来表示资源利用率。

例如，“1000 用户并发进行登录操作时，服务器的 CPU 使用率不超过 70%，内存占用率不超过 8%”，在该描述中 70%和 8%就是具体的资源利用率。通常在性能测试中，会用监控得到的资源利用率数值和预先设定的期望值或业界的一些通用值进行比较，当超出后，则有可能是该资源不足导致了系统瓶颈。如，当 CPU 使用率经常高达 90%以上甚至达到 100%时，则 CPU 可能就是系统瓶颈所在。

3.4 性能计数器

影响一个系统性能的因素主要有：软件因素，包括系统软件、第三方软件等；硬件因素，如内存、磁盘、CPU、网卡等；网络因素，如网络吞吐量、带宽、网络传输速率等。

性能计数器是描述服务器或操作系统性能的一些数据指标。例如，对 Windows 系统来说，使用内存数、进程时间等都是常见的计数器。它在性能测试中起“监控和分析”的作用，在分析系统的可扩展性、进行性能瓶颈的定位时，对计数器取值的分析非常关键。一般单一的性能计数器只能体现系统性能的某一个方面，对性能测试结果的分析必须基于多个不同的计数器。

与性能计数器相关的另一个术语是“资源利用率”，它是指系统各种资源的使用状况。本节介绍常用的各种性能计数器以及部分性能分析方法。常用的 Windows 性能计数器如下。

3.4.1 内存

内存(Memory)问题主要检查应用程序是否存在内存泄漏。如果发生了内存泄漏，Private Bytes 计数器和 Work Set 计数器的值往往会升高，同时 Available MBytes 的值会降低。内存泄漏应该通过一个长时间的，用来研究分析当所有内存都耗尽时，应用程序反应情况的测试来检验。内存性能计数器如表 3-1 所示。

表 3-1　　内存性能计数器

性能计数器	计数器描述	参考信息
Available MBytes	可用物理内存数	如果 Available MBytes 的值很小(4MB 或更小),则说明计算机上总的内存可能不足,或某程序没有释放内存
Pages/Sec	表明由于硬件页面错误而从磁盘取出的页面数,或由于页面错误而写入磁盘以释放工作集空间的页面数	一般如果 Pages/Sec 持续高于几百,有可能需要增加内存,以减少换页的需求(可以把这个数字乘以 4k 就得到由此引起的硬盘数据流量)。Pages/Sec 的值很大不一定表明内存有问题,而可能是运行使用内存映射文件的程序所致
Page Read/Sec	页的硬故障,Pages/Sec 的子集,为了解析对内存的引用,必须读取页文件的次数	阈值>5
Page Faults/Sec	处理器中的页面错误的计数	当处理器向内存指定的位置请求一页出现错误时,这就构成一个 Page Faults,如果该页在内存的其他位置,该错误被称为软错误,用 Transition Faults/Sec 数器衡量;如果该页必须从硬盘上重新读取时,被称为硬错误。许多处理器可以在有大量软错误的情况下继续操作。但是,硬错误可以导致明显的拖延
Cache Bytes	文件系统缓存	默认情况下为 50%的可用物理内存,如 50%的物理内存不够时,它会自己整理物理内存缓存,要关注该计数器的变化趋势
File Cache Hits	文件缓存命中率	File Flushes 是自服务器启动后文件缓存 Cache Hits 刷新次数,如果刷新太慢,会浪费内存;如果刷新太快,缓存中的对象会太频繁的丢弃生成,起不到缓存的作用。通过 File Cache Hits 和 File Cache Flushes 可以得到一个适当的刷新值
Work Set	处理线程时使用的内存页,反映每一个进程使用的内存页的数量	—
Private Bytes	指这个处理不能与其他处理共享的,已分配的当前字节	其值越小越好

3.4.2 处理器

监视“处理器”(Processor)和“系统”对象计数器可以提供关于处理器使用的有价值的信息,帮助测试人员决定是否存在瓶颈。处理器计数器如表 3-2 所示。

表 3-2 处理器性能计数器

性能计数器	计数器描述	参考信息
Processor Time	处理器所用的总时间	如果该值持续超过 95%,表明瓶颈是 CPU。可以考虑增加一个处理器或换一个更快的处理器
User Time	表示耗费 CPU 的数据库操作。如排序、执行聚集函数等	如果该值很高,可考虑增加索引,尽量使用简单的表连接、水平分割大表格等方法来降低该值
Privileged Time	CPU 内核时间是在特权模式下处理线程执行代码所花时间的百分比	如果该参数值和 Physical Disk 参数值一直很高,表明 I/O 有问题。可考虑更换更快的硬盘系统。另外设置 Tempdb in RAM,减低 max async I/O、max lazy writer I/O 等措施都会降低该值。此外,跟踪计算机的服务器工作队列当前长度的 Server Work Queues/Queue Length 计数器会显示出处理器瓶颈。队列长度持续大于 4 则表示可能出现处理器拥塞。此计数器是特定时间的值,而不是一段时间的平均值
DPC Time	CPU 消耗网络上的时间	该比值越低越好。在多处理器系统中,如果这个值大于 50%并且 Processor Time 非常高,加入一个网卡可能会提高性能,提供的网络已经不饱和
Context Switches/Sec	计算机上的所有处理器全都从一个线程转换到另一个线程的综合速率	当正在运行的线程自动放弃处理器时出现上下文转换,由一个有更高优先级就绪的线程占先或在用户模式和特权(内核)模式之间转换系统服务。它是在计算机的所有处理器上运行的所有线程的 Thread: Context Switches/Sec 的总数并且用转换数量衡量。在系统和线程对象上有上下文转换计数器,该值越快越好
Interrupts/Sec	处理器用在处理中断以及推迟处理调用的时间百分比	如果处理器使用率超过 90%,且 Interrupts/Sec 大于 15%则处理器可能负载过重,并发生中断

3.4.3 物理磁盘

物理磁盘(Physical Disk)性能计数器如表 3-3 所示。

表 3-3　　物理磁盘性能计数器

性能计数器	计数器描述	参考信息
Disk Time	指所选磁盘驱动器忙于为读或写入请求提供服务所用的时间的百分比	如果三个计数器都比较大，那么硬盘不是瓶颈。如果只有 Disk Time 比较大，另外两个都比较适中，硬盘可能会是瓶颈。若数值持续超过 80%，则可能是内存泄漏
Avg. Disk Queue Length	指读取和写入请求（为所选磁盘在实例间隔中列队的）的平均数	该值应不超过磁盘数的 1.5～2 倍。要提高性能，可增加磁盘
Average Disk Read/ Write Queue Length	指读取（写入）请求（列队）的平均数	该值越小越好
Disk Reads(Writes)/Sec	物理磁盘上每秒钟磁盘读、写的次数	两者相加应小于磁盘设备最大容量
Average DiskSec/Read	指以秒计算的在磁盘上读取数据的所需平均时间	该值越小越好
Average DiskSec/Transfer	指以秒计算的在磁盘上写入数据的所需平均时间	该值越小越好

判断磁盘瓶颈的方法是通过以下公式来计算：

$$每磁盘\ I/O\ 数=[读次数+(4\times 写次数)]/磁盘个数$$

如果计算出的每磁盘的 I/O 数大于磁盘的处理能力，那么磁盘存在瓶颈。否则，磁盘不存在瓶颈。

3.4.4　IIS 应用服务器计数器

常用的 IIS 应用服务器计数器如表 3-4 所示。

表 3-4　　IIS 服务器计数器

类别	性能计数器	计数器描述	参考信息
CPU	% Total Processor Time	被处理器消耗的处理器时间数量	对于 IIS 应用服务器来说，该计数器的值一般不应超过 85%
物理磁盘	% Total Processor Time	显示磁盘进行读、写活动所花费的时间百分比	—
内存	Available Bytes	可用的剩余物理内存量	IIS 基本占用 2.5 MB，每个附加连接将在此基础上占用 10 kB 左右
	Pool Paged Bytes and PoolNonpaged Bytes	缓冲池，由应用程序和操作系统创建并使用的对象	如果池被填满，则可能发生内存泄露
	Pages/Sec	每秒显示的页数	如果服务器没有足够的内存处理其工作负荷，此数值将一直很高

（续表）

类别	性能计数器	计数器描述	参考信息
Object	Threads	线程是执行处理器指令的基本可执行实体	如果该数值一直持续上升，请打开 Process：Thread Count 计数器并查看所有线程是由哪个实例创建的
进程	Private Bytes Total	显示所有实例已经分配但无法与其他进程共享的当前字节数	—

3.4.5 J2EE 应用服务器计数器

常用的 J2EE 应用服务器计数器如表 3-5 所示。

表 3-5　　J2EE 应用服务器计数器

类别	性能计数器	计数器描述	参考信息
虚拟机（JVM）	Heap Size	JVM 堆大小	该计数器的值是一个实时值
	Heap Free	JVM 可用堆大小	该计数器的值是一个实时值
JDBC 连接池	Waiting For Connection Current Count	等待的连接数量	如果该计数器的值持续比较大，可能需要考虑增加应用服务器的 JDBC 连接池大小
	Connections Total Count	总的 JDBC 连接数	—
	Max Capacity JDBC	连接池的总容量	可以通过该计数器和上两个计数器值的比较获得连接池设置是否合理的结论
	Active Connections Current Count	当前活跃的 JDBC 连接数	当前活跃的 JDBC 连接数。通过该计数器的值可以知道 JDBC 连接的利用率是否合理
执行队列	Execute Thread Current Idle Count	空闲的线程数量	—
	Pending Request Oldest Time	队列请求的最久时间	通过该计数器可以看到队列有无明显的拥塞情况
	Serviced Request Total Count	已处理的请求总数	该计数器的值可以用来和性能测试工具测试后得到的“单击数”等进行对比
	Pending Request Current Count	挂起请求的数量	—

3.4.6　数据库计数器

常用的数据库计数器如表 3-6 所示。

表 3-6　　**数据库性能计数器**

类别	性能计数器	计数器描述	参考信息
系统	Total Processor Time	数据库进程占用 CPU 的时间	在不同的数据库中以不同的名称表示。例如：在 Oracle 中，该计数器被称为 CPU used by this session
	Users Connections	当前的用户连接数	数据库服务器一般都有用户连接数的限制，当应用不合理时，有可能出现连接数超过限制的情况，导致一些异常的发生
内存	Cache Hits Ratio	缓存命中率	该计数器结合其他一些计数器，如 Connection Memory、SQL Cache Memory、Lock Memory 等，可以很清楚知道 Memory 的使用情况
	Total Server Memory（仅用于 SQLServer）	SQL Server 数据库进程当前所使用的内存量	—
	PGA Memory/UGA Memory（仅用于 Oracle）	Oracle 数据库进程的内存情况	—
锁	Average Wait Time	锁平均等待时间	—
	Lock Requests/Sec	每秒的锁请求数	—
	Numbers of Deadlocks/Sec	每秒产生的死锁的数量	当该计数器的值比较大时，需要查找产生死锁的原因
I/O	Outstanding Reads（Writes）	被挂起的物理读（写）	当该技术器的值比较大时，可能是 CPU、磁盘 I/O 产生了瓶颈，可以通过服务器的 CPU 和 I/O 分析了解进一步的原因
	Page Reads/Sec	每秒页面读写的次数	—
	Transactions/Sec	每秒产生的事物量	—

3.5 性能测试流程

性能测试主要是通过自动化的测试工具模拟多种正常峰值以及异常负载条件来对系统的各项性能指标进行测试。性能测试原理的实现主要包含三点:用户行为模拟、性能指标监控、性能调优。其中用户行为模拟借助于自动化测试工具,产生用户行为脚本,并运用负载生成器将用户操作模拟为成千上万的虚拟用户对被测系统进行操作。在系统运行过程中需要监控各项性能指标,并分析指标的变化情况。通过指标的监控发现系统存在的性能问题,利用分析工具进行定位并找到修复方案。

在每种不同的系统架构的实施中,开发人员可能选择不同的实现方式,因此,对测试人员来说,被测试系统多种多样,情况比较复杂。很难说有一种通用的性能测试方法或者技术能适用于所有类型系统的性能测试。不过仍然有一些通用的步骤帮助我们完成一个性能测试项目。通用的步骤为:性能测试计划、性能测试设计、性能测试执行、性能测试结果分析。

3.5.1 性能测试的计划

制定测试计划的目的是为了约束测试各个活动的起止时间,为性能测试的准备、执行、分析与报告、总结等环节给出合理时间估算。每一个性能测试计划中第一步都会制定目标和分析系统构成。只有明确目标和了解系统构成才会澄清测试范围,知道在测试中要掌握什么样的技术。计划阶段一般需要确定如下问题。

(1)目标。

确定客户需求和期望。例如:客户能接受的响应时间、每日单交易处理能力、系统资源利用率、系统环境搭建方式、并发用户数、日交易数量等。

(2)系统组成。

系统组成这里包含几方面含义:系统类别,系统构成,系统功能等。了解这些内容有助于测试人员明确测试的范围,选择适当的测试方法来进行测试。

(3)系统功能。

系统功能指系统提供的不同子系统,如办公管理系统中的公文子系统、会议子系统等,系统功能是性能测试中要模拟的环节,了解这些是必要的。

性能测试计划没有统一的模板,基本内容主要是所谓的 5W1H,即:When 何时、Who 何人、Where 何地、What 何事、Why 为什么、How 如何进行。

3.5.2 性能测试的设计

设计阶段主要是设计测试用例。设计测试用例是在了解软件业务流程的基础上进行的。设计的原则是受最小的影响提供最多的测试信息,设计测试用例的目标是一次尽可能的包含多个测试要素。这些测试用例必须是测试工具可以实现的,不同的测试场景将测试不同的功能。

因为性能测试不同于平时的测试用例，尽可能把性能测试用例设计的复杂，才有可能发现软件的性能瓶颈。

性能测试用例设计通常不会一次设计到位，它是一个不断迭代完善的过程，即使在使用过程中，也不是完全按照设计好的测试用例来执行，需要根据需求的变化进行调整和修改。

3.5.3　性能测试的执行

通过性能测试工具运行测试用例。同一环境下做的性能测试得到的测试结果是不准确的，所以在运行这些测试用例的时候，需要在不同的测试环境下、不同的机器配置上运行。

3.5.4　性能测试的分析

运行测试用例后，收集相关信息，进行数据统计分析，找到性能瓶颈。通过排除误差和其他因素，让测试结果体现接近真实情况。不同的体系结构分析测试结果的方法也不同，B/S 结构我们会分析网络带宽、流量对用户操作响应的影响，而 C/S 结构我们可能更会关心系统整体配置对用户操作的影响。

3.6　性能测试需求分析

本节重点介绍性能需求的获取方法。

3.6.1　什么是性能需求

性能需求可分为隐式性能需求和显式性能需求。隐式性能需求通常由普通客户提出，他们往往不了解性能指标，不能明确地提出具体的性能需求，因此这类需求需要需求人员采用合理的方式去协助客户明确需求，甚至需要开发方来提供需求指标，然后再由客户进行确认。因此，隐式性能需求需要读者结合实际情况仔细分析，最终得出显式性能需求。显式性能需求一般由专业客户提出，这类客户往往具备自己的开发部门和测试团队，他们非常清楚系统处理业务量的分布，能够明确指出系统应该达到的目标，这类需求更加明确。

3.6.2　常用的性能需求获取方法

(1)根据用户明确要求。

依据用户明确给出的测试相关数据和指标是分析系统性能需求最直接、最简单的方法。专业客户如金融、医疗、政府部门等以及国外客户大多会给出比较明确的性能需求，如响应时间、并发量、服务器资源等，作为开发方，只需进行整理后参照明确指标进行测试即可。

(2)根据用户提供的已有数据整理分析得出。

客户提供的已有数据是指客户业务交易的纸质数据、客户旧版本系统中的历史数据，如服务器日志、数据库记录等。如一个未曾使用过电子系统的保险公司，获取需求时可以通过汇总

已有投保纸质单据进行分析，得出各地域及每年某个时间段的投保、理赔数量、主要投保及理赔的险种业务等来进行性能测试。若该公司已有旧版本的电子投保系统，则旧版本的运行系统中存在了大量有价值的数据。例如，Web 服务器（Tomcat、WebLogic 等）的日志中记录了系统访问情况以及出错信息等内容，测试时可以依据日志信息分析客户业务量，以及每年、每月、每周、每天的峰值业务量。此时，以充分的真实业务数据做参考得出的性能需求显得更加真实有效。

（3）依据同行业中类似项目或类似行业中的数据。

此方法包含了两种情况：一种是“依据同行业中类似项目的数据”，另一种为“依据类似行业中的数据”。这两种情况所表达的含义是一致的。当自己没有某些资源时，可以借助外部力量来实现性能目标的获取。比如，有人可能会问：需求中没有说明性能需求，只说要做一个网站的性能测试。此时，如何开展性能测试呢？这时候，要先分析用户群特征，然后参考其他同行业公布出来的数据进行测试。

下面给出几个依据同行业中类似项目的数据或类似行业中的数据得出性能需求的几个例子：

在某企业网站的成功解决方案中介绍该方案的优势为“实现了 7×24 小时稳定运行要求，系统可承受 4000 用户同时访问，0.5 秒快速响应用户的请求……”，其中的数据可作为性能需求的参考。

- 有一些网站首页本身就提供了点击量、浏览量等统计信息，尽管在许多时候不能完全照搬这些数据，但这些信息仍然具有较强的参考价值。
- 在开发一个保险类软件时，除了可从客户发布的一些成功解决方案中获取数据，还能主动去搜索一些数据，例如，可以咨询某保险公司：“我想购买贵公司的保险，但首先要了解一下投保情况和理赔的数据或比例等，好让自己更确信公司的诚信……”。但这种方式获得的数据肯定是不够真实的。
- 可以借助一般的 B/S 架构的项目性能目标，例如 2/5/10 原则，即 2 秒的响应最可接受的，5 秒是可接受的，10 秒是最大可忍受的时间。

（4）根据二八原则分析计算得出。

在性能测试需求获取中，经典的二八原则可以理解为：每个工作日中 80％的业务集中在 20％的时间内完成；80％的功能只会在 20％的用户群内访问，或者说 80％的用户只使用 20％的功能。

举例如下：

每年业务量集中在 8 个月，每个月 20 个工作日，每个工作日 8 小时，即每天 80％的业务在 1.6 小时完成。去年全年处理业务约 100 万笔，其中 15％的业务处理中每笔业务需对应用服务器提交 7 次请求；其中 70％的业务处理中每笔业务需对应用服务器提交 5 次请求；其余 15％的业务处理中每笔业务需对应用服务器提交 3 次请求。根据以往统计结果，每年的业务增量为 15％，考虑到今后 3 年业务发展的需要，测试需按现有业务量的两倍进行。请根据上述数据进行测试强度的估算。

首先，每年总的请求数为：

(100×15%×7+100×70%×5+100×15%×3)×2=1000 万次/年

每天请求数为：

1000/(20×8)=6.25 万次/天

每秒请求数为：

(62500×80%)/(8×20%×3600)=8.68 次/秒

即，服务器处理请求的能力应达到 9 次/秒。

(5)任务分布图。

如果使用上面的某种方法获取了客户业务相关的某些数据，则可借助于任务分布图作进一步的分析。任务分布图是以一种直观的方式展示性能测试需求，描述一些交易任务在 24 小时内的交易情况，重点关注并发用户的数量以及典型的业务操作。如表 3-7 是某网上书店的任务分布图，从中可见：12:00～16:00 的交易混合程度较高，“查询图书”任务的并发用户在 14:00 达到最大。

表 3-7　　某网上书店网站的任务分布图

典型业务	并发用户数											
登录				2	5	30	35	6		70	75	3
注册					2	20	10	3		30	25	
查询				3	8	50	40	8				
放入购物车				2	3	30	30	6				
支付					1	25	20	5				
时间	2	4	6	8	10	12	14	16	18	20	22	24

任务分布图同时也支持对全年或某个月的任务分布情况作统计分析。

3.6.3　通过服务器日志获取需求

一般的 Web 服务器有两部分日志：一是运行中的日志，它主要记录运行的一些信息，尤其是一些异常错误日志信息；二是访问日志信息，它记录访问的时间、IP、访问的资料等相关信息。为了获取系统性能测试的需求，可以分析 Web 服务器的访问日志以了解系统更多真实负载和主要的业务场景。下面介绍一下利用 Tomcat 产生的访问日志数据所做的有效分析。首先是配置 tomcat 访问日志数据，默认情况下访问日志没有打开，配置的方式如下：

编辑 ${catalina}/conf/server.xml 文件(注：${catalina}是 Tomcat 的安装目录)，把以下的注释(<! ---->)去掉即可。

```
<! --
<Valve className="org.apache.catalina.valves.
AccessLogValve"
directory="logs"
prefix="localhost_access_log." suffix=".txt"
pattern="common" resolveHosts="false"/>
-->
```

其中，directory 是产生的目录 tomcat 安装 ${catalina}作为当前目录；pattern 表示日志生产的格式，common 是 tomcat 提供的一个标准设置格式。其具体的表达式为%h %l %u %t "%r" %s %b。通过这个配置能得到的数据有：

```
%h 访问的用户 IP 地址
%l 访问逻辑用户名，通常返回'-'
%u 访问验证用户名，通常返回'-'
%t 访问日志时
%r 访问的方式(post 或者是 get)，访问的资源和使用的 http 协议版本
%s 访问返回的 http 状态
%b 访问资源返回的流量
%T 访问所使用的时间
```

有了这些数据，可以根据时间段作以下的分析处理：

(1)独立 IP 数统计。

(2)访问请求数统计。

(3)访问资料文件数统计。

(4)访问流量统计。

(5)访问处理响应时间统计。

(6)统计所有 404 错误页面。

(7)统计所有 500 错误的页面。

(8)统计访问最频繁页面。

(9)统计访问处理时间最久页面。

(10)统计并发访问频率最高的页面。

3.7 IBM Rational Performance Tester

3.7.1 RPT 概述

IBM Rational Performance Tester(RPT)也是一款性能测试工具，适用于基于 Web 的应用程序的性能和可靠性测试。RPT 将易用性与深入分析功能相结合，从而简化了测试创建、负载生成和数据收集，以帮助确保应用程序具有支持数以千计并发用户并稳定运行的性能。

安装 RPT 7.0 完成之后，启动 RPT，方法是通过“开始＞程序＞ IBM 软件开发平台＞ IBM Rational Performance Tester”。启动过程中，RPT 会弹出工作空间启动程序窗口，用户通过该窗口设置工作空间目录，这与 MyEclipse 启动时的窗口是一样的，这是因为 RPT 是基于 Eclipse 平台的工具。

启动后，进入 RPT 主界面，通过该界面中的按钮，可以了解并根据教程学习 RPT。点击最右侧的  图标，转至工作台窗口，如图 3-3 所示。

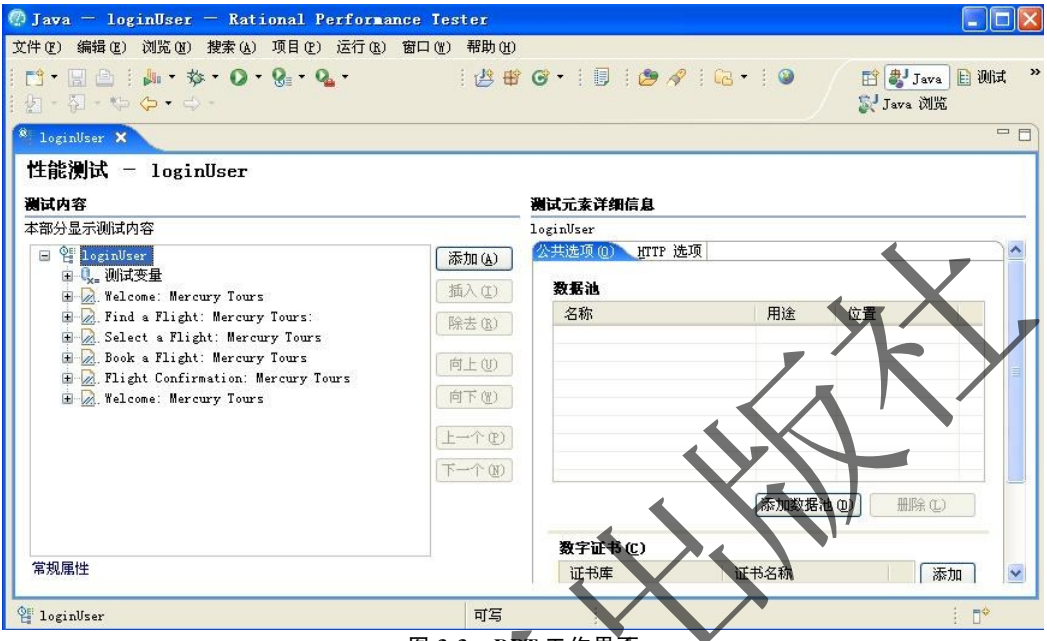


图 3-3　RPT 工作界面

3.7.2　RPT 的基本使用

(1)新建测试项目。

RPT 中的测试脚本由测试项目进行管理,因此在录制测试脚本前,需要首先创建一个测试项目,这一点与 LoadRunner 不同。新建性能测试项目的方法是,在菜单栏中选择“文件>新建>性能测试项目”,弹出窗口中输入项目名称,比如 myProject,点击“完成”,如图 3-4 所示。

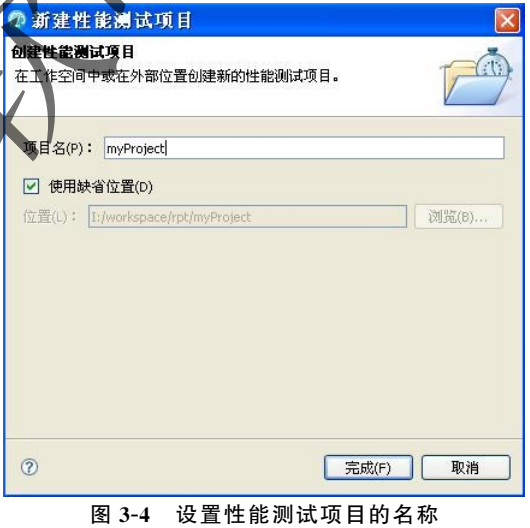


图 3-4　设置性能测试项目的名称

点击“完成”后,会弹出如图 3-5 所示的对话框,提示用户选择记录器来录制测试脚本。一

般新建的测试脚本都是选择第一项“根据新记录来创建测试”，记录器选择“HTTP 记录”。

图 3-5　选择记录器

选择完毕后，点击“下一步”，提示用户选择项目和输入文件名，如 webTours，输入完毕后，点击“完成”按钮，就将启动测试脚本的录制，如图 3-6 所示。

图 3-6　选择项目和文件名

(2)录制测试脚本。

在图 3-6 中点击“完成”后，将启动脚本录制过程。RPT 将启动浏览器，用户在 IE 地址栏中输入被测系统的路径 http://newtours.demoaut.com。进入 Mercury Tours 网站后，输入用户名和密码登录系统(用户名和密码需要提前注册)，登陆成功后，点击“SignOff”退出系统。点击停止记录按钮停止录制，在窗口下方“记录控制器”中显示内容如图 3-7 所示。如果“记录的千字节数”为 0，说明本次录制失败，需要重新录制。

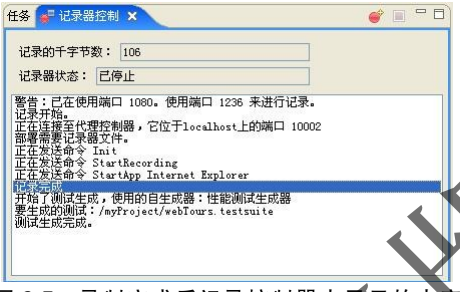


图 3-7　录制完成后记录控制器中显示的内容

将视图切换到 Package Explorer(包资源管理器)视图，在 src 目录下生成了三个文件：webTours.rec、webTours.recmodel、webTours.testsuite，其中 webTours.testsuite 为测试定义文件，可以通过该文件查看本次测试的内容，如图 3-8 所示。

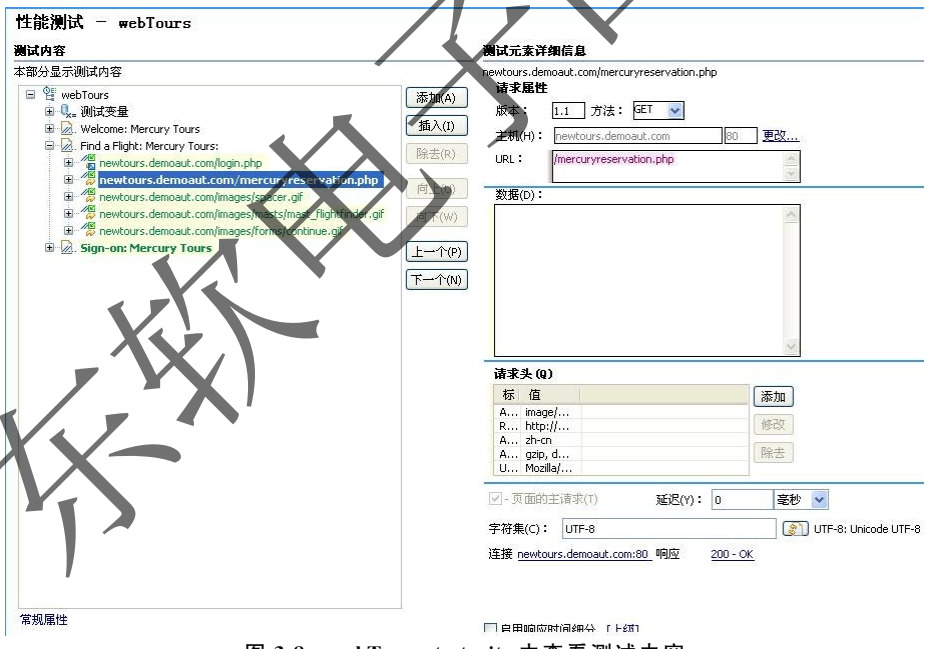


图 3-8　webTours.testsuite 中查看测试内容

(3)执行测试脚本。

要执行测试脚本，可以在图 3-8 中选中测试名称，在工具栏中点击“运行”按钮，然后在运行方式中选择“性能测试”，如图 3-9 所示。

图 3-9　选择运行方式

或者使用快捷键 Alt＋Shift＋X，弹出列表后，选择“运行性能测试”。运行完成后，生成性能报告，其中总体性能报告如图 3-10 所示。

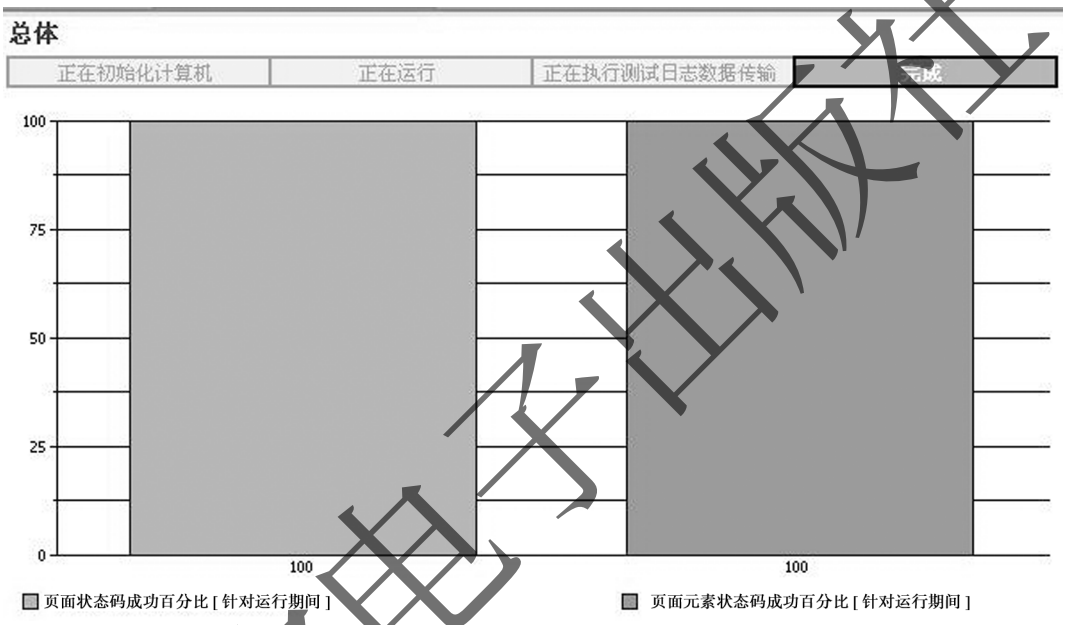


图 3-10　性能报告

通过柱状图中的百分比信息，可以判断测试是否执行成功。在性能报告中还有其他结果信息，如响应时间、页面性能等，可以在视图中选择对应的标签来查看。

①选择“摘要选项卡”，能够看到运行摘要信息，如图 3-11 所示。

运行概要

已执行的测试	webTours
活动用户	0
已完成的用户数	1
用户总数	1
耗用时间 [H:M:S]	0:00:46
运行状态	完成
显示计算机的结果：	所有主机

图 3-11　运行摘要

可以看出本次运行的测试名称、运行状态、耗用的时间等信息。

页面元素摘要如图 3-12 所示。可以看到页面元素的平均响应时间等信息。

页面元素摘要

页面元素尝试总数 [针对运行期间]	37
页面元素命中总数 [针对运行期间]	37
所有页面元素的平均响应时间 [毫秒] [针对运行期间]	462.86
所有页面元素的响应时间的标准偏差 [毫秒] [针对运行期间]	307.76

图 3-12　页面元素摘要

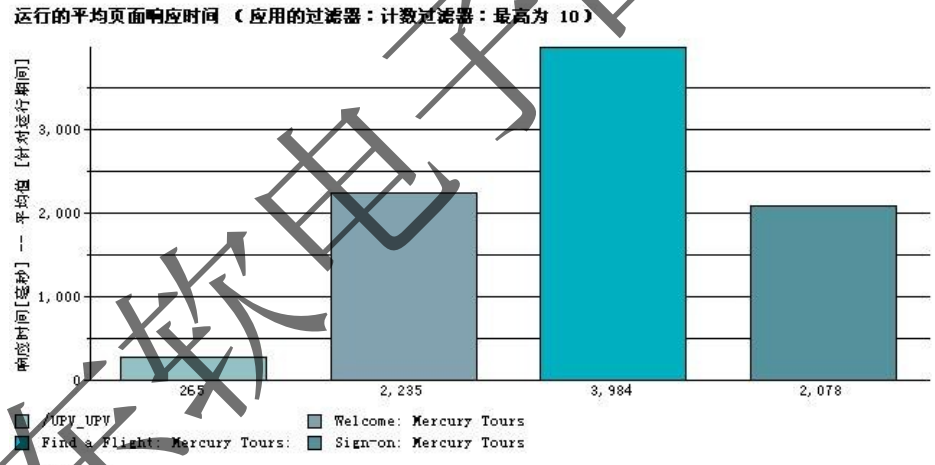
页面摘要如图 3-13 所示。其中列出了所有页面的平均响应时间以及最长响应时间、最短响应时间等信息，可以据此对系统性能做出大体的判断。

页面摘要

页面尝试总数 [针对运行期间]	4
页面命中总数 [针对运行期间]	4
所有页面的平均响应时间 [毫秒] [针对运行期间]	2,140.5
所有页面的响应时间的标准偏差 [毫秒] [针对运行期间]	1,519.74
所有页面的最长响应时间 [毫秒] [针对运行期间]	3,984
所有页面的最短响应时间 [毫秒] [针对运行期间]	265

图 3-13　页面摘要

②选择页面性能选项卡，如图 3-14 所示。



性能摘要

	响应时间[毫秒] -- 最小值 [针对运行期间]	响应时间[毫秒] -- 平均值 [针对运行期间]	响应时间[毫秒] -- 标准偏差 [针对运行期间]	响应时间[毫秒] -- 最大值 [针对运行期间]	尝试次数 -- 速率[每秒] [针对运行期间]	尝试次数 -- 计数 [针对运行期间]
/UPV_UPV	265	265	0	265	0.02	1
Welcome: Mercury Tours	2,235	2,235	0	2,235	0.02	1
Find a Flight: Mercury Tours:	3,984	3,984	0	3,984	0.02	1
Sign-on: Mercury Tours	2,078	2,078	0	2,078	0.02	1

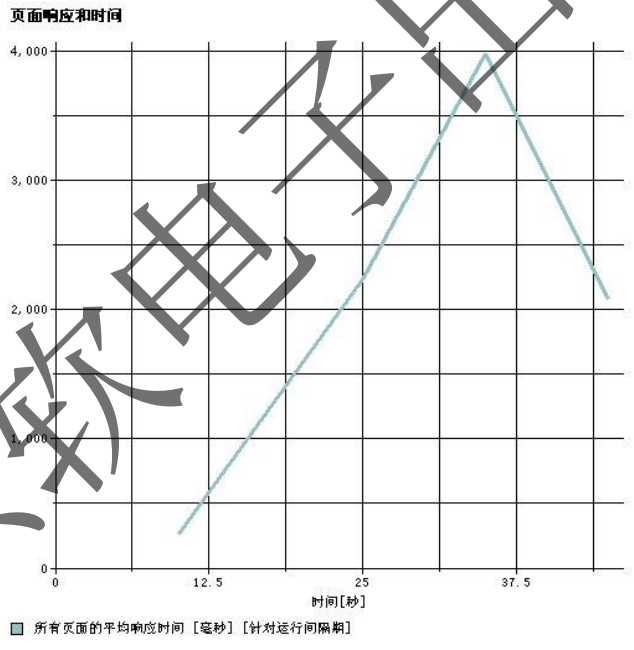
图 3-14　页面性能选项卡信息

这里显示了运行测试的页面性能信息，柱状图为本次运行访问的各个页面的响应时间信息。下方的表格则以另一种方式显示了各页面的响应时间信息，例如表格第三行中显示的是“Find a Flight：MercuryTours”页面，其响应时间的最小值为 3984 毫秒，平均值为 3984 毫秒，标准偏差为 0，最大值为 3984 毫秒。本次运行只访问了该页面一次，因此其最小值、最大值、平均值是一样的。

③响应与时间摘要选项卡如图 3-15 和图 3-16 所示。分别显示的是页面响应和时间及性能摘要，页面元素响应和时间及性能摘要。

④响应与时间详细信息如图 3-17 所示，它以图形和表格的形式详细的展示平均页面响应时间及性能摘要信息。

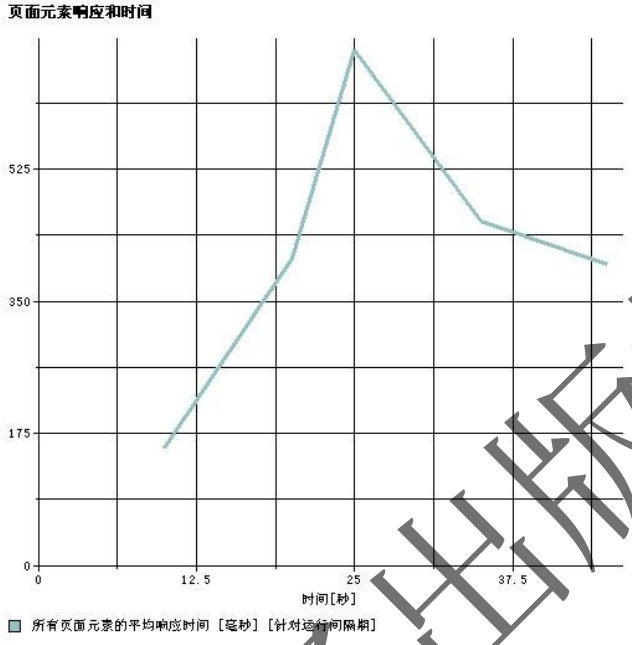
⑤页面吞吐量选项卡显示了页面点击率、用户负载等性能信息。页面点击率及性能摘要如图 3-18 所示。用户负载及性能摘要如图 3-19 所示。



性能摘要

所有页面的平均响应时间 [毫秒] [针对运行期间]	2,140.5
所有页面的响应时间的标准偏差 [毫秒] [针对运行期间]	1,519.74

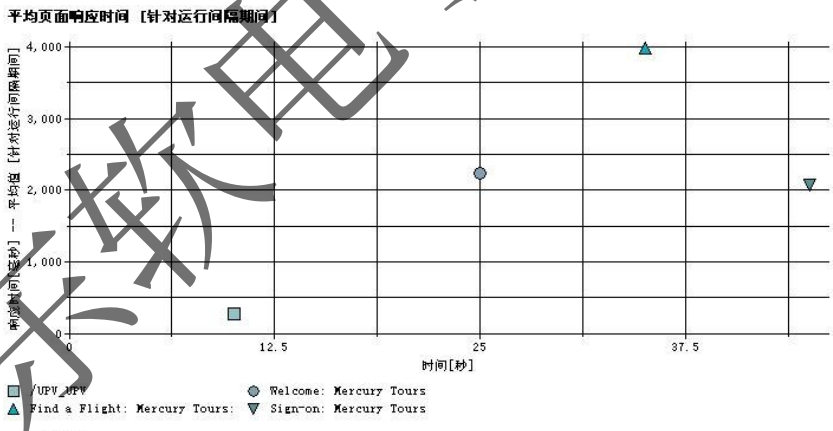
图 3-15　页面响应和时间及性能摘要



性能摘要

所有页面元素的平均响应时间 [毫秒] [针对运行期间]	462.86
所有页面元素的响应时间的标准偏差 [毫秒] [针对运行期间]	307.76

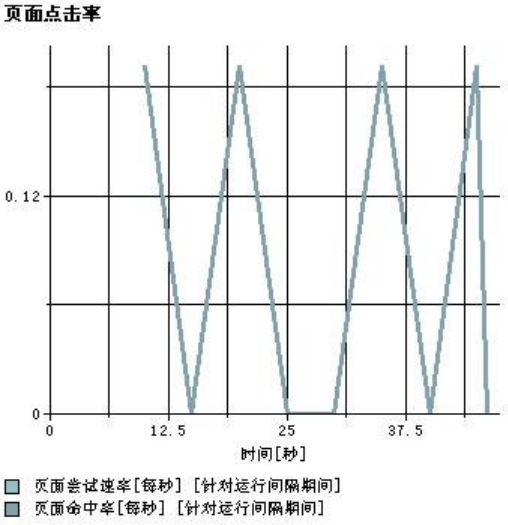
图 3-16　页面元素响应和时间及性能摘要



性能摘要

	响应时间 [毫秒] -- 最小值 [针对运行期间]	响应时间 [毫秒] -- 平均值 [针对运行期间]	响应时间 [毫秒] -- 最大值 [针对运行期间]	响应时间 [毫秒] -- 标准偏差 [针对运行期间]	尝试次数 -- 速率 [每秒] [针对运行间隔期间]	尝试次数 -- 计数 [针对运行间隔期间]
/UPV_UPV	265	265	265	0	0	0
Welcome: Mercury Tours	2,235	2,235	2,235	0	0	0
Find a Flight: Mercury Tours:	3,984	3,984	3,984	0	0	0
Sign-on: Mercury Tours	2,078	2,078	2,078	0	0	0

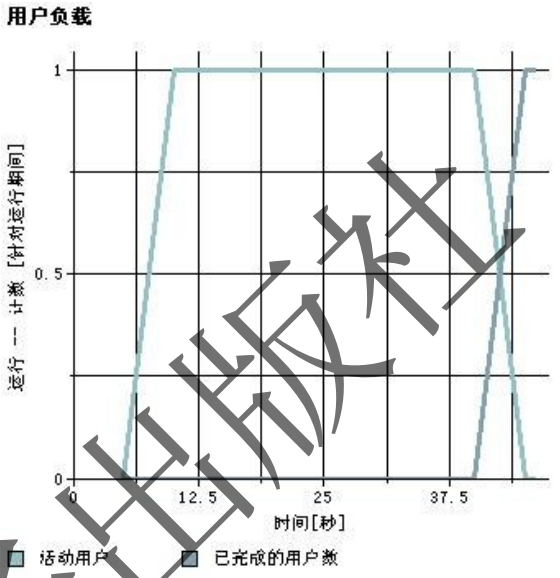
图 3-17　平均页面响应时间及性能摘要



性能摘要

	命中次数 -- 速率[每秒] [针对运行期间]	命中次数 -- 计数 [针对运行期间]
/UPV_UPV	0.02	1
Welcome: Mercury Tours	0.02	1
Find a Flight: Mercury Tours:	0.02	1
Sign-on: Mercury Tours	0.02	1

图 3-18　页面点击率及性能摘要

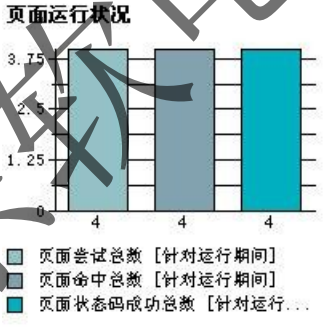


性能摘要

活动用户	0
已完成的用户数	1
用户总数	1

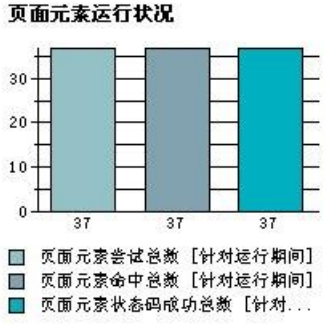
图 3-19　用户负载及性能摘要

⑥服务器运行状况摘要选项卡如图 3-20 所示。



性能摘要

页面尝试总数 [针对运行期间]	4
页面命中总数 [针对运行期间]	4
页面状态码成功总数 [针对运行期间]	4



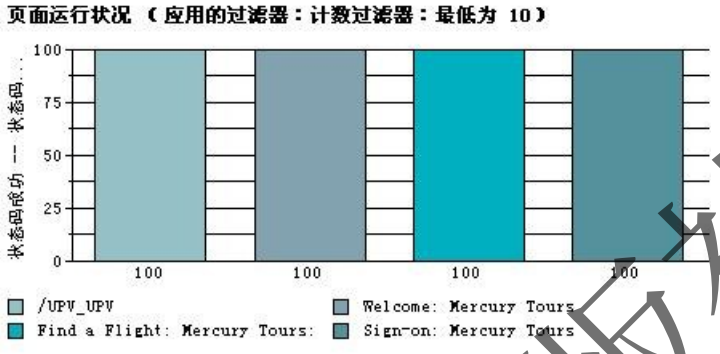
性能摘要

页面元素尝试总数 [针对运行期间]	37
页面元素命中总数 [针对运行期间]	37
页面元素状态码成功总数 [针对运行期间]	37

图 3-20　服务器运行状况摘要

这里显示了页面运行状况及页面元素运行状况。

⑦服务器运行状况详细信息则显示了状态码成功计数、百分比等信息。如图 3-21 所示。



性能摘要

	尝试次数 -- 计数 [针对运行期间]	命中次数 -- 计数 [针对运行期间]	状态码成功 -- 计数 [针对运行期间]	状态码成功 -- 状态码成功百分比 [针对运行期间]	尝试次数 -- 速率[每秒] [针对运行期间]
/UPV_UPV	1	1	1	100	0.02
Welcome: Mercury Tours	1	1	1	100	0.02
Find a Flight: Mercury Tours:	1	1	1	100	0.02
Sign-on: Mercury Tours	1	1	1	100	0.02

图 3-21　页面运行状况及性能摘要

3.7.3　测试验证点的设置

RPT 中的验证点和 LoadRunner 中的检查点功能类似，用于验证期望的行为是否发生，如果实际值和期望值不一致，验证点就会运行失败。

RPT 提供了四种验证点：

(1)页面标题验证点。

用于验证网页的标题是否和预期的一致，对大小写敏感。插入页面标题验证点的方法是：在图 3-22 所示的窗口中，点击左侧“Welcome：Mercury Tours”步骤，然后在窗口右下角勾选“启用验证点”，在“预期的页面标题”中输入期望的页面标题“Welcome：Mercury Tours”(预期值可以设置为其他值)，如图 3-22 所示。在执行测试时，如果期望值和实际值不一致，测试将失败。

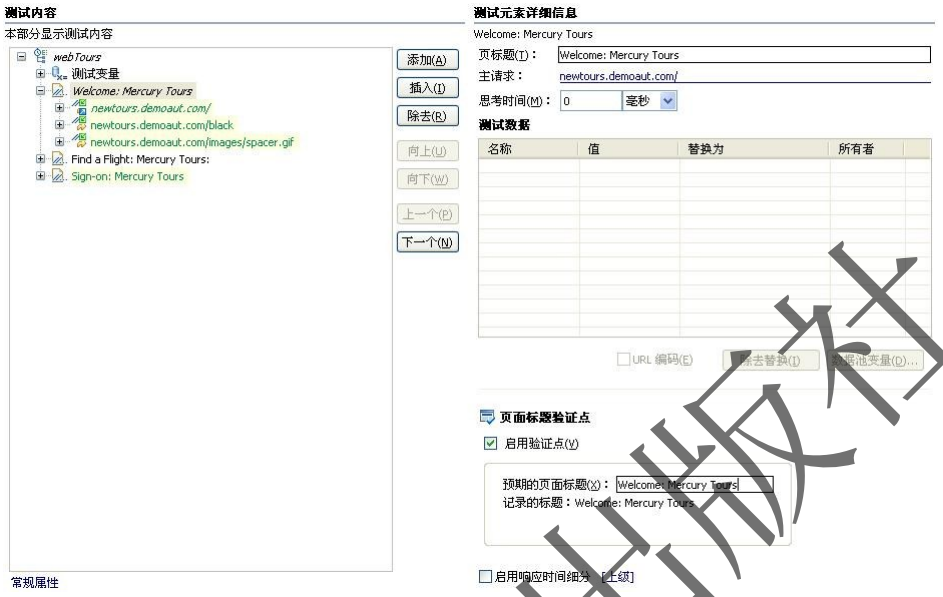


图 3-22　页面验证点的建立

(2)响应代码验证点。

响应代码验证点可以用来验证发出请求后得到的响应的代码是否与期望的一致，如 200 表示客户端请求成功，302 表示对象已经移动，404 表示未找到客户端的请求资源。在图 3-23 所示窗口中，在左侧的树形列表中展开“newtours. demoaut. com/”，然后选中“响应 200：OK”，点击按钮列表中的“添加”按钮，然后选择“响应代码验证点”，在该请求的响应下将增加一个“响应代码验证点”文件夹，如图 3-23 所示。

图 3-23　响应代码验证点

在匹配方法中，有模糊和精确两种匹配方法。如果选择模糊匹配，响应代码允许与期望值存在一定的偏差，执行时也会通过。

(3)响应大小验证点。

插入响应大小验证点的方法与响应代码验证点类似，如图 3-24 所示。响应大小的匹配方法有精确、至少、至多、范围(字节)、范围(百分比)等几种，用户在测试过程中根据实际测试需求

进行选择。

图 3-24　响应大小验证点

(4)内容验证点。

内容验证点用来测试响应内容中是否有指定的内容，插入方法同响应大小验证点，如图3-25所示。在“验证失败”中有两个选项，可以根据实际需要选择其中一种。“可用字符串”中可以编辑用于内容验证的字符串。

图 3-25　内容验证点设置

3.7.4 RPT 数据池的应用

RPT 中数据池(Datapool)的原理和前面章节中学习的 RFT 中的数据池的原理是一样的，通过数据池可以获得动态的数据，它主要用于存储测试数据，在脚本中插入数据池命令并增加相关的控制命令后，在脚本回放时就可以自动从数据池中取出数据，完成多组测试数据的测试。

在“测试导航器”中选择要创建数据池的项目，右键单击，选择“新建＞数据池”，出现如图 3-26所示的窗口。

图 3-26 新建数据池

在弹出的窗口中输入数据池文件的名称 users，点击“下一步”，出现图 3-27 所示的窗口。

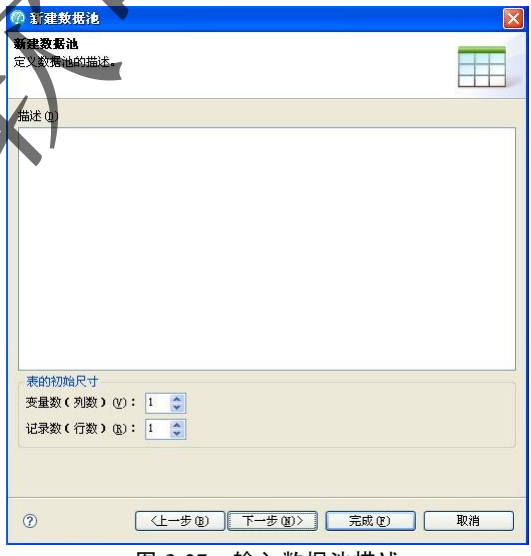


图 3-27 输入数据池描述

在该窗口中输入数据池的描述文字“用户名称”，其他默认，点击“完成”按钮，得到一个空的数据池，在 RPT 窗口中可以看到该数据池，如图 3-28 所示。

图 3-28　数据池

修改变量名称可单击上图中的变量名，如图 3-29 所示，然后在弹出窗口中输入新的变量名称 username，然后点击“确定”。

然后单击单元格，输入一批数据，输入完成后，保存，如图 3-30 所示。

图 3-29　修改变量名

图 3-30　添加数据之后的数据池

数据池及数据准备完成后，就可以替换数据了，步骤如下：

在 RPT 窗口中显示测试内容标签，单击选中测试名称 webTours，在右侧公共选项中，单击“添加数据池”，如图 3-31 所示。

图 3-31　添加数据池按钮

弹出如图 3-32 所示的添加数据池窗口，选中 username 数据池，然后点击“下一步”。

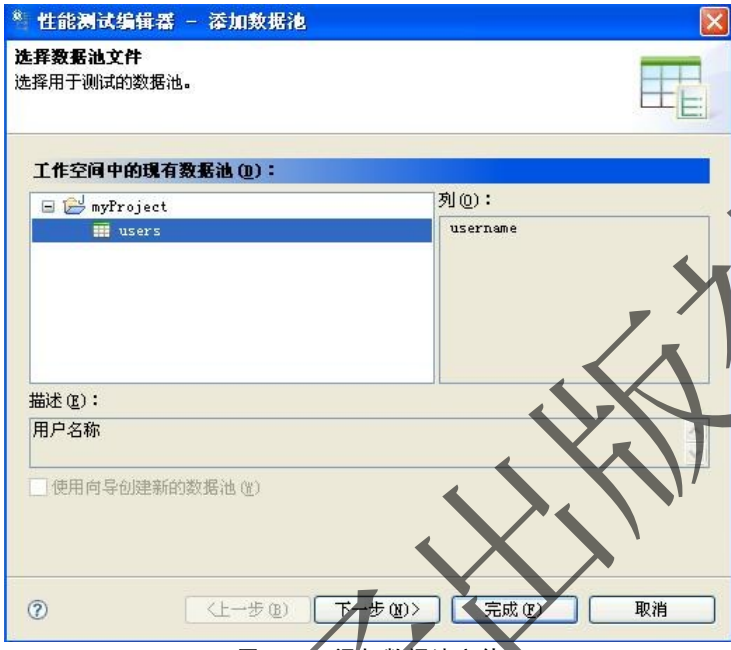


图 3-32　添加数据池文件

如图 3-33 所示，要求用户选择数据池的访问方式。提供了三种，分别是：共享(每台机器)、专用、分段(每台机器)。这里使用默认的共享方式，点击“完成”按钮。

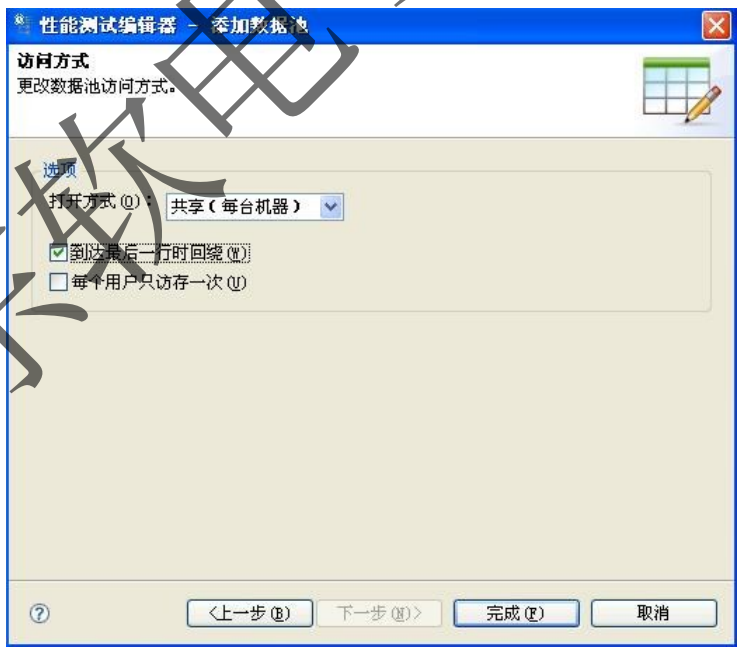


图 3-33　数据池的访问方式

添加完成后，在测试内容中找到需要替换数据的测试步骤，本例中是输入用户名和密码的

步骤“Find a Flight：Mercury Tours”，选中该步骤，在右侧“测试数据”中，选中名称为userName 的记录，然后单击“数据池变量”，如图 3-34 所示。

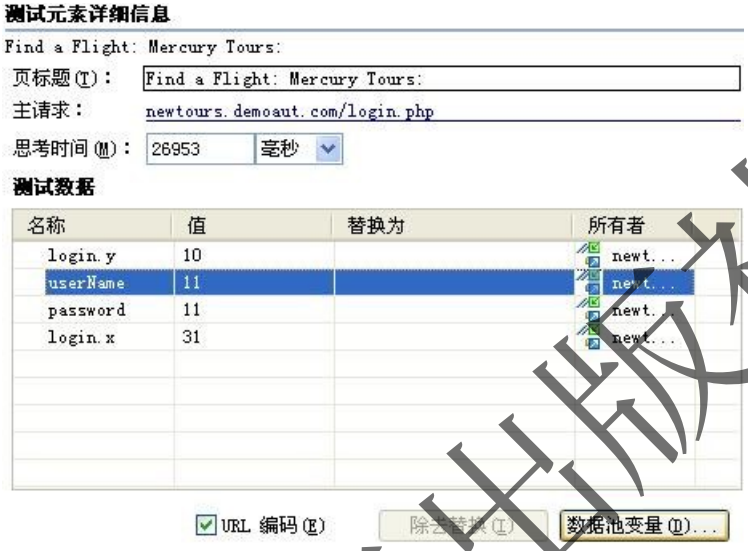


图 3-34　选择数据池变量

弹出“选择数据池列”窗口，选择 username 数据池，然后选择 username 列，单击“完成”，数据替换完成。如图 3-35 所示。

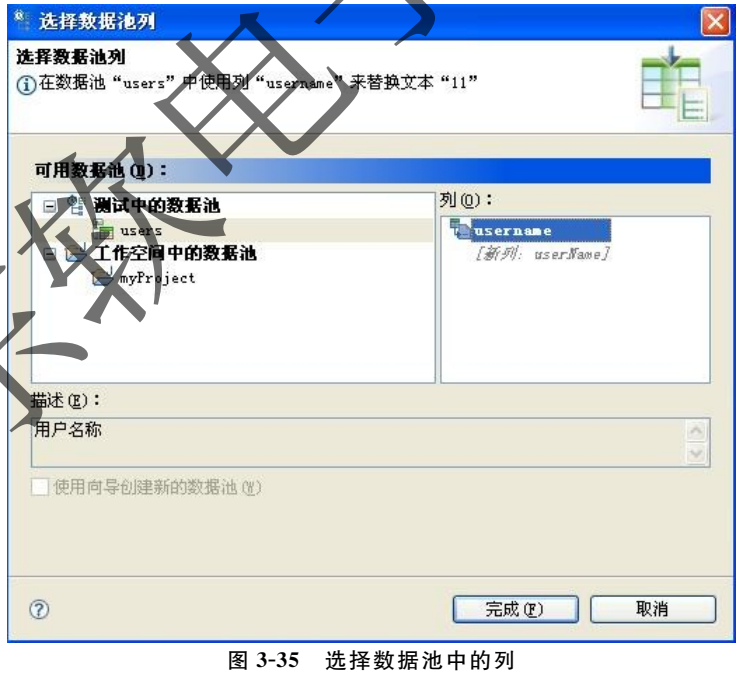


图 3-35　选择数据池中的列

替换完成后，在“测试数据”中就可以看到 userName 对应的记录中“替换为”字段显示了数据池的列。根据其“所有者”，在测试内容中找到该步骤，可以在对应的“数据”部分，看到