文章编号:1671-4598(2022)08-0031-07 DOI:10.16526/j.cnki.11-4762/tp.2022.08.006 中图分类号:TP202 文献标识码:A

一种标准化的软件性能测试流程设计与研究

韦亚军，张文文
（南京国图信息产业有限公司，南京 210000）

摘要：为了解决软件性能测试过程中测试效率低、测试过程复杂度高、性能瓶颈定位难等问题，文章借助常用的性能测试工具LoadRunner，提出了一种基于LoadRunner的标准化软件性能测试流程，并对其中关键难点分析研究，确定并发用户数、吞吐量等常用性能指标的计算方法；同时，将该流程应用到实际的项目案例中，应用结果表明，该流程不仅将测试过程进一步简易化、标准化，还解决了过去在软件性能测试过程中存在较多冗余步骤、测试不规范的问题，有效地降低了测试难度，提高了测试效率。

关键词：软件测试；性能测试；测试流程；性能测试模型；测试场景；LoadRunner

Design and Research of a Standardized Software Performance Testing Flow

WEI Yajun，ZHANG Wenwen
（Nanjing Guotu Information Industry Co.，Ltd.，Nanjing 210000，China）

Abstract：During the software performance testing，in order to solve the problems of low testing efficiency，high complexity in the testing process and difficulty in locating performance bottlenecks. With the help of the common performance testing tool LoadRunner，a standardized software performance testing process based on the LoadRunner is proposed，the key difficulties are analyzed and studied，the calculation methods for the common performance indicators such as the number of concurrent users and throughput are determined. At the same time，the process is applied to the actual project. The application results show that the process does not only further simplifies and standardizes the testing process，but also solves the problems of many redundant steps and non-standard testing in the past software performance testing，which effectively reduces the difficulty of testing and improves the efficiency of testing.

Keywords：software testing；performance testing；test flow；performance testing model；test scenario；LoadRunner

0 引言

近年来，随着计算机网络化应用的进一步普及，大数据、云计算、区块链等技术快速发展和应用，软件环境变得极其复杂，通常由网络、数据库、负载均衡器、应用服务（中间件）、硬件等多个部分组成，任何一部分都有可能成为整个应用软件的性能瓶颈，从而导致软件整体性能低下。同时，软件业务复杂度日益提升、用户量急剧增加、操作行为多样化使得保证软件性能的过程变得极为困难，而如何对软件性能进行测试和验证，一直以来都缺少标准化的流程和方法[1-3]。

针对这一现状，本文借助常用的性能测试工具LoadRunner，提出一种基于LoadRunner的软件性能测试标准化流程，并对其中关键技术分析研究，以对软件性能测试过程和方法给出指导。同时，结合淮安市“互联网+不动产登记”一体化平台项目对流程进行应用和验证，结果表明，该流程有效地剔除测试过程中的冗余步骤，提高测试效率，不仅满足大部分企业级软件的性能测试和验证，还可以指导测试人员快速准确地定位软件的性能瓶颈，分析存在的问题。

1 软件性能测试

软件性能测试是质量保证过程必不可少的环节，是验证软件是否满足用户期望的性能需求，发现软件可能存在的性能瓶颈，分析性能问题，并提供性能优化方案的过程。在实际测试过程中，可以根据具体的性能需求和应用领域采取不同的测试方法，完成对软件的性能需求和应用领域验证。常用的性能测试方法主要包括以下几种[4-8]：

1）验收性能测试：模拟实际业务量和使用场景，测试软件性能是否满足实际生产的性能要求，即在特定运行条件下验证软件的能力状况，是最常见的一种性能测试方法。

2）负载测试：在一定的软硬件环境下，利用增加虚拟用户数实现对被测软件增加负载，直到规定的性能指标或各项资源利用率超过预期目标。该方法可以找到软件的最大承载能力，了解系统的性能容量。

3）压力测试：在服务器CPU、内存处于饱和状态下，测试软件处理会话的能力、是否会出错等。一般通过模拟负载实现服务器资源达到较高水平，测试软件的稳定性。

收稿日期:2022-02-08； **修回日期**:2022-03-11。

作者简介:韦亚军(1990-)，男，安徽安庆人，硕士，工程师，主要从事大数据处理与软件自动化测试技术方向的研究。
通讯作者:张文文(1987-)，女，江苏南通人，大学本科，工程师，主要从事自然资源信息化建设方向的研究。

引用格式:韦亚军，张文文.一种标准化的软件性能测试流程设计与研究[J].计算机测量与控制，2022，30(8)：31-37.

4）配置测试：通过不断对软硬件的各项配置参数进行调整、测试、验证，掌握软件在不同环境中的运行效率，从而获得最优的参数配置。可以用于探索软件最大的性能潜力。

5）并发测试：测试较多用户对软件的同一个功能、同一个接口请求进行访问时，软件是否能正确响应。主要验证软件是否存在内存泄露、线程锁或资源竞争等情况。

6）可靠性测试：检验软件在一定的负载压力下，是否可以持续、稳定的运行。测试时间 2～3 天，测试过程需随时关注软件各项性能指标、资源使用情况是否满足要求。

7）失效恢复测试：主要针对采用了负载均衡、集群部署、分布式软件设计，当其中一台或者多台服务发生故障时，软件是否能正常被使用，且各项性能指标是否都满足要求。

2 LoadRunner 简介

LoadRunner 是一款广泛应用于企业级软件的性能测试工具，拥有良好的操作界面，可以通过模拟多虚拟用户对软件产生负载压力。同时，还可以与开发工具进行持续集成，实时监控软件性能测试过程中各项指标的运行情况，支持广泛的协议和编程技术，适用于多种体系架构的软件性能测试[9-11]。LoadRunner 主要包括用户脚本生成器、控制器和分析器三大组件[12]，其工作原理如图 1 所示。

1）用户脚本生成器（Vugen，virtual user generator），用于捕捉用户业务操作流并生成脚本。脚本记录了用户的操作，同时包含一系列用来监测软件性能的函数，用户可以对生成的脚本进行修改完善，来满足实际性能测试工作的要求。

2）控制器（Controller），是整个软件性能测试的控制中心，用来构造和维护性能测试场景，管理协调虚拟用户在系统上的操作行为。同时，通过控制器将任务分派给负载生成器，并收集 Vuser、服务器、软件各项性能指标数据。

3）分析器（Analysis），将控制器收集的各项性能指标数据以图表的形式进行展示并生成测试报告，供测试人员分析和性能调优。

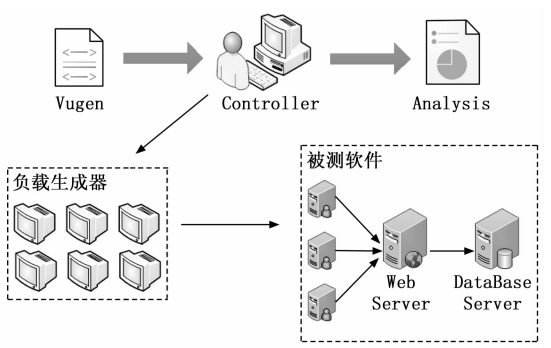

图 1 LoadRunner 工作原理

3 软件性能测试流程

基于 LoadRunner 的软件性能测试流程结合大部分软件应用现状，将测试流程分为规划测试工程、设计测试工程和测试结果分析 3 个阶段，每个阶段细分多个工作流程。实现将复杂的测试过程简单化、流程化，降低测试难度，提高测试效率，为软件的性能测试过程和方法提供指导。基于 LoadRunner 的软件性能测试流程如图 2 所示。

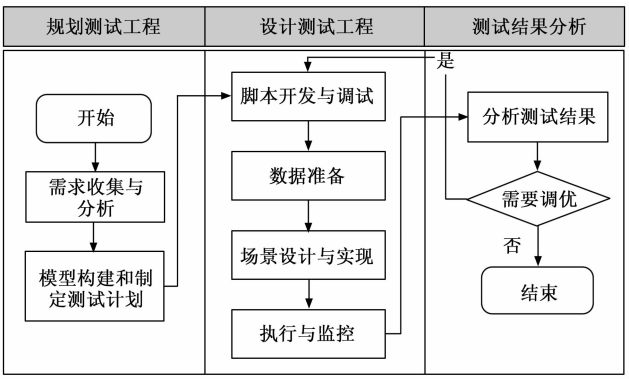

图 2 基于 LoadRunner 的软件性能测试流程

3.1 规划测试工程

软件性能测试通常是利用工具模拟大量用户操作来验证软件能够承受的负载情况，找出潜在的性能问题，并对问题进行分析、解决，是一项复杂度高、综合性强的工作，可以将其定义为一项测试工程，在开始阶段需对工程进行规划，主要包括收集并分析性能需求、构建性能模型和制定测试计划。

3.1.1 需求收集与分析

需求收集与分析主要是了解被测软件的系统架构、业务流程、用户特征，以及软件的运行环境，确定软件性能测试的目的和范围，分析哪些业务纳入性能测试范围以及用户期望的性能指标或测试终止标准，形成需求收集表。同时，还需要分析用户使用行为、业务分布、业务量统计，并根据分析结果估算出 TPS 和并发用户数等指标。

以淮安市“互联网＋不动产登记”一体化平台项目为例，该平台主要是面向全市 450 余万用户提供不动产登记业务线上办理服务。业务需求收集示例表格对部分业务量、未来业务量、响应时间、事务通过率等进行了收集，如表 1 所示。

表 1 业务需求收集示例

业务名称（描述）	业务量/（万笔/年）	未来业务量/（万笔/年）	响应时间/s	事务通过率/%
用户注册	100	120	＜3	＞99
用户登入	500	800	＜3	＞99
进度查询	50	80	＜3	＞99
指南查询	50	80	＜3	＞99
在线申请	20	30	＜3	＞95
在线预约	40	50	＜3	＞95

3.1.2 模型构建和制定测试计划

在软件性能测试过程中，构建性能模型类似编写测试

用例，主要是根据收集的性能需求和应用软件的相关信息完成模型的构建，指导性能测试的实现和测试结果的输出。软件性能模型构建主要包括业务模型和测试模型[13-20]。

1）业务模型（BM，business model），指经过性能需求分析后，明确软件的各个业务流程、业务相关功能在某个时间段内运行的业务种类及其业务占比，即哪些业务在哪个时间段运行、运行多久、业务量多少、并发用户多少等。业务模型是构建测试模型的重要基础，是性能测试关注的重点。

2）测试模型（TM，test model），从业务模型中分析和整理出来需要进行性能测试的业务，通常都是高频业务、高资源占用的业务，这些业务需要具有可测性、可验证性。测试模型作为性能测试场景设计的依据，通常会继承业务模型的大多数业务功能，针对因特殊原因无法进行性能测试的（如第三方非开源加密程序，测试程序无法模拟），测试模型中会去掉这部分业务，或者设计替代的等价方案。以淮安市“互联网＋不动产登记”一体化平台项目为例，部分测试模型如图 3 所示。

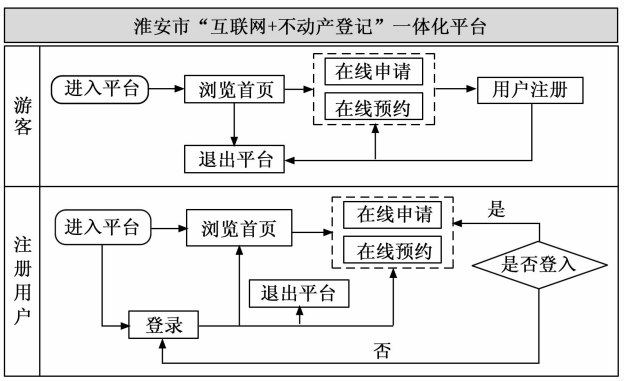

图 3　测试模型示例

制定测试计划的主要目的是用来规划软件性能测试工作，保证测试工作高效顺利的进行，同时防范和抵御可能出现的各类风险，是正式开展性能测试工作的前提。测试计划主要包括系统概述、测试环境、测试策略、测试场景、进度计划等内容。同时还需明确人员安排、系统风险，对测试过程中可能涉及的风险加以评估，确定风险的应对策略，如人员风险可以通过加强人力储备、采用 AB 角色、明确各自职责等来完成。

3.2　设计测试工程

设计性能测试工程主要是利用 LoadRunner 组件完成测试脚本开发、数据准备、场景设计、执行与监控等工作，是基于 LoadRunner 的软件性能测试流程的重要环节。该阶段的目标是对测试工程的进一步实现和深化，同时输出软件各项指标记录和报告，为后续性能调优提供数据基础。

3.2.1　脚本开发与调试

LoadRunner 提供了用户脚本生成器（Vugen）组件，支持 Web（HTTP/HTML）、Flex、SAPGUI 等多种主流协议和 JAVA、C 语言、VB 等多种编程语言进行测试脚本开发，可以根据系统架构设计选择对应的协议，参考构建的性能模型完成脚本开发。开发完成后，需对脚本进行回放、调试，同时还可以添加集合点、参数化使脚本更加接近真实用户操作，也可以插入事务和检查点来判断脚本在执行过程中系统是否能正确响应。

以用户登入业务脚本为例，在脚本上方插入集合点 lr _ rendezvous（“login”），可以实现不同虚拟用户在同一时间点执行登入操作，实现真正意义上的并发。在脚本上方添加开始事务 lr _ start _ transaction（“PC _ login”），在操作结尾添加结束事务和检查点，根据响应码判断事务成功或失败，通过该操作可以输出在指定压力测试下登入操作的成功率、平均响应时间等性能指标。

```
    lr_rendezvous("login");
      lr_start_transaction("PC_login");
      web_reg_save_param("access_token","LB=\"access_token
\"\:\"","RB\=\"\,",LAST);
      web_custom_request("newLogin",
      "URL=http://localhost/estateplat-register/v2/loginModel/
newLogin",
      "Method=POST",
      "Resource=0",
      "RecContentType=application/json",
      "Referer=http://localhost/estateplat-register/page/v2.1/
login.html",
      "Snapshot=t2.inf",
      "Mode=HTML",
      "EncType=application/json;charset=UTF-8",
      "Body={\"head\":{\"origin\":\"2\",\"sign\":\"<sign>
\",\"xzqydm\":\"\",\"access_token\":\"\"},"
      "\"data\":{\"lxDh\":\"<lxdh>\",\"userPwd\":\"\",\"
yzm\":\"123456\",\"loginType\":\"person\"}}",
      LAST);
    web_reg_find("Search=Body","SaveCount=pass_login","
Text=个人中心",LAST);
      web_url("new_user_index_个人中心",
      "URL=http://localhost/estateplat-register/page/v2.1/new
_user_index",
      "Resource=0",
      "RecContentType=text/html",
      "Referer=http://localhost/estateplat-register/page/v2.1/
login",
      "Snapshot=t9.inf",
      "Mode=HTML",
      if(web_get_int_property(HTTP_INFO_RETURN_CODE)
  200){

      lr_end_transaction("PC_login",LR_PASS);
      }
      else{
      lr_end_transaction("PC_login",LR_FAIL);
      }
```

3.2.2 数据准备

如何生成满足软件性能测试需要的大批量测试数据非常重要，因为在软件的实际使用过程中，用户可能会输入各种各样的数据进行操作，LoadRunner 提供了多种数据生成方法，主要包括如下几种：

1）数据生成池技术。在 LoadRunner 参数列表中，提供了多种数据类型生成技术，包括日期型、随机数等，在测试脚本中插入定义的参数名或函数，并设置参数循环方式，在脚本回放时就可以自动从数据池取出数据，完成大批量数据测试工作[21-25]。

2）自定义数据文件。用户可以将自己定义的数据文件（*.dat 格式）保存在脚本文件夹下，并对其进行参数化，添加到脚本中。在脚本回放时就可以通过参数化读取用户自定义数据文件中的测试数据。

3）执行 SQL 脚本文件。通过 LoadRunner 提供的数据库向导功能，编写、执行 SQL 语句从指定的数据库中批量选择出满足要求的测试数据并存储在本地 *.dat 文件中，该方法是一种比较高效的测试数据批量生成方法。

在本文项目中，采用第三种方法生成测试数据，即通过执行 SQL 语句，从数据库中获取足够多的且符合要求的测试数据，并保存至本地 *.dat 文件，为测试过程做数据准备。当数据库中的数据量不足以满足性能测试要求时，可以通过执行自动化脚本或编写插入数据 SQL 等方式批量添加数据，当性能测试结束后，需对本轮测试所产生的数据进行清洗或重新恢复数据库。

3.2.3 场景设计与实现

性能测试场景设计与实现是根据已构建的软件性能模型设计出相应的测试场景，并利用 LoadRunner 控制器（Controller）组件，完成测试场景的实现。

1）场景设计：在构建软件性能模型时已经确定了需要测试的业务种类，场景设计则是结合虚拟用户，将这些业务种类组合到一个测试单元。以淮安市“互联网＋不动产登记”一体化平台项目为例，根据收集的需求和构建的性能模型，设计出部分业务场景示例，如表 2 所示。

表 2 场景设计示例

场景编号	测试类型	业务名称	并发数	运行时间	目的
Sec_01	基准测试	用户注册	1	N/A	验证脚本、测试环境、性能基准
		用户登入	1		
		进度查询	1		
		指南查询	1		
Sec_02	配置测试	用户注册	20	10 min	优化配置
		用户登入	200		
		进度查询	10		
		指南查询	10		
Sec_03	负载测试	用户注册	20/40/60	30 min	分析性能变化趋势、分析性能问题、定容定量
		用户登入	200/400/600		
		进度查询	10/20/40		
		指南查询	10/20/40		

Sec _ 01 基准测试：主要是用来验证测试环境、测试脚本的正确性、得到软件的性能基准，为后续的测试执行提供参考依据。基准测试采用单业务场景、单用户方式来执行，执行时长根据响应时间来调整，测试结果采样样本尽量大。如响应时间为 1 s，1 000 个事务就需要运行 1 000 s 以上；响应时间 200 ms，运行 600 s 就可以完成 3 000 个事务的采样。

Sec _ 02 配置测试：帮助分析软件相关性能配置，确保软件配置适合于当前性能需求，一般为综合场景（多个业务同时执行）。测试过程是一个反复实验的过程，先找出不合理的配置，然后进行修改、验证，直到配置满足要求为止。

Sec _ 03 负载测试：负载测试的目的是找出软件的性能问题，并对软件进行定容定量，分析软件性能变化趋势，为软件优化、性能调优提供数据支撑。负载测试分为单场景和综合场景，单场景可以排除其他业务场景的干扰，有利于分析性能问题；综合场景更接近用户实际使用情况，可以对软件进行综合的性能评估。

2）场景实现：性能测试场景设计完成后，就可以利用 Controller 组件来实现具体场景。Controller 支持手动场景（manual scenario）和面向目标的场景（goal-oriented scenario）两种类型，手动场景是指完全由用户设置每个脚本的虚拟用户、运行时间等信息；面向目标的场景则是用户提前设定好性能测试要达到的目标和虚拟用户的增长模式，场景运行后，直到达到用户设定的目标后停止。

以 Sec _ 02 配置测试为例，选择手动场景，在 Controller 中添加测试脚本，设置每个脚本的虚拟用户数量和行为方式，完成配置测试场景的实现。如图 4 所示。

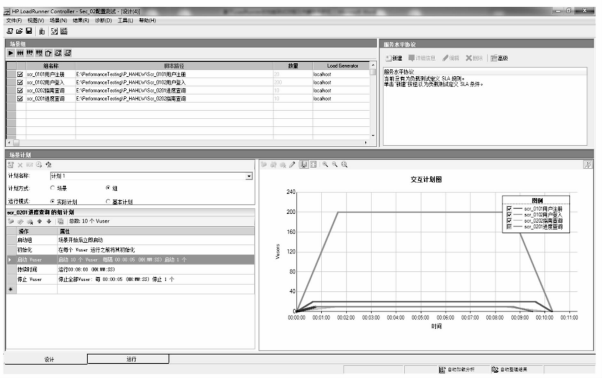

图 4 配置测试场景示例

3.2.4 执行与监控

场景设计与实现完成后，就可以在 Controller 中运行场景，对软件进行性能测试，同时，实时监控场景运行情况、获取虚拟用户的运行状态、收集测试结果。测试结果不仅包括平均事务响应时间、事务通过率、吞吐量等软件性能指标，还包括数据库性能状态、JVM 使用情况及服务器硬件性能指标，如 CPU 使用率、内存利用率、磁盘空间等，如表 3 所示。

表 3　性能监控项目

指标名称	阈值	指标说明
CPU 使用率	＜75％	过高会导致响应慢，服务不稳定
内存利用率	＜75％	过高会导致响应慢，服务不稳定
Disk Time	＜70％	过高会导致 IO 等待时间变长，服务水平降低
网络使用率	＜70％	过高会导致网络阻塞，网络延时变长，响应时间变长
慢查询 SQL		检查是否存在慢 SQL 和数据库资源状态
JVM GC		内存回收效率
Tomcat 相关		检查连接数、线程数、内存是否最优

3.3　测试结果分析

结果分析是测试流程中的一个重要组成部分。场景运行结束后，Analysis 组件会自动加载运行数据并生成各类图表信息，包括虚拟用户运行状态、吞吐量、响应时间及硬件资源等。同时，还提供了图表合并、分解、关联等功能，可以将多个性能指标数据关联后进行综合分析，并确定测试结果是否满足性能目标要求、是否需要进行性能调优等。

3.3.1　分析测试结果

性能测试结果分析是一项复杂度高且难度大的工作，需结合软硬件环境、测试报告进行综合分析，对测试人员综合能力要求较高，其分析步骤主要包括如下：

1）首先查看性能测试整体运行状况，收集运行日志信息，检查运行过程中是否存在异常报错，若存在，则需进一步分析并确定引起异常报错的具体原因。

2）其次分析性能测试执行结果，查看响应时间、吞吐量、事务通过率、资源利用率等指标是否满足性能要求，如图 5 所显示的是部分业务响应时间结果图。若不满足要求，则首先需对硬件环境进行排查，分析是否是硬件问题引起的性能瓶颈，硬件问题排除后，则需要对被测软件本身进行分析，确定引起性能问题的具体原因。

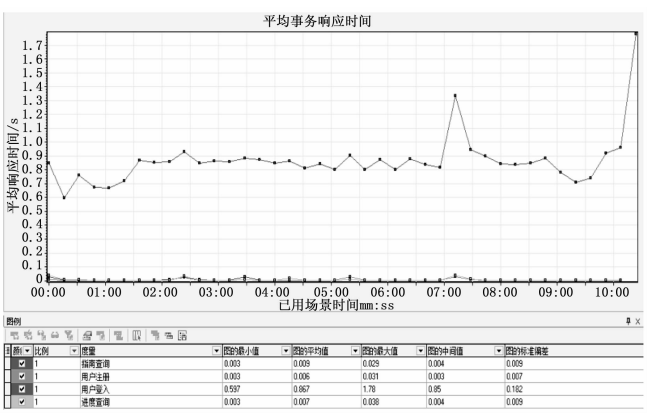

图 5　平均事务响应时间

3.3.2　性能调优

针对不满足性能需求的指标，需结合硬件、数据库及被测软件综合分析存在的问题，提出解决方案，并进行性能调优。通常性能调优的方法包括空间换时间、时间换空间、并行、异步处理等，其中异步处理是指当业务链路上有任务耗时较长，可以拆分业务，甚至使用异步的方式减少阻塞影响，也就是常说的解耦。当性能调优完成后，还需对测试脚本进行更新，以满足软件调优后的实际情况，同时对软件进行性能复测。

4　关键技术研究

在软件性能测试过程中，除了制定统一、标准化的测试流程外，还需对各个性能指标的度量进行统一，如并发用户数、每秒事务数、吞吐量等。同时，在工具的应用过程中，为了使得测试结果更加准确和有效，还应结合被测软件实际情况采用一些增强技术，如脚本参数化、脚本关联。

4.1　并发用户数

并发用户是指在软件中操作业务，并对后台服务器产生压力的用户，在性能测试工具中通常被称为虚拟用户（VU，virutal user）。并发用户数是由软件具体的业务场景来决定，因此，在计算并发用户数前，需首先确定典型的业务场景、具体的性能需求、业务模型等，然后再根据并发用户计算公式计算出每个业务场景的并发用户数[26-30]。

$$V_u = \frac{SL}{T} \tag{1}$$

$$\hat{V}_u = V_u + 3\sqrt{V_u} \tag{2}$$

在式（1）中，V_u 是指业务场景的并发用户数，S 是指该业务场景平均每天实际访问的用户量，L 是指用户从登入系统到退出的平均耗时，T 是指系统每天对外开放的总时长。式（2）是用来计算并发用户数峰值，其中，$\hat{V}_u$ 是指峰值，V_u 是由式（1）计算得到的并发用户数。

示例：以淮安市“互联网＋不动产登记”一体化平台项目为例，该软件注册用户数为 80 万，每天有 1.5 万用户访问软件进行各项操作，操作平均时间为 0.5 小时，该软件每天对外开放时间为 12 小时。则根据式（1）和式（2）可以得到：

$$V_u = \frac{(15000 \times 0.5)}{12} = 625$$

$$\hat{V}_u \approx 625 + 3 \times \sqrt{625} = 700$$

4.2　吞吐量

吞吐量是指单位时间内系统处理客户请求的数量，直接体现软件系统的性能承载能力。一般情况下，以不同计量单位来表达吞吐量的方式可以表明软件受不同层次的制约。如以 Bytes/Second 和 Pages/Second 为单位表示的吞吐量主要是受网络、服务器架构的制约；以 Requests/Second 为单位表示的吞吐量则主要是受应用服务器、软件代码逻辑的制约。

一般情况下，在软件未遇到性能瓶颈时，吞吐量与并发用户数之间存在一定的关联关系，如图 6（a）所示，可以采用式（3）计算吞吐量：

$$F_v = \frac{(V_u \cdot R)}{T} \tag{3}$$

其中：F_v 是指吞吐量，V_u 是指并发用户数，R 是指每个用户的实际请求数量，T 是指性能测试场景执行的时长。如果软件遇到性能瓶颈，如图 6（b）所示，吞吐量与并发用户数之间就不再存在关联关系，不能通过式（3）进行计算。

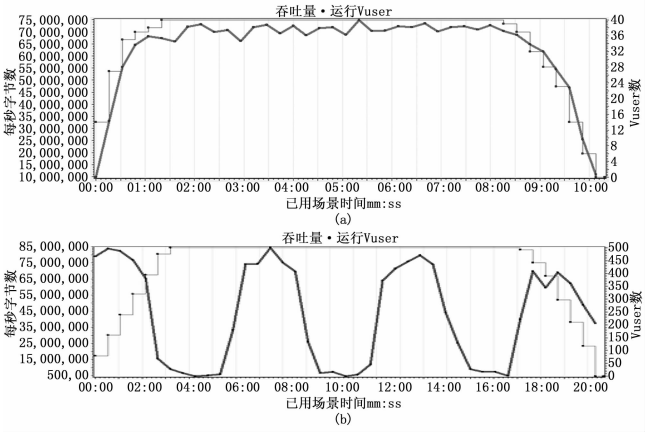

图 6 吞吐量－运行 Vuser

4.3 参数化

参数化是软件性能测试过程中最常用的技术之一。在脚本录制时，只是简单记录了一组固定从客户端发送到服务端的数据，这并不符合实际软件操作情况，同时也不满足软件性能测试大数据量的要求，因此就必须将这部分数据设为“参数”，并允许参数以某种方式和形式取值，在脚本运行时就可以通过预先设定的规则获取不同的值。

如图 7 所示，以办件进度查询业务为例。对脚本中 slbh＝2022012405262 中的受理编号进行参数化，修改为 slbh＝{slbh}，同时在数据池中添加多组不同的受理编号数据，并按照一定规则赋值给 slbh 参数。当脚本回放时，slbh 参数根据规则获取不同的数据执行脚本，实现进度查询业务的性能测试。

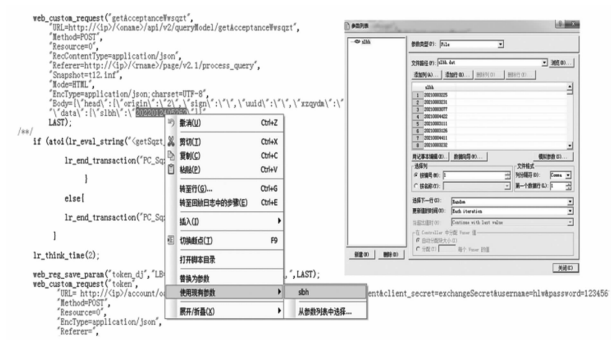

图 7 参数化示例

4.4 脚本关联

由于很多软件架构都采用 SessionID、SeqID 等方法来标识不同的任务和数据包，软件在每次运行时发送的数据并不完全相同。在这种情况下，为了让脚本能够满足测试的要求，就需要对脚本进行关联处理。

如图 8 所示，以用户登入业务为例。在脚本录制时，客户端首先向服务端发送 login 请求，服务端验证口令正确后，返回一个以某种规则生成的 SessionID，客户端将 sessionID＝375825QW7 作为 URL 的一部分提交服务端，服务端返回客户端相应的信息。在脚本回放时，客户端发送 login 请求后，服务端返回新的 SessionID，而此时由于测试脚本中的 SessionID 是写死的，客户端会提交过期的 SessionID，服务端验证失败后，返回会话失败。

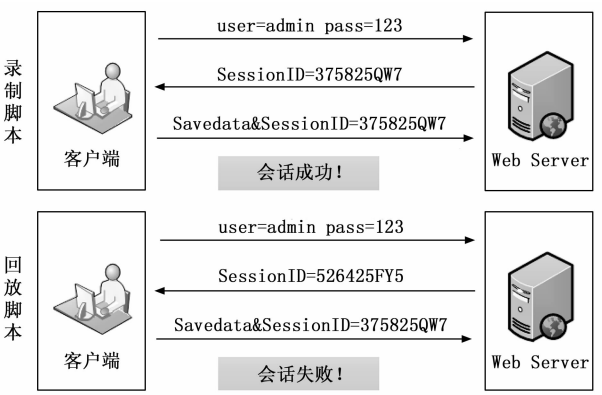

图 8 关联示例

LoadRunner 提供了手动和自动两种关联操作方式，并支持了一系列相关函数实现关联操作。结合上文，需要对脚本进行关联的条件是：客户端需要获取服务端返回的部分数据，并将这部分数据作为下一次请求的一部分发出。一般操作步骤如下：

1）从服务端返回的数据中选取需要进行关联的数据；

2）将该数据存入脚本的一个参数中；

3）将脚本中需要使用到该参数的地方用参数来替代。

5 实验结果与分析

依据该流程对淮安市“互联网＋不动产登记”一体化平台项目进行性能测试。以负载测试场景为例，对用户登入业务分别按照 200、400、600 并发用户进行负载测试，测试时间为 30 min，部分测试结果如表 4 所示。

表 4 负载测试用户登入业务部分测试结果

并发数	响应时间/s	吞吐量/(MB/s)	事务通过率/%	CPU 使用率/%	内存利用率/%
200	0.539	0.285	100	＜75	＜75
400	0.562	0.570	100	＜75	＜75
600	0.593	0.738	99.93	＜75	＜75

测试结果分析显示，针对该业务，随着并发用户的逐步增加，平均响应时间和事务通过率未发生明显的变化，且均满足目标要求。同时，系统吞吐量呈倍数递增趋势，表明系统在单位时间内处理客户请求的能力表现良好，且存在进一步提升的空间。在该场景下，应用服务器的 CPU 使用率、内存利用率均在阈值范围内，满足用户指标要求，如图 9 所示。

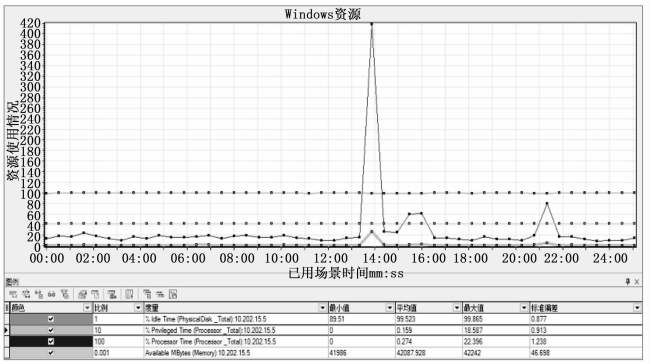

图 9　用户登入资源监控

在本项目中，利用该流程成功完成对淮安市“互联网＋不动产登记”一体化平台的性能测试工作，包括基准性能测试、稳定性测试、可靠性测试及失效恢复性测试四部分，并出具多份规范性测试报告。本次项目实际应用的结果，不仅获得用户一致认可，也进一步确定了该流程方法的可行性和正确性。

6　结束语

本文结合目前软件性能测试现状，借助常用的软件性能测试工具 LoadRunner，提出了一种标准化的软件性能测试流程。该流程将测试过程分为规划测试工程、设计测试工程和结果分析 3 个阶段，每个阶段对应不同的工作模块，如需求收集与分析、性能模型设计、场景设计与实现等。通过该流程，可以将复杂的测试过程简单化、标准化，有效地降低了测试难度，提高了测试效率。同时，本文还在实际项目中应用该流程，解决了过去在软件性能测试过程中存在较多冗余步骤、测试不规范等问题。最后，对流程中的关键难点分析研究，确定了并发用户数、吞吐量等常用性能指标的计算方法。

目前，该流程已广泛应用于不动产统一登记、自然资源一体化审批等相关软件的性能测试中，反响良好。未来将不断增加试点项目，优化流程设计，为各类企业的软件性能测试提供更多参考。

参考文献：

[1] 苏　璇. 基于 LoadRunner 的报表系统的性能测试 [J]. 通讯世界，2020 (6)：221－222.

[2] 陈梦云，高建华. 基于 LoadRunner 的理财平台的性能测试 [J]. 上海师范大学学报（自然科学版），2016 (4)：428－433.

[3] 杨　莹. 基于 Jmeter 的智能法律问答系统性能测试 [J]. 计算机测量与控制，2019，27 (12)：134－137.

[4] 朱少民. 软件测试课程的问题驱动教学模式探索 [J]. 中国大学教学，2018 (10)：32－36.

[5] 张　明，程宝雷，查伟忠，等. 面向安卓手机 App 功能测试技术的方法 [J]. 计算机工程与设计，2018 (3)：684－689.

[6] 辜萍萍. 软件测试课程实验教学体系设计与实践 [J]. 实验室研究与探索，2019 (7)：180－184.

[7] 冯兴利，范兆忠，罗军锋，等. 基于 Fiddler＋Loadrunner 的微信公众号自动化压力测试 [J]. 计算机应用，2018 (s2)：262－264.

[8] 张艳华. 基于 LoadRunner 的网络考试系统性能测试实践 [J]. 电脑知识与技术，2019 (21)：106－108.

[9] 刘小飞，李美满. 基于工作过程的软件测试课程实践教学改革 [J]. 计算机教育，2018 (6)：68－71.

[10] 陈　强，陈　双，吴立金，等. 分布式复杂系统软件测试建模方法与应用研究 [J]. 计算机测量与控制，2019，27 (2)：129－134.

[11] 雷　程，马多贺，张红旗，等. 基于最优路径跳变的网络移动目标防御技术 [J]. 通信学报，2017 (3)：133－143.

[12] 于　涌. 精通软件性能测试与 LoadRunner 最佳实战 [M]. 北京：人民邮电出版社，2013.

[13] 韩新宇，何　伟，张　凯. 基于 LoadRunner 自动化应用程序接口的舰船装备软件测试系统设计与实现 [J]. 计算机测量与控制，2016，24 (9)：163－166.

[14] 戴　乐. 基于 LoadRunner 的医疗云测试技术探索 [J]. 电信技术，2016 (12)：68－69，73.

[15] 张君栋，戴洪磊. 基于 LoadRunner 的 WebGIS 网站性能测试 [J]. 北京测绘，2016 (4)：23－27，37.

[16] 高金波，窦　文. LoadRunner 在 EJB 自动化测试中的应用 [J]. 计算机与数字工程，2006 (12)：132－134.

[17] 杨　萍，李　杰. 利用 LoadRunner 实现 Web 负载测试的自动化 [J]. 计算机技术与发展，2007 (1)：242－244.

[18] 钟珀辰，谭庆平，张　祥，等. 分布式压力测试工具的研究与设计 [J]. 电子技术与软件工程，2013 (18)：102－103.

[19] 刘　鸽. Web 压力测试及测试工具分析 [J]. 科技资讯，2015 (11)：22－23.

[20] 邱　亮. BIND 可靠性设计与压力测试技术 [J]. 中国科技信息，2016 (8)：86－90.

[21] 李　娟. 基于 LoadRunner 的 Web 系统性能测试的研究和应用 [J]. 信息技术与信息化，2019 (12)：226－228.

[22] 王兴野. 如何利用 LoadRunner 开展网站性能测试 [J]. 电脑编程技巧与维护，2018 (5)：100－102.

[23] 丰　丽. 基于 LoadRunner 的移动终端应用服务器性能测试脚本设计方法研究 [J]. 电脑与电信，2015 (Z1)：75－78.

[24] 邢承杰，宋式斌，林　莉，等. LoadRunner 在系统性能优化中的应用 [J]. 中山大学学报（自然科学版），2009 (s1)：301－304.

[25] 伊文斌，郑　剑. 基于 LoadRunner 的 Web 负载测试 [J]. 江西理工大学学报，2008 (4)：13－15.

[26] 李东昱，苗　放. LoadRunner 在 Web 应用程序性能测试中的应用 [J]. 软件导刊，2007 (19)：49－51.

[27] 董跃华，彭稷栋. 利用 LoadRunner 实现网页负载压力测试 [J]. 江西理工大学学报，2010 (5)：52－56.

[28] 姚庚梅. 移动网上商城中的压力测试应用 [J]. 科技信息，2010 (23)：505－506.

[29] 李素铎，马仲海. 浅谈网站测试的基本方法 [J]. 计算机时代，2008 (8)：14－15.

[30] 曾建国. 压力测试在真实环境下进行的意义——财务信息管理系统压力测试 [J]. 中国传媒科技，2015 (6)：76－77.