小试牛刀—完整实例带你探究 LR 性能测试

◆ 作者：姜林斌

一、性能测试理论知识

(ps: 我们先来了解下性能测试理论方面知识)

1.1、性能测试及其目的

性能测试的定义:

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。负载测试和压力测试都属于性能测试，两者可以结合进行。

性能测试的手段:

是通过模拟真实业务从而向服务器发送大量并发请求进而对被测系统产生负载，分析被测系统在不同压力下的表现。

我们进行性能测试的常见目的如下:

a: 评估系统的性能（在局域网测试环境或生产环境下，通过测试结果的分析评估当前系统的服务级别）。

b: 定位性能瓶颈（通过性能测试找出影响系统整体性能的关键步骤或过程，为系统调优提供方向性依据）。

c: 验证调优结果(通过比对优化后和优化前的测试结果，确认性能优化策略是否生效)。

1.2、性能测试的种类细分

1.2.1、压力测试:

通过逐步增加系统负载，测试系统性能的变化，并最终确定在什么负载条件下系统性能处于失效状态来获得系统能提供的最大服务级别的测试。

压力测试是逐步增加负载，使系统某些资源达到临界点。

1.2.2、负载测试:

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前提下，系统所能够承受的最大负载量的测试。

1.2.3、稳定性测试:

通过给系统加载一定的业务压力（如 CPU 资源在 70%～90%的使用率）的情况下，运行一段时间，检查系统是否稳定。因为运行时间较长，所以通常可以测试出系统是否有内存泄露等问题。

1.2.4、容量测试:

在一定的软、硬件条件下，在数据库中构造不同数量级的记录数量，通过运行一种或多种业务场景，在一定虚拟用户数量的情况下，获取不同数量级别的性能指标，从而得到数据库能够处理的最大会话能力、最大容量等。

1.2.5、配置测试:

通过对被测试软件的软硬件配置的测试。配置测试能充分利用有限的软硬件资源，发挥系统的最佳处理能力，同时可以将其与其他性能测试类型联合应用，为系统调优提供参考。

1.3、性能测试的实施流程

性能测试流程体系框架图

需求阶段	准备阶段	执行阶段	报告阶段	总结阶段
▶根据内部或者外部客户的需求，项目经理需进行可行性分析、进行项目组筹建。 ▶项目经理针对客户的系统，需召集相关的人员开项目启动会。 ▶项目经理组织人员进行详细的需求调研与分析，并通过广泛的评审，确定具体的需求范围。 ▶制定详细的测试和优化分析计划并通过广泛的评审	▶制定具体的性能测试技术方案，并评审通过。 ▶环境搭建准备，由客户方实施，并通过冒烟测试。 ▶测试人员进行脚本设计、数据准备，监控部署等。 ▶编写日报或者周报并汇报上级管理层，接收上级管理层的监管与指导。	▶按照测试计划中的测试内容规定项目执行各项测试 ▶详细记录测试工作步骤以及测试结果 ▶根据测试结果和性能跟踪数据监控数据进行系统瓶颈分析	▶编制《性能测试分析报告》汇报测试结果以及对系统性能的评估	▶编制《性能测试总结报告》，总结性能测试过程、方法、经验、教训等，改进工作流程的建议等

(PS:在实施性能测试的过程中，整体工作流程是 1:分析性能测试需求-->2:设计性能测试方案→3:开发性能测试脚本-→4: 搭建性能测试环境-→5: 执行测试-:6: 分析结果后多轮测试进行验证优化-→7: 编写性能测试报告-→8: 编写性能测试总结报告)

二、性能需求分析

(ps: 以我之前做过的一个小需求逐步开始吧~~)

目前公司开发人员 15 名，测试人员 7 名；使用 TeamFoundation 进行文档和测试用例以及 bug 的管理，团队考虑使用开源版禅道系统代替现有的 teamfoundation。

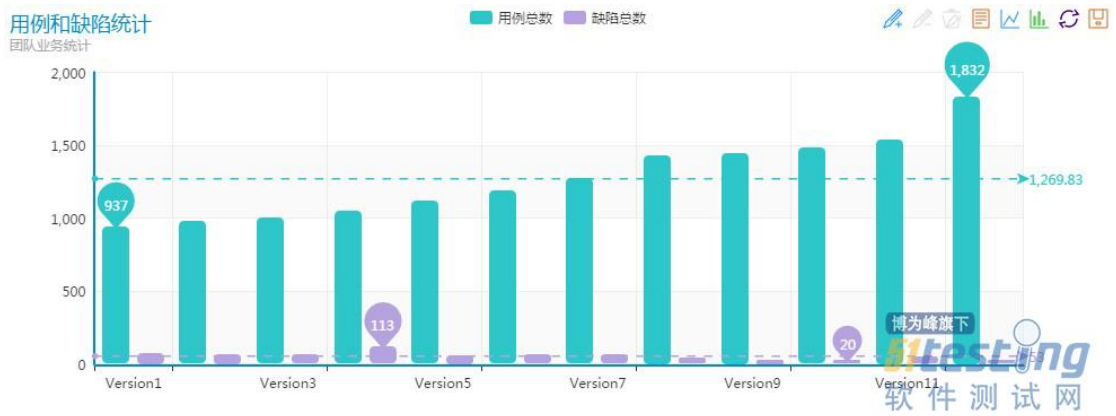
此次性能测试活动目标如下:

1: 能否使用禅道开源版(8.0.1)代替 TeamFoundation 进行项目活动管理。

2: 评估禅道开源版(8.0.1)在一定的服务器硬件配置(cpui53.3GHz+内存 8G)下的最大负载。

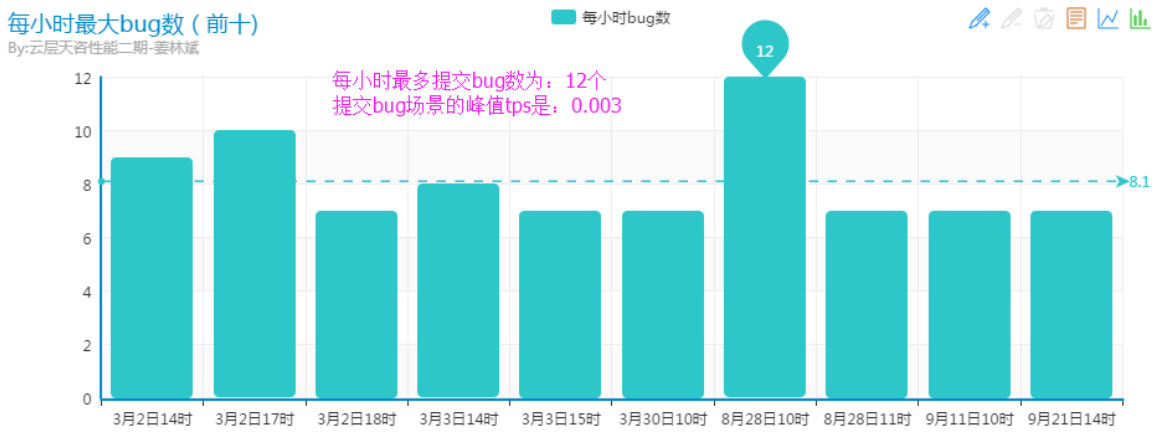
（ps: 分析团队历史数据）

- **每版本测试用例数统计:**

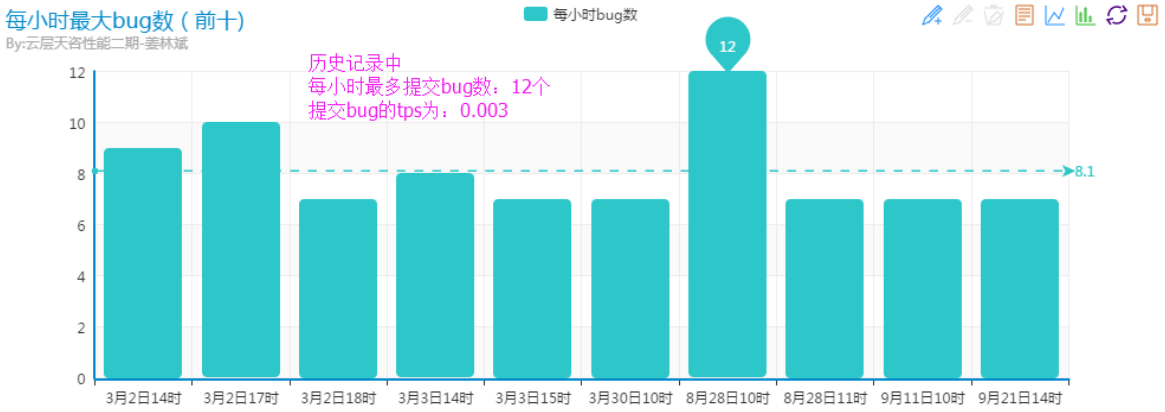


(ps: 在 12 个迭代版本中最多新建用例 1832 个)

- **单日提交 bug 数统计:**



- **每小时提交 Bug 数统计:**



- **以近一年来 12 个版本的数据进行分析，团队现状如下:**

版本最大 Case 数：1832 单个版本最大 Bug 数：113

每月 Release 一个新版本，即：每 22 个工作日进行一次发布。

在每次 releae 过程中编写测试用例的时间为：3 天执行测试的时间为 7 天(一轮回归测试)。

按峰值大致算出团队目前各场景的事务数如下:

团队目前各场景事务统计

场景名称	每日事务数	TPS
添加用例	1832/3日=610/日	0.0212
执行用例	1832/7日=261/日	0.0091
提交Bug	(峰值)12/3600s	0.003
解决Bug	同提交bug(ps:要求测试过程中bug日结)	0.003
关闭Bug	同提交bug(ps：要求测试过程中bug生命周期<=24小时)	0.003

(ps:测试过程中对峰值 tps 也需要留 20%的富余)

- 团队成员对业务及禅道环境的要求：

禅道开源版(8.0.1)负载测试需求

场景名称	响应时间	CPU使用率	内存使用率	磁盘读/写时间占比	事务成功率
登陆禅道	小于5s	小于85%	小于90%	小于90%	等于100%
添加用例					
执行用例					
提交Bug					
确认Bug					
解决Bug					
关闭Bug					

（ps：性能需求分析是性能测试流程中的第一步，如果这一步做好了接下来的测试方案设计，脚本开发，测试执行，测试报告都会轻松很多；反推也是成立的，如果不清楚需求是什么后面多的一切都是白做！

另外：收集需求数据的途径有 1：运维拉取生产环境的历史数据。2：参考竞品。3：对数据增量可以进行容量建模。

切记一点：千万别用什么所谓的二八原则，没有数据依据一切都是胡扯！）

三、性能测试方案设计

（ps：完整的性能测试方案直接拿去用吧~嘿嘿~）

3.1、测试目的、范围与目标

3.1.1、测试目的

本次性能测试的主要目的在于：

- 测试已完成系统的综合性能表现，检验交易或系统的处理能力是否满足系统运行的性能要求；
- 发现交易中存在的性能瓶颈，并对性能瓶颈进行修改；
- 模拟发生概率较高的单点故障，对系统得可靠性进行验证；

3.1.2、测试功能范围

序号	业务名称	优先级	备注
1	登录系统	中	
2	添加测试用例	高	
3	执行用例	高	
4	提交 Bug	高	
5	解决 Bug	高	
6	关闭 Bug	高	
8	确认 Bug	高	
9			
10			

3.1.3、测试指标范围

*/***明确列出说明本次测试需要关注的测试指标的定义及范围，不需要关注的测试指标也应列出。下面的内容供参考。***/*

本次性能测试需要获得的性能指标如下:

- 交易的响应能力：即在单交易负载和模拟生产交易情况的混合场景负载压力情况下，系统的响应时间。
- 每秒处理事务数：即应用系统在单位时间内完成的交易量（TPS）。
- 系统可支持的并发用户数量。

本次性能测试的限制性指标为:

- 系统资源使用情况：在正常压力下，应用服务器和数据库服务器的 CPU、Memory 占用率应分别低于 80%、80%，数据库存储空间和文件系统空间占用率应低于 80%。
- 交易的成功率：交易成功率不低于 99.5%。

本次性能测试不需要关注的指标:

- 业务流程/路径覆盖率。
- 业务数据的完整、正确性。
- 其他诸如系统易用性、可管理性等属于专项测试的内容。

3.1.4、测试目标

*///***明确本次测试各功能项的测试指标需要达到的测试目标，该目标须由项目组提出或最终确认。***///*

- 针对不同类型交易的单业务事务平均响应时间
- 针对不同类型交易的单业务事务 TPS 值
- 在负载情况下的单业务事务平均响应时间
- 在负载情况下的单业务事务 TPS 值
- 在负载情况下的系统综合 TPS 值

3.2、测试资源

3.2.1、系统生产环境物理架构

*///***说明本项目生产环境的物理架构，可以以物理架构图或网络拓扑图的方式。***///*

3.2.2、性能测试环境物理架构

*///*说明本项目性能测试环境的物理架构，可以以物理架构图的方式。*///*

可使用 Visio 画出测试环境和生产环境的网络拓扑图。

测试环境的网络拓扑图(单 Web 服务器+单数据库服务器)很简单在此省略。

3.2.3、性能测试环境与生产环境资源对比

说明本项目测试环境与生产环境的差异，确定性能测试环境的软硬件资源，包括待测系统各组成部分的配置。下表供参考，非强制使用。

服务器	性能测试环境（规划）		生产环境（规划）	
	硬件配置	软件配置及 IP	硬件配置	软件配置
Web&DB 服务器	Cpu:i5 Disk:500G Memory: 8G	192.168.10.206	Cpu:i3 Disk:500G Memory: 8G	Os:Win7(64 位) WebServer:Apache2.4 DB:MySql5.5 PHP:5.4.19
负载生成服务器	Cpu:i3 Disk:500G Memory:8G Net:100Mb 局域网	192.168.10.188 OS:Win7(64 位) LR11	--	--

3.3、测试准备

3.3.1、测试环境安装

*/*说明本次测试的测试环境安装情况。*/*

XAMPP 集成环境:

```
控制面板 1.2.6
phpmyadmin 版本: Version4.0.8
php 版本: PHP5.4.19(cli)(built:Aug21201301:12:03)
apache 版本: Serverversion:Apache/2.4.4(Win32)
```

mysql 版本：Ver5.5.32

负载机：LoadRunner11(patch3+patch4)(192.168.10.206)

3.3.2、测试工具

*/*说明本次测试使用到的测试工具和监控工具。*/*

浏览器：IE11，Chrome43

协议抓包：HttpWatch9.3

性能脚本：LoadRunner11

监控工具：MonyogMySQL 和 LoadRunner11Controller。

测试数据图表生成：ECharts

3.2.3、性能测试环境与生产环境资源对比

说明本项目测试环境与生产环境的差异，确定性能测试环境的软硬件资源，包括待测系统各组成部分的配置。下表供参考，非强制使用。

服务器	性能测试环境（规划）		生产环境（规划）	
	硬件配置	软件配置及 IP	硬件配置	软件配置
Web&DB 服务器	Cpu:i5 Disk:500G Memory：8G	192.168.10.206	Cpu:i3 Disk:500G Memory: 8G	Os:Win7(64 位) WebServer:Apache2.4 DB:MySql5.5 PHP:5.4.19
负载生成服务器	Cpu:i3 Disk:500G Memory:8G Net:100Mb 局域网	192.168.10.188 OS:Win7(64 位) LR11	--	--

3.3、测试准备

3.3.1、测试环境安装

/*说明本次测试的测试环境安装情况。*/

XAMPP 集成环境:

控制面板 1.2.6

phpmyadmin 版本：Version4.0.8

php 版本：PHP5.4.19(cli)(built:Aug21201301:12:03)

apache 版本：Serverversion:Apache/2.4.4(Win32)

mysql 版本：Ver5.5.32

负载机：LoadRunner11(patch3+patch4)(192.168.10.206)

3.3.2、测试工具

/*说明本次测试使用到的测试工具和监控工具。*/

浏览器：IE11，Chrome43

协议抓包：HttpWatch9.3

性能脚本：LoadRunner11

监控工具：MonyogMySQL 和 LoadRunner11Controller。

测试数据图表生成：ECharts

3.4、测试脚本、数据及其预验证

/**说明本次测试的测试脚本、测试数据以及混合场景的交易配比情况等。**/

3.4.1、基础测试数据

系统用户：开发角色的用户 40 个测试角色的用户 20 个。

```
SELECT COUNT( role ) , role
FROM zt_user
WHERE account != 'admin'
GROUP BY role
LIMIT 0 , 30
```

显示：起始行： 0 行数： 30 每 100 行重复表头

按索引排序： 无

+ 选项

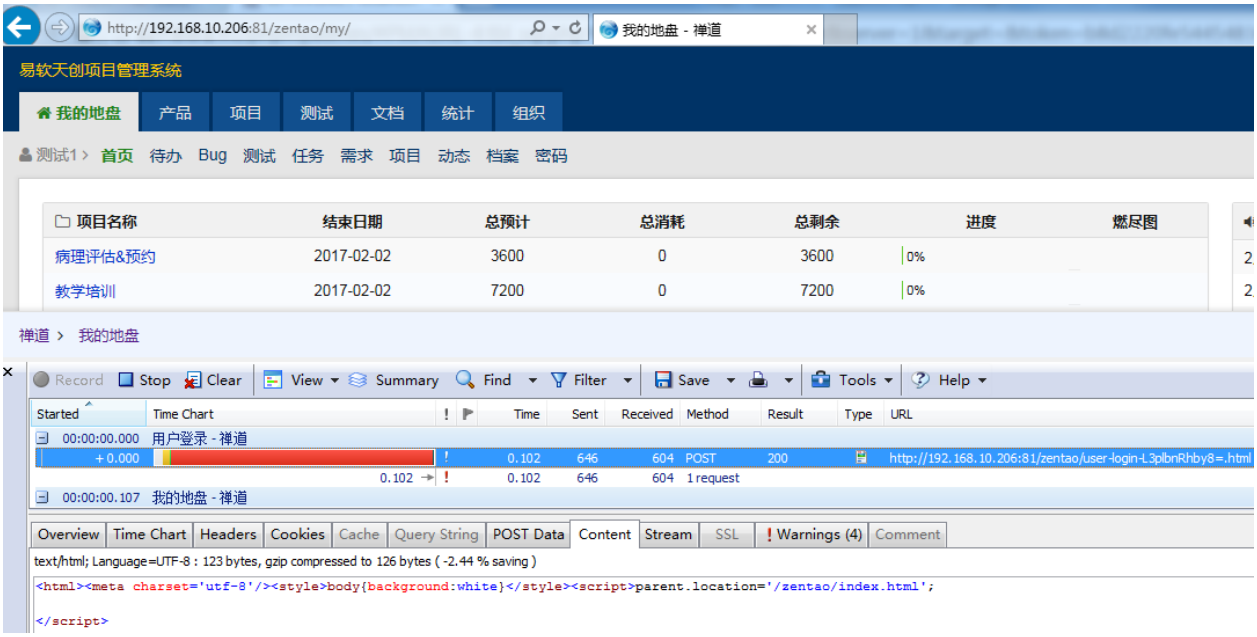
count(role)	role
40	dev
20	qa

项目：3 个项目(Storage,Training,Clinical)

3.4.2、脚本及其验证

Script1：登录禅道(zentao)脚本

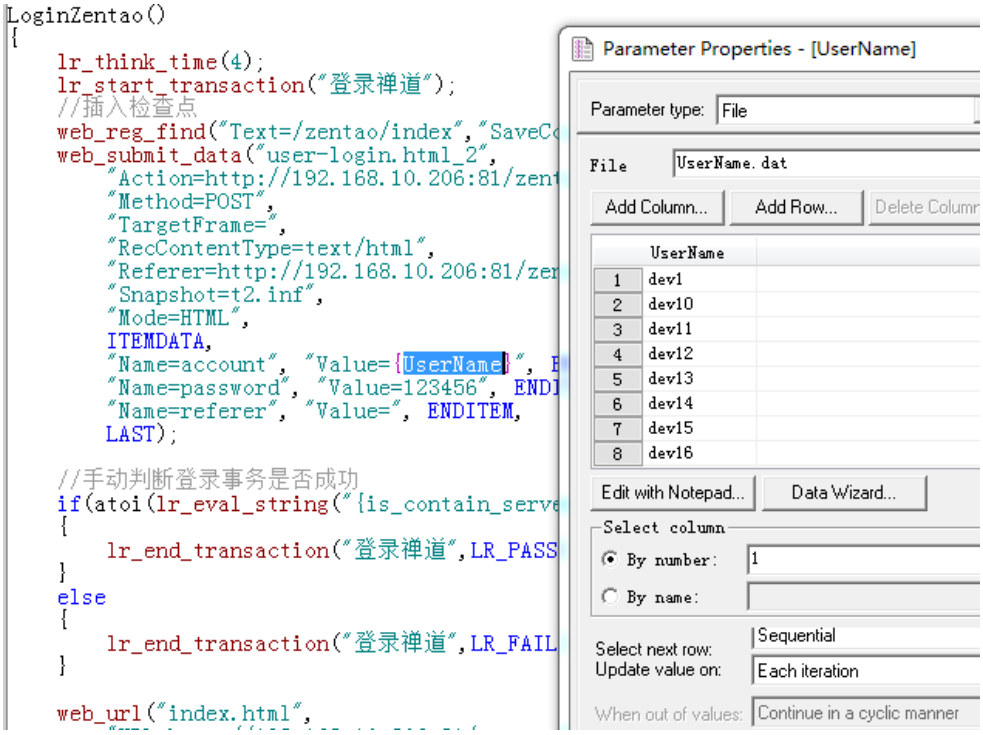
HttpWatch 抓去登录禅道的请求：



登录请求返回的内容中没有标识登录是否成功的字符

所以：插入检查点时检查登录请求中从服务器返回的内容里是否包括/zentao/index 字符串。

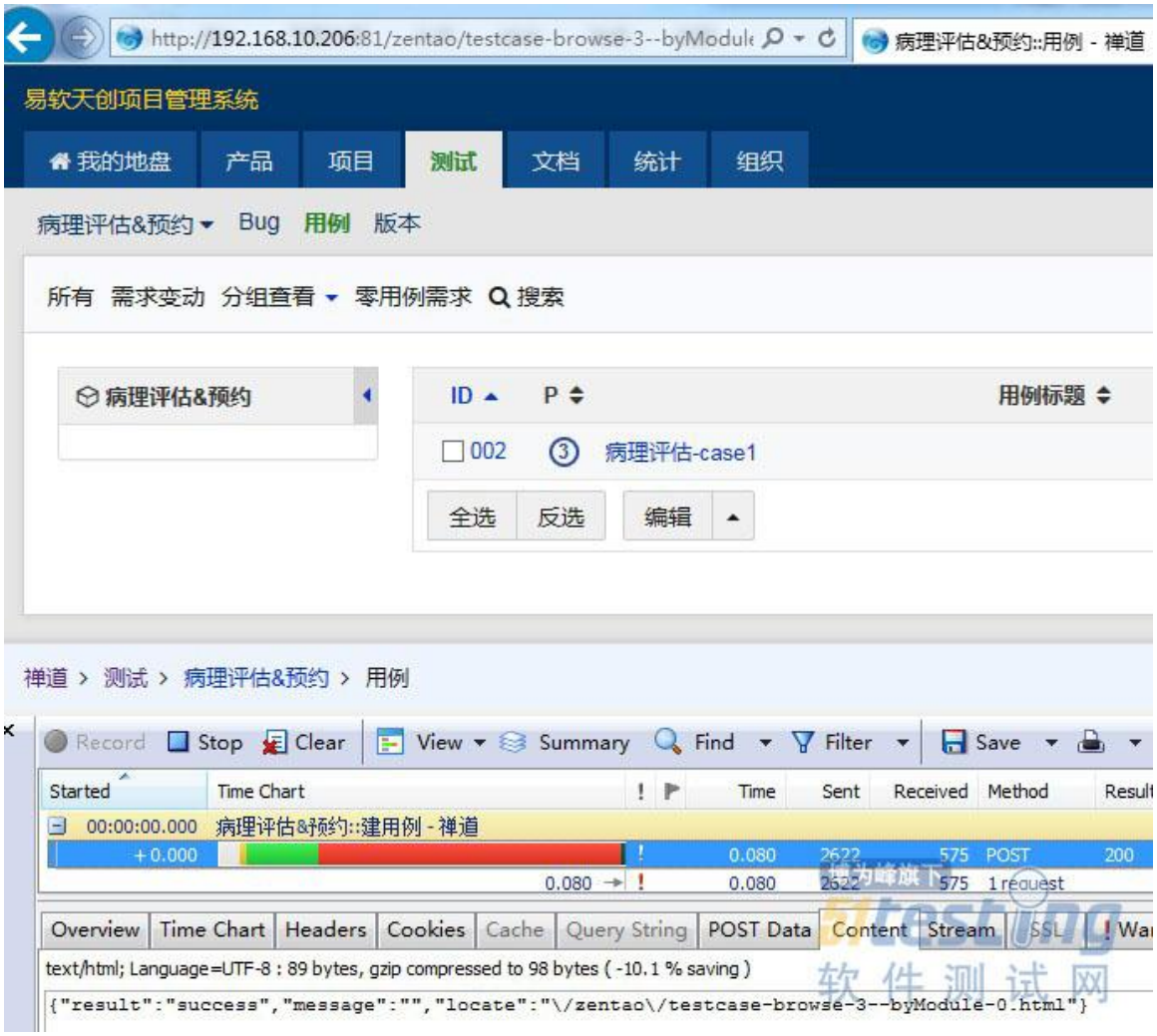
登录禅道事务中参数 UserName 取值为测试+开发用户名。




```
Action()
{
lr_think_time(4);
  lr_start_transaction("登录禅道");
  //插入检查点
  web_reg_find("Text=/zentao/index","SaveCount=is_contain_server",LAST);
  web_submit_data("user-login.html_2",
        "Action=http://192.168.10.188/zentao/user-login.html",
        "Method=POST",
        "TargetFrame=",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/user-login.html",
        "Snapshot=t2.inf",
        "Mode=HTML",
        ITEMDATA,
        "Name=account","Value={UserName}",ENDITEM,
```

```
        "Name=password","Value=123456",ENDITEM,
        "Name=referer","Value=",ENDITEM,
        LAST);

//手动判断登录事务是否成功
if(atoi(lr_eval_string("{is_contain_server}"))>0)
{
lr_end_transaction("登录禅道",LR_PASS);
}
else
{
lr_end_transaction("登录禅道",LR_FAIL);
}

  web_url("index.html",
        "URL=http://192.168.10.188/zentao/index.html",
        "TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/user-login.html",
        "Snapshot=t3.inf",
        "Mode=HTML",
        LAST);
  return0;
}
```

Script2： 添加用例



成功添加测试用例时会从服务器返回添加用例是否成功的标识

插入检查点，手动判断添加用例事务是否成功

```
Action()
{
//点击测试进入测试 Tab 页
  web_url("qa",
        "URL=http://192.168.10.188/zentao/qa/",
        "TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/my/",
        "Snapshot=t4.inf",
        "Mode=HTML",
        LAST);
```

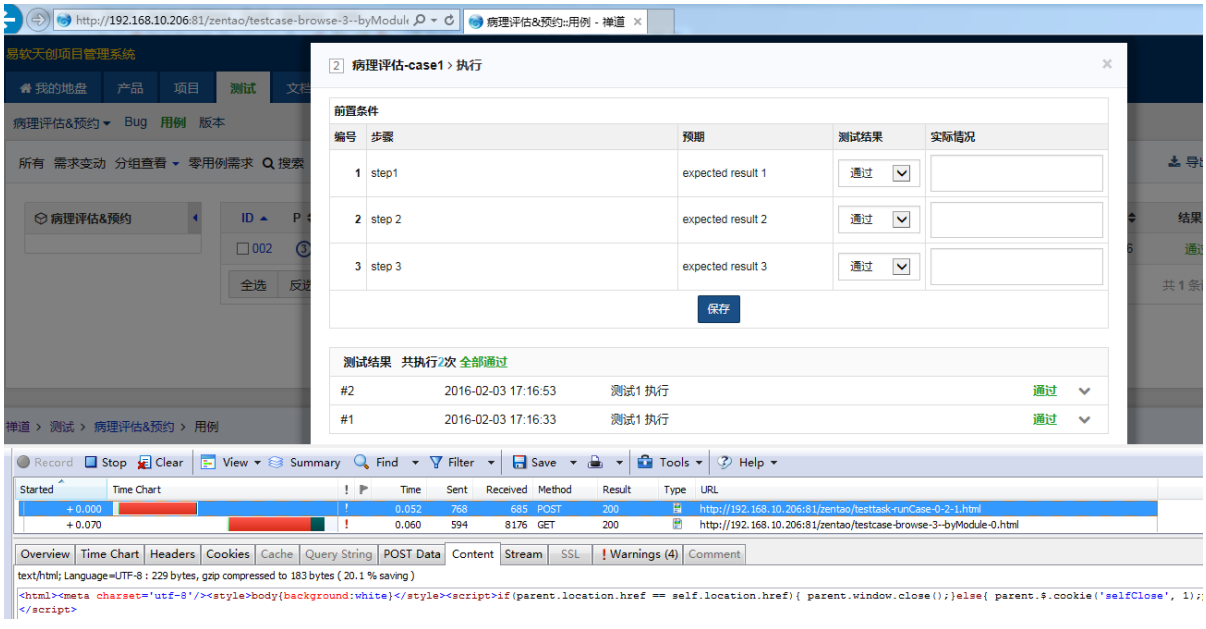
```
//点击用例进入用例页面
web_url("用例",
        "URL=http://192.168.10.188/zentao/testcase-browse-3.html",
        "TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/bug-browse.html",
        "Snapshot=t5.inf",
        "Mode=HTML",
        LAST);
//点击新建用例
web_url("建用例",
        "URL=http://192.168.10.188/zentao/testcase-create-3-0-0.html",
        "TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/testcase-browse-3.html",
        "Snapshot=t6.inf",
        "Mode=HTML",
        LAST);
web_url("story-ajaxGetProductStories-3-0-11-0-false-noclosed-50.html",
"URL=http://192.168.10.188/zentao/story-ajaxGetProductStories-3-0-11-0-false-noclosed-
50.html",
"TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/testcase-create-3-0-0.html",
        "Snapshot=t7.inf",
        "Mode=HTML",
        LAST);
lr_think_time(39);
//添加事务-添加测试用例
lr_start_transaction("添加测试用例");
```

```
//添加检查点
web_reg_find("Text=success","SaveCount=addcase_result",LAST);
  web_submit_data("testcase-create-3-0-0.html",
        "Action=http://192.168.10.188/zentao/testcase-create-3-0-0.html",
        "Method=POST",
        "EncType=multipart/form-data",
        "TargetFrame=",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/testcase-create-3-0-0.html",
        "Snapshot=t8.inf",
        "Mode=HTML",
        ITEMDATA,
        "Name=product","Value=3",ENDITEM,
        "Name=module","Value=11",ENDITEM,
        "Name=type","Value=feature",ENDITEM,
        "Name=stage[]","Value=",ENDITEM,
        "Name=stage[]","Value=system",ENDITEM,
        "Name=title","Value=testcase{CaseNum}",ENDITEM,
        "Name=pri","Value=3",ENDITEM,
        "Name=precondition","Value=",ENDITEM,
        "Name=steps[]","Value=step1",ENDITEM,
        "Name=expects[]","Value=expectresult{CaseNum}",ENDITEM,
        "Name=steps[]","Value=",ENDITEM,
        "Name=expects[]","Value=",ENDITEM,
        "Name=steps[]","Value=",ENDITEM,
        "Name=expects[]","Value=",ENDITEM,
        "Name=keywords","Value=",ENDITEM,
        "Name=files[]","Value=",ENDITEM,
        "Name=labels[]","Value=",ENDITEM,
        LAST);

//手动判断添加测试用例是否成功
if(atoi(lr_eval_string("{addcase_result}"))>0)
```

```
{
lr_end_transaction("添加测试用例",LR_PASS);
}
else
{
lr_end_transaction("添加测试用例",LR_FAIL);
}

  web_url("testcase-browse-3--byModule-11.html",
        "URL=http://192.168.10.188/zentao/testcase-browse-3--byModule-11.html",
        "TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/testcase-create-3-0-0.html",
        "Snapshot=t9.inf",
        "Mode=HTML",
        LAST);
  return0;
}
```

Script3：执行用例

HttpWatch 抓取执行用例请求：



执行用例的请求中从服务器端并未返回事务是否成功的标识

所以插入检查点验证是否返回 selfClose 来手动判断执行用例事务的成功与否

每个测试用例可多次执行无权限问题

```
Action()
{
  lr_think_time(8);
  //添加事务-执行测试用例
  lr_start_transaction("执行用例");
  //添加检查点
web_reg_find("Text=selfClose","SaveCount=executecase_result",LAST);

  web_submit_data("execuate_case",
        "Action=http://192.168.10.188/zentao/testtask-runCase{case_str}.html",
        "Method=POST",
        "TargetFrame=",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/testtask-runCase{case_str}.html",
        "Snapshot=t8.inf",
        "Mode=HTML",
        ITEMDATA,
        "Name=steps[5]","Value=pass",ENDITEM,
        "Name=reals[5]","Value=",ENDITEM,
        "Name=case","Value={CaseIndex}",ENDITEM,
        "Name=version","Value=1",ENDITEM,
        LAST);

  //手动判断事务是否成功
  if(atoi(lr_eval_string("{executecase_result}"))>0)
  {
        lr_end_transaction("执行用例",LR_PASS);
  }
  else
```

```
    {
        lr_end_transaction("执行用例",LR_FAIL);
    }

    web_url("testcase-browse-3.html",
        "URL=http://192.168.10.188/zentao/testcase-browse-3.html",
        "TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/testcase-browse-3.html",
        "Snapshot=t9.inf",
        "Mode=HTML",
        LAST);

    return0;
}
Script4:  提交 Bug
Action()
{
    lr_think_time(50);
    //添加事务-提交 Bug
    lr_start_transaction("提交 Bug");
    //添加检查点
    web_reg_find("Text=success","SaveCount=submitbug_result",LAST);
    web_submit_data("bug-create-3-0-moduleID=0.html_3",
        "Action=http://192.168.10.188/zentao/bug-create-3-0-moduleID=0.html",
        "Method=POST",
        "EncType=multipart/form-data",
        "TargetFrame=",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/bug-create-3-0-moduleID=0.html",
        "Snapshot=t15.inf",
        "Mode=HTML",
```

```
        ITEMDATA,
        "Name=product","Value=3",ENDITEM,
        "Name=module","Value=11",ENDITEM,
        "Name=project","Value=1",ENDITEM,
        "Name=openedBuild[]","Value=trunk",ENDITEM,
        "Name=assignedTo","Value=dev{DevNum}",ENDITEM,
        "Name=type","Value=codeerror",ENDITEM,
        "Name=os","Value=win7",ENDITEM,
        "Name=browser","Value=ie11",ENDITEM,
        "Name=title","Value=BugTitle{Num}",ENDITEM,
        "Name=severity","Value=3",ENDITEM,
        "Name=pri","Value=3",ENDITEM,
        "Name=steps","Value=<p>[步骤]</p>\r\n<p>[结果]</p>\r\n<p>[期望]</p>",ENDITEM,
        "Name=story","Value=",ENDITEM,
        "Name=task","Value=",ENDITEM,
        "Name=mailto[]","Value=",ENDITEM,
        "Name=keywords","Value=",ENDITEM,
        "Name=files[]","Value=",ENDITEM,
        "Name=labels[]","Value=",ENDITEM,
        "Name=case","Value=0",ENDITEM,
        "Name=caseVersion","Value=0",ENDITEM,
        "Name=result","Value=0",ENDITEM,
        "Name=testtask","Value=0",ENDITEM,
        LAST);

    //手动判断提交 Bug 是否成功
    if(atoi(lr_eval_string("{submitbug_result}"))>0)
    {
        lr_end_transaction("提交 Bug",LR_PASS);
    }
    else
    {
        lr_end_transaction("提交 Bug",LR_FAIL);
```

```
 }
 web_url("bug-browse-3.html",
       "URL=http://192.168.10.188/zentao/bug-browse-3.html",
       "TargetFrame=",
       "Resource=0",
       "RecContentType=text/html",
       "Referer=http://192.168.10.188/zentao/bug-create-3-0-moduleID=0.html",
       "Snapshot=t16.inf",
       "Mode=HTML",
       LAST);

 return0;
}
```

Script4: 解决 Bug

```
Action()
{
 web_url("bug-resolve-12.html",
       "URL=http://192.168.10.188/zentao/bug-resolve-12.html?onlybody=yes",
       "TargetFrame=",
       "Resource=0",
       "RecContentType=text/html",
       "Referer=http://192.168.10.188/zentao/bug-browse.html",
       "Snapshot=t19.inf",
       "Mode=HTML",
       EXTRARES,

 "Url=js/kindeditor/themes/default/default.png","Referer=http://192.168.10.188/zentao/js/kindeditor/themes/default/default.css",ENDITEM,
       "Url=theme/zui/css/min.css","Referer=http://192.168.10.188/zentao/bug-resolve-12.html?onlybody=yes",ENDITEM,
       LAST);

 lr_think_time(10);
```

```
//添加事务-解决 Bug
lr_start_transaction("解决 Bug");
//添加检查点
web_reg_find("Text=selfClose","SaveCount=fixbug_result",LAST);

web_submit_data("bug-resolve-12.html_2",
        "Action=http://192.168.10.188/zentao/bug-resolve-12.html?onlybody=yes",
        "Method=POST",
        "EncType=multipart/form-data",
        "TargetFrame=hiddenwin",
        "RecContentType=text/html",
        "Referer=http://192.168.10.188/zentao/bug-resolve-12.html?onlybody=yes",
        "Snapshot=t20.inf",
        "Mode=HTML",
        ITEMDATA,
        "Name=resolution","Value=fixed",ENDITEM,
        "Name=duplicateBug","Value=",ENDITEM,
        "Name=resolvedBuild","Value=trunk",ENDITEM,
        "Name=resolvedDate","Value={TimeNow}",ENDITEM,
        "Name=assignedTo","Value={TesterList}",ENDITEM,
        "Name=files[]","Value=","File=Yes",ENDITEM,
        "Name=labels[]","Value=",ENDITEM,
        "Name=comment","Value=已经解决请验证",ENDITEM,
        LAST);

//手动判断解决 Bug 事务是否成功
if(atoi(lr_eval_string("{fixbug_result}"))>0)
{
        lr_end_transaction("解决 Bug",LR_PASS);
}
else
{
        lr_end_transaction("解决 Bug",LR_FAIL);
```

```
 }


 web_url("bug-browse.html",
       "URL=http://192.168.10.188/zentao/bug-browse.html",
       "TargetFrame=",
       "Resource=0",
       "RecContentType=text/html",
       "Referer=http://192.168.10.188/zentao/bug-browse.html",
       "Snapshot=t21.inf",
       "Mode=HTML",
       LAST);
 return0;
}
```

Script5: 关闭 Bug

```
Action()
{
 web_url("bug-close.html",
       "URL=http://192.168.10.188/zentao/bug-close-{Bug_Index}.html?onlybody=yes",
       "TargetFrame=",
       "Resource=0",
       "RecContentType=text/html",
       "Referer=http://192.168.10.188/zentao/bug-browse-3-0-assignToMe-0.html",
       "Snapshot=t6.inf",
       "Mode=HTML",
       EXTRARES,

 "Url=js/kindeditor/themes/default/default.png","Referer=http://192.168.10.188/zentao/js/kindeditor/themes/default/default.css",ENDITEM,
       "Url=theme/zui/css/min.css","Referer=http://192.168.10.188/zentao/bug-close-{Bug_Index}.html?onlybody=yes",ENDITEM,
       LAST);


 lr_think_time(19);
```

```
//开始事务-关闭 Bug
lr_start_transaction("关闭 Bug");
//添加检查点
web_reg_find("Text=selfClose","SaveCount=closebug_result",LAST);

web_submit_data("bug-close",
    "Action=http://192.168.10.188/zentao/bug-close-{Bug_Index}.html?onlybody=yes",
    "Method=POST",
    "TargetFrame=hiddenwin",
    "RecContentType=text/html",
    "Referer=http://192.168.10.188/zentao/bug-close-{Bug_Index}.html?onlybody=yes",
    "Snapshot=t7.inf",
    "Mode=HTML",
    ITEMDATA,
    "Name=comment","Value=关闭 Bug",ENDITEM,
    LAST);

//手动判断事务是否成功
if(atoi(lr_eval_string("{closebug_result}"))>0)
{
    lr_end_transaction("关闭 Bug",LR_PASS);
}
else
{
    lr_end_transaction("关闭 Bug",LR_FAIL);
}

web_url("bug-browse-3-0-assignToMe-0.html",
    "URL=http://192.168.10.188/zentao/bug-browse-3-0-assignToMe-0.html",
    "TargetFrame=",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=http://192.168.10.188/zentao/bug-browse-3-0-assignToMe-0.html",
```

```
        "Snapshot=t8.inf",
        "Mode=HTML",
        LAST);

  return0;
}
```

Script6: 确认 Bug

```
//开始事务
  lr_start_transaction("确认 Bug");
  //插入检查点
  web_reg_find("Text=selfClose","SaveCount=SureBugResult",LAST);

  web_submit_data("bug-confirmBug.html_2",
        "Action=http://192.168.10.206:81/zentao/bug-confirmBug-{BugId}.html?onlybody=yes",
        "Method=POST",
        "TargetFrame=hiddenwin",
        "RecContentType=text/html",
        "Referer=http://192.168.10.206:81/zentao/bug-confirmBug-{BugId}.html?onlybody=yes",
        "Snapshot=t9.inf",
        "Mode=HTML",
        ITEMDATA,
        "Name=assignedTo","Value=dev{DevIndex}",ENDITEM,
        "Name=pri","Value=3",ENDITEM,
        "Name=mailto[]","Value=",ENDITEM,
        "Name=comment","Value=可重现确认为 Bug ",ENDITEM,
        LAST);

  //手动判断事务结果
if(atoi(lr_eval_string("{SureBugResult}"))>0)
  {
lr_end_transaction("确认 Bug",LR_PASS);
  }
```

```
else
{
    lr_end_transaction("确认 Bug",LR_FAIL);
}

web_url("bug-browse.html",
    "URL=http://192.168.10.206:81/zentao/bug-browse.html",
    "TargetFrame=",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=http://192.168.10.206:81/zentao/bug-browse.html",
    "Snapshot=t10.inf",
    "Mode=HTML",
    LAST);
```

3.5、测试方法及案例设计

说明本次测试的测试方法（内容）及测试案例、测试场景设计。下面章节供参考。

- 系统登录场景设计如下:

场景 1	登陆系统				
目的	测试多人同时登陆系统的性能情况				
方法					
Vusers 和资源					
并发用户数	CPU/Mem/Disk 数据	Apache 数据	MySql 数据	网络使用率	事务平均响应时间
10 人					
20 人					

- 添加测试用例场景设计如下:

场景 2	添加测试用例				
目的	测试多人同时添加测试用例的性能情况				
方法					
Vusers 和资源					
并发用户数	CPU/Mem/Disk 数据	Apache 数据	MySql 数据	网络使用率	事务平均响应时间
10 人					

- 提交 Bug 场景设计如下:

场景 3	提交 Bug				
目的	测试多人同时提交 Bug 的性能情况				
方法					
Vusers 和资源					
并发用户数	CPU/Mem/Disk 数据	Apache 数据	MySql 数据	网络使用率	事务平均响应时间
10 人					

- 解决 Bug 场景设计如下:

场景 4	解决 Bug				
目的	测试多人同时更改 Bug 为 fixed 状态的性能情况				
方法					
Vusers 和资源					
并发用户数	CPU/Mem/Disk 数据	Apache 数据	MySql 数据	网络使用率	事务平均响应时间
15 人					
30 人					

- 关闭 Bug 场景设计如下:

场景 5	关闭 Bug				
目的	测试多人同时关闭 Bug 的性能情况				
方法	10 个 Vusers 并发（测试人员才使用关闭 Bug 功能）				
Vusers 和资源					
并发用户数	CPU/Mem/Disk 数据	Apache 数据	MySql 数据	网络使用率	事务平均响应时间
10 人					

- 确认 bug 场景设计如下:

场景 6	确认 bug				
目的	测试多人同时确认 Bug 的性能情况				
方法	20 个 Vusers 并发（测试人员才使用关闭 Bug 功能）				
Vusers 和资源					
并发用户数	CPU/Mem/Disk 数据	Apache 数据	MySql 数据	网络使用率	事务平均响应时间
20 人					

- 登录负载测试:

场景 7	登陆系统				
目的	评估登录场景最大 TPS				
方法	120-270 人登陆系统的 TPS				
Vusers 和资源					
并发用户数	CPU 使用率	Mem 使用率	TPS	网络使用率	事务平均响应时间

100-140 人					
100-180 人					
180-220 人					

- 添加用例最大 TPS:

场景 8	添加用例				
目的	评估添加用例场景最大 TPS				
方法	120-270 人添加用例的 TPS				
Vusers 和资源					
并发用户数	CPU 使用率	Mem 使用率	TPS	网络使用率	事务平均响应时间
100-140 人					
100-180 人					
180-220 人					

- 执行用例最大 TPS:

场景 9	执行用例				
目的	评估执行用例场景最大 TPS				
方法	120-270 人执行用例的 TPS				
Vusers 和资源					
并发用户数	CPU 使用率	Mem 使用率	TPS	网络使用率	事务平均响应时间
100-140 人					
100-180 人					
180-220 人					

- 提交 Bug 最大 TPS:

场景 10	提交 Bug				
目的	评估提交 Bug 场景最大 TPS				
方法	120-270 人提交 bug 的 TPS				
Vusers 和资源					
并发用户数	CPU 使用率	Mem 使用率	TPS	网络使用率	事务平均响应时间
100-140 人					
100-180 人					
180-220 人					

- 解决 Bug 最大 TPS:

场景 11	解决 Bug
目的	评估解决 Bug 场景最大 TPS
方法	120-270 人解决 bug 的 TPS
Vusers 和资源	

并发用户数	CPU 使用率	Mem 使用率	TPS	网络使用率	事务平均响应时间
100-140 人					
100-180 人					
180-220 人					

- 关闭 Bug 最大 TPS:

场景 12	关闭 Bug				
目的	评估关闭 Bug 场景最大 TPS				
方法	120-270 人关闭 bug 的 TPS				
Vusers 和资源					
并发用户数	CPU 使用率	Mem 使用率	TPS	网络使用率	事务平均响应时间
100-140 人					
100-180 人					
180-220 人					

4.1、基准测试

在测试环境经过确认，脚本预验证之后对本次测试涉及的全部联机交易做基准测试。目的是验证测试脚本及后台环境、初步检查交易本身是否存在性能缺陷。

测试方法:

使用 LoadRunner 测试工具向 192.168.10.188 服务器发送交易请求，接收并分析返回结果。拟采用 10Vuser 负载执行，取交易的平均响应时间作为衡量指标，并计算吞吐量

4.2、单场景负载测试

对本次测试涉及的全部联机交易完成基准测试后，分别执行单交易负载测试。目的是获得交易本身的性能表现，诊断交易是否存在性能缺陷。

测试方法:

使用 LoadRunner 测试工具向 206 服务器发送交易请求，接收并分析返回结果。

4.3、稳定性测试

多场景压测系统 7x24 小时

3.6、测试输出

/*说明在测试完成后需要输出的阶段性成果，作为检验测试的衡量标准。

当测试完成以后，需提交的主要文档包括，但不仅限于：

《禅道开源版 v8.0.1 性能测试方案》

《禅道开源版 v8.0.1 性能测试记录及问题跟踪表》

《禅道开源版 v8.0.1 性能测试报告》

3.7、测试进度计划

在测试工作量估算数据的基础上，考虑现有的资源情况，对资源进行具体安排，根据项目整体进度计划，列出进度表，即是谁在什么时间内完成什么任务。下表供参考，非强制使用。

序号	名称	责任人	工期	开始时间	完成时间
1	禅道开源版 v8.0.1 **性能测试**		**X 工作日**	**2016-2-22**	**2016-2-28**
1.1	**测试准备**		**x 工作日**	**2016-2-22**	**2016-2-22**
1.1.1	测试实施方案制定		x 工作日	**2016-2-22**	**2016-2-22**
1.1.2	测试主机、数据库环境就绪		x 工作日	**2016-2-22**	**2016-2-22**
1.1.4	性能测试业务数据就绪		x 工作日	**2016-2-22**	**2016-2-22**
1.1.5	服务部署就绪		x 工作日	**2016-2-22**	**2016-2-22**
1.1.6	测试脚本编制、参数就绪		x 工作日	**2016-2-23**	**2016-2-23**
1.2	**基准、单交易负载测试**		**x 工作日**	**2016-2-23**	**2016-2-24**
1.2.1	单交易基准测试		x 工作日	**2016-2-23**	**2016-2-24**
1.4	**负载测试**		**x 工作日**	**2016-2-24**	**2016-2-26**
1.5	**测试总结**		x 工作日	**2016-2-27**	**2016-2-28**

3.8、实施风险及规避措施

/*风险管理是对影响项目测试的各种可能发生的风险进行估计，以及对风险的发生几率和严重程度进行估计，并按照估计结果对风险进行排序*/

风险描述	风险发生的可能性	风险对项目的影响	责任人	规避方法
测试环境与运行环境差距较大，通过测试得到的运行参数偏差。在试运行阶段需要重新进行参数验证。	中	低		根据测试结果反推生产环境，由运维同学监控数据并作及时调整。
测试数据量和数据库中预埋数据量较小，通过测试时间推算的批量处理交易的运行时间满足要求，生产环境下数据不能满足。	中	高		等比缩放，减小系统数据容量。
由于发现较严重缺陷引发较长	高	中		预留测试环境维护

时间的程序修改，或因环境准备、数据等原因造成测试进度延迟。				时间，不定时备份测试环境和数据

四、脚本开发详解

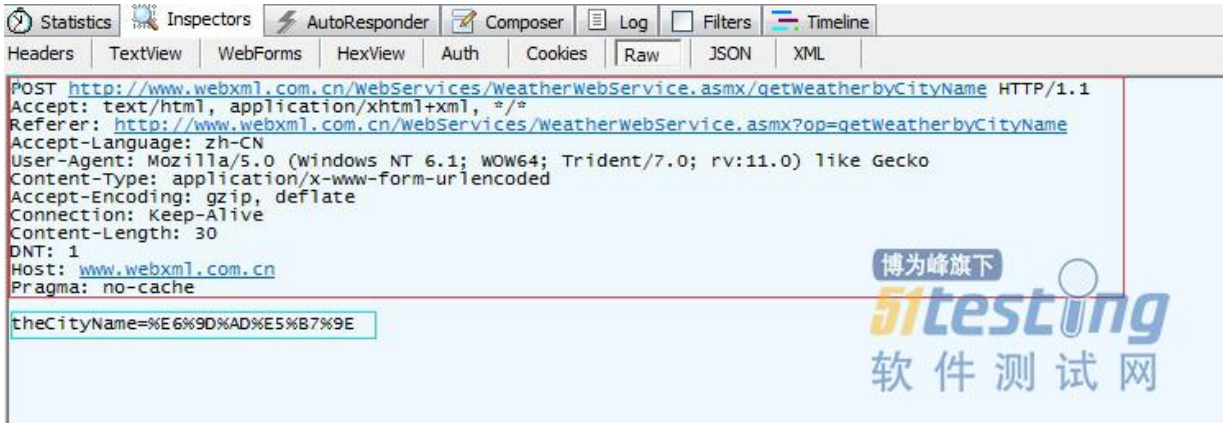
(PS：不管是录制还是手工开发脚本，对协议报文体系的了解是基本功必不可少的，我们先熟悉下 http 协议的报文体系哈)

4.1、协议报文体系

抓包工具有很多，重量级的有 wireshark 次之有 fiddler,charles 和 Omnipeek，轻量级的小插件当然也可以比如 firebug 和 httpwatch 都很轻巧好用抓包必备神器。

4.1.1、http 协议报文体系

4.1.1.1、http 协议的请求格式



我们现在从上向下依次看下请求报文里都是些什么鬼

POST 表示 http 的方法

(ps：http 的方法有以下几种

- GET 请求获取 Request-URI 所标识的资源
- POST 在 Request-URI 所标识的资源后附加新的数据
- HEAD 请求获取由 Request-URI 所标识的资源的响应消息报头
- PUT 请求服务器存储一个资源，并用 Request-URI 作为其标识
- DELETE 请求服务器删除 Request-URI 所标识的资源
- TRACE 请求服务器回送收到的请求信息，主要用于测试或诊断

- CONNECT 保留将来使用
- OPTIONS 请求查询服务器的性能，或者查询与资源相关的选项和需求)

/WebServices/TrainTimeWebService.asmx/getStationAndTimeByStationName 请求的资源对象(这里是一个 webservice 接口中的一个方法)

HTTP/1.1 表示所使用的 http 协议版本号

Accept:text/html,application/xhtml+xml,*/*

请求报文头部的 Accept 表示浏览器端可接受的媒体类型。

(PS:Accept:*/*代表浏览器可以处理所有返回的媒体类型；Accept:text/html 代表浏览器可以接受服务器回发的类型为 text/html；如果服务器无法返回 text/html 类型的数据，服务器应该返回一个 406 错误(nonacceptable))

Referer:http://ws.webxml.com.cn/WebServices/TrainTimeWebService.asmx?op=getStationAndTimeByStationName

告诉服务器当前请求是从哪个页面链接过来的，服务器可以获得一些信息用于处理。

(PS:我们抓包可以看到每次请求的 Header 中，基本上都有 Referer 值，值就是当前页面的 url。

浏览器所做的工作相当简单。如果在页面 A 上点击一个链接，然后进入 B 页面，浏览器就会在请求中插入一个带有页面 A 的 Referer 首部；自己输入的 URL 中不会保护 Referer 首部)。

Accept-Language:zh-CN

指定客户端可接受或优选哪些语言(比如，内容所使用的自然语言)

User-Agent:Mozilla/5.0(WindowsNT6.1;WOW64;Trident/7.0;rv:11.0)likeGecko

客户端应用程序信息(浏览器内核，操作系统内核信息等)

Content-Type:application/x-www-form-urlencoded

指定请求报文中对象的媒体类型(MIME)。

(PS: 我们常见的媒体类型如下:

1.text: 用于标准化地表示的文本信息，文本消息可以是多种字符集和或者多种格式的；默认是 text/plain;

2.multipart: 用于连接消息体的多个部分构成一个消息，这些部分可以是不同类型的数据；默认是 multipart/mixed;

3.application: 用于传输应用程序数据或者二进制数据；默认是 application/octet-stream;

4.message：用于包装一个 E-mail 消息；

5.image：用于传输静态图片数据；

6.audio：用于传输音频或者音声数据；

7.video：用于传输动态影像数据，可以是与音频编辑在一起的视频数据格式。)

Accept-Encoding:gzip,deflate

指定客户端接受的编码方式。通常指定压缩方法，是否支持压缩，支持什么压缩方法（gzip，deflate，compress;q=0.5 这不是指字符编码）

Connection:Keep-Alive

声明与服务器的连接机制，如 keep-alive 等

主要用于有代理网络环境，这样服务器或其他代理就可以指定不应传递的跳转首部了。

Content-Length:30

请求的实体报文长度

DNT:1

表示是否开启 DNT（Donottrack）功能，1 表示开启，0 表示关闭。

Host:www.webxml.com.cn

声明需要请求 URI 的主机信息。

Pragma:no-cache

Pragma 头域用来包含实现特定的指令，最常用的是 Pragma:no-cache。在 HTTP/1.1 协议中，它的含义和 Cache-Control:no-cache 相同。

theCityName=%E6%9D%AD%E5%B7%9E

(PS:这里是请求的 body(主体)部分，我也把=号后面的字符串解码后是杭州)

http 协议的请求头部总结:

常用的标准请求头部包括下面几个:

1) Accept，声明哪种相应是可接受的，如 text\plain、application\json 等。

2) Cache-Control，声明缓存控制机制，如 no-cache 声明不做缓存。

3) Connection，声明与服务器的连接机制，如 keep-alive 等。

4) Cookie，声明 Cookie 信息。

5) Content-Type，声明请求体的 MIME 类型。

6) Host，声明需要请求 URI 的主机信息。

7) If-Match，声明匹配这个请求的 Key，如果服务器的 ETag 与这个 Key 一致，则认为这个请求的资源没有发生改变，客户端可以选择从还从中加载这个请求。

8) Origin，声明跨域请求的时候支持什么域名进行访问。

9) User-Agent，声明发出这个请求的客户端的描述，如果是浏览器发出的请求，可以根据这个头判断是哪个浏览器的哪个版本。

非标准请求头部包括下面几个:

1) X-Request-With，通常通过这个头告诉服务器这个请求是 XMLHttpRequest 发送的。

2) DNT，表示是否开启 DNT（Donottrack）功能，1 表示开启，0 表示关闭。

3) Front-End-Https，是微软用的一个自定义头，与负载均衡有关。

4) Proxy-Connection，与标准头 Connection 一致，是早期 HTTP 协议的产物。

4.1.1.2、http 协议的响应格式

响应报文头部信息

HTTP/1.1200OK

http 协议的版本，服务器返回的响应状态码，

Date:Thu,18Aug201606:37:22GMT

报文创建的日期和时间。

Server:Microsoft-IIS/6.0

Web 服务器的信息

X-Powered-By: ASP.NET

声明该响应是通过哪种语言生成的

X-AspNet-Version: 2.0.50727

Cache-Control: private,max-age=0

指定响应遵循的缓存机制

(PS: Cache-Control 主要有以下几种类型:

(1)请求 Request:

[1]no-cache----不要读取缓存中的文件，要求向 WEB 服务器重新请求

[2]no-store----请求和响应都禁止被缓存

[3]max-age: ----表示当访问此网页后的 max-age 秒内再次访问不会去服务器请求，其功能与 Expires 类似，只是 Expires 是根据某个特定日期值做比较。一但缓存者自身的时间不准确.则结果可能就是错误的，而 max-age,显然无此问题.。Max-age 的优先级也是高于 Expires 的。

[4]max-stale----允许读取过期时间必须小于 max-stale 值的缓存对象。

[5]min-fresh----接受其 max-age 生命期大于其当前时间跟 min-fresh 值之和的缓存对象

[6]only-if-cached----告知缓存者,希望内容来自缓存且并不关心被缓存响应是否是最新的.

[7]no-transform----告知代理,不要更改媒体类型.

(2)响应 Response:

[1]public----数据内容皆被储存起来，就连有密码保护的网页也储存，安全性很低
[2]private----数据内容只能被储存到私有的 cache，仅对某个用户有效，不能共享
[3]no-cache----可以缓存，但是只有在跟 WEB 服务器验证了其有效后，才能返回给客户端

[4]no-store----请求和响应都禁止被缓存

[4]max-age: -----本响应包含的对象的过期时间
[5]Must-revalidate----如果缓存过期了，会再次和原来的服务器确定是否为最新数据，而不是和中

间的 proxy

[6]max-stale----允许读取过期时间必须小于 max-stale 值的缓存对象。

[7]proxy-revalidate----与 Must-revalidate 类似，区别在于：proxy-revalidate 要排除掉用户代理的缓存的。即其规则并不应用于用户代理的本地缓存上。

[8]s-maxage----与 max-age 的唯一区别是,s-maxage 仅仅应用于共享缓存.而不应用于用户代理的本地缓存等针对单用户的缓存.另外,s-maxage 的优先级要高于 max-age.

[9]no-transform----告知代理,不要更改媒体类型,比如 jpg,被改成 png

)

Content-Type: text/xml;charset=utf-8

响应报文的媒体类型。

Content-Length: 7325

服务器响应报文 body 长度。

常用的 http 协议的响应头信息总结:

1: Access-Control-Allow-Origin，声明这个响应可以参与到哪个域的跨域访问中。*表示可以参与到任何域的跨域访问。

2: Allow，声明这个 HTTP 响应是使用哪个 HTTP 方法，如 GET、POST 等。如果是一个不支持的 HTTP 方法，则会返回错误码 405Methodnotallowed。

3: Content-Type，声明这个响应的 MIME 类型。

4: ETag，声明这个响应版本的 key，可以标识一个资源是否有改变过（参考请求头 If-Match）。

5: Pragma，声明这个响应是否支持缓存，可以设置 no-cache 禁用这个响应的缓存。

6: Refresh，声明这个响应在特定时间后刷新或者跳转到新的 URL。

7: Status，响应的状态码。

非标准的响应头部包括下面几个:

X-Frame-Options，声明防止 Clickjacking 攻击的参数，如 deny 就是防止响应被渲染在 iframe 里面，而 sameorigin 就是防止响应在非本域的页面中渲染。

X-Content-Type-Options，帮助 IE 能不识别 MIME 类型不对的 stylesheet 和 script，也用于 Chrome 下载其扩展。

X-Powered-By，声明该响应是通过哪种语言生成的。

4.1.1.3、已定义的状态码及其意义

状态码:	原因短语:	含义:
100	Continue(继续)	收到请求的起始部分，客户端应该继续请求
101	SwitchingProtocols(切换协议)	服务器正根据客户端的指示协议切换成 update 首部列出的协议
200	OK	服务器已经成功处理请求
201	Created(已创建)	对那些要服务器创建对象的请求来说资源已创建完毕
202	Accept(已接受)	请求已接受，但服务器尚未处理
203	Non-AuthoritativeInformatino(非权威信息)	服务器成功处理，只是实体头部来自资源的副本
204	NoContent(没有内容)	响应报文包含一些首部和一个状态行但不包含实体的主题内容
205	ResetContent(重置内容)	浏览器应该重置当前页面上所有的 HTML 表单
206	PartialContent(部分内容)	部分请求成功
300	MultipleChoices(多项选择)	客户端请求了实际指向多个资源的 URL
301	MovedPermanently(永久搬离)	请求的 URL 已移走，响应中应包含一个 LocationURL 说明资源新位置
302	Found(已找到)	请求的 URL 临时性移走。
303	SeeOther(参考其他)	告诉客户端应用另一个 URL
304	NotModified(未修改)	资源未发生过改变
305	UseProxy(使用代理)	必须通过代理访问资源，代理的位置在 Location 首部
307	TemporaryRedirect(临时重定向）	
400	BadRequest(坏请求）	服务器不能理解收到的请求，即发出了异常请求。
401	Unauthorized(未授权）	在客户端获得资源访问权之前，先要进行身份认证

403	Forbidden(禁止)	服务器拒绝了请求
404	NotFound(未找到)	服务器无法找到所请求的 URL
405	MethodNotAllowed(不允许使用的方法)	请求中有一个所请求的 URI 不支持的方法
406	NotAcceptable(无法接受)	
407	ProxyAuthenticationRequired(要求进行代理认证)	
408	RequestTimeout(请求超时)	
409	Conflict(冲突)	发出的请求在资源上造成了一些冲突
410	Gone(已消失)	请求的资源曾经在服务器存在
411	LengthRequired(要求长度指示)	服务器要求在请求报文中包含 Content-Length 头部
412	PreconditionFailed(先决条件失败)	服务器无法满足请求的条件
413	RequestEntityTooLarge(请求实体太大)	客户端发送的请求所携带的请求 URL 超过了服务器能够或者希望处理的长度
414	RequestURITooLong(请求 URI 太长)	
415	UnsupportedMediaType(不支持的媒体类型)	服务器无法理解或不支持客户端所发送的实体的内容类型
416	RequestedRangeNotSatisfiable(所请求的范围未得到满足)	
417	ExpectationFailed(无法满足期望)	
500	InternalServerError(服务器内部错误)	
501	NotImplemented(未实现)	
502	BadGateway(网关故障)	作为代理或网关使用的服务器遇到了来自响应链中上游的无效响应
503	ServiceUnavailable(未提供次服务)	服务器目前无法为请求提供服务
504	GatewayTimeout(网关超时)	
505	HTTPVersionNotSupported(不支持的 HTTP 版本)	

4.2、录制生成的脚本

4.2.1、录制 PC 端应用脚本

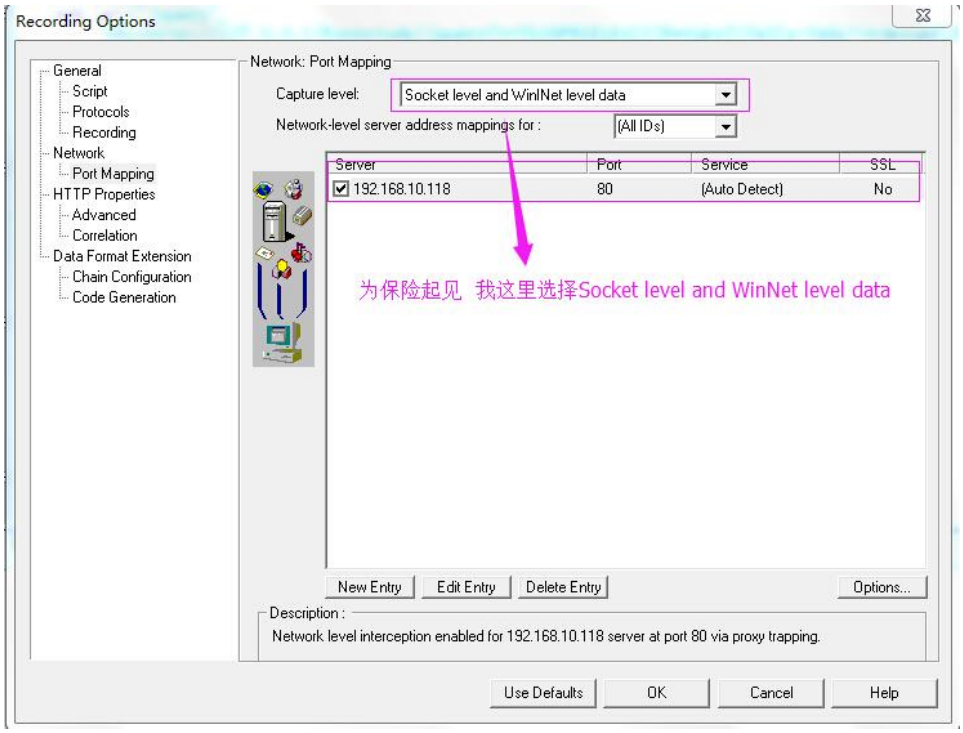
1: 配置浏览器网络代理(以 Chrome51 为例)

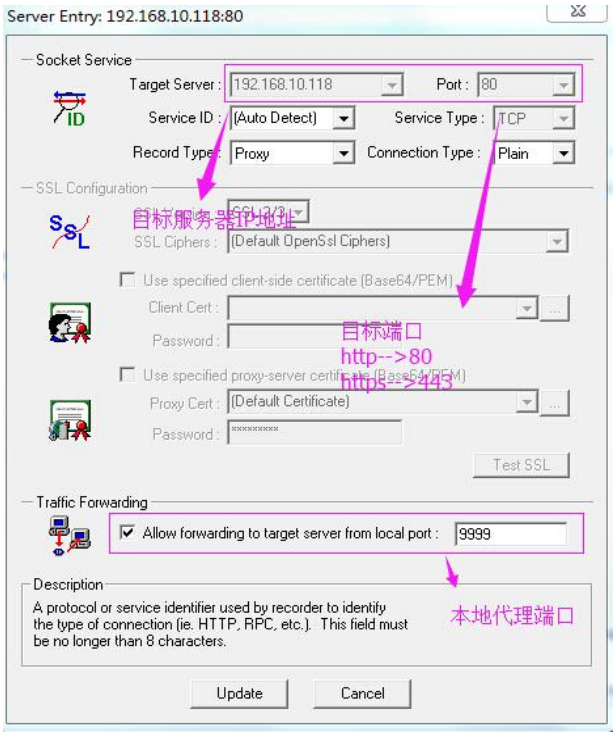
“设置”→“网络”→“更改代理服务器设置”

代理地址设置为 127.0.0.1 端口要使用未被占用的端口(此处为 9999)

2: 配置 LoadRunnerRecordOptions 中的 PortMapping

“RecordingOptions”→“PortMapping”→“NewEntry”





TargetServer: 目标服务器的 IP 地址

比如：我想访问 118 主机上的 Easystudy 服务这里 IP 地址填写如上图。

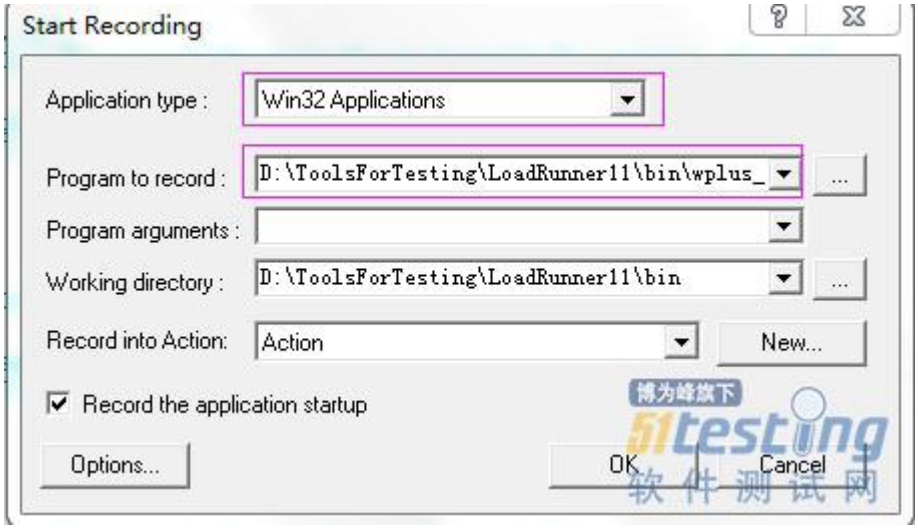
TrafficForwarding: 勾选允许通过本地端口，这里的本地端口要填写一个没被占用的（和浏览器端的设置一致 9999）。

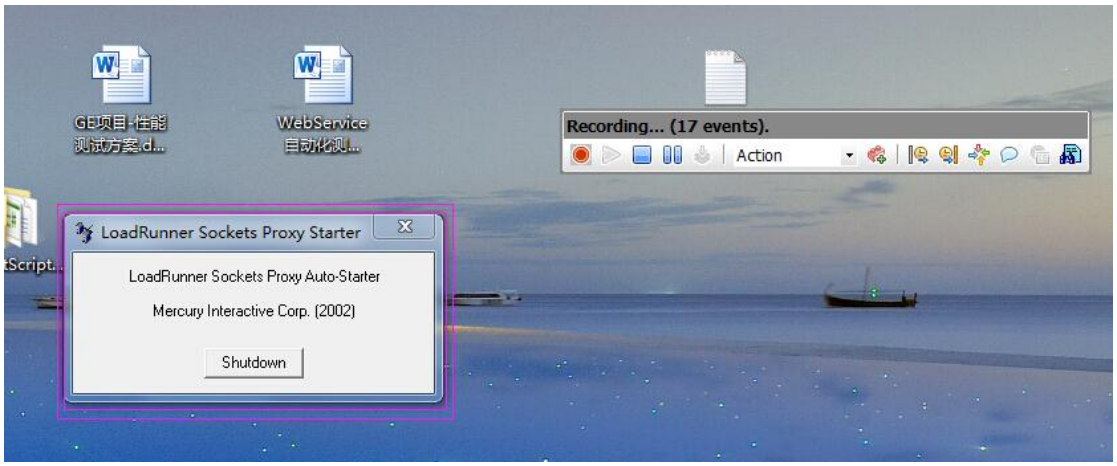
注意事项如下:

1) Applicationtype 要设置为 Win32Applications

通过 PortMapping 录制脚本的时候

Applicationtype 已经不是 InternetApplication 了,这里需要选择为 Win32Applications，以为录制的时候 LR 会启动一个“LoadRunnerSocketProxyStarter”的小窗口。

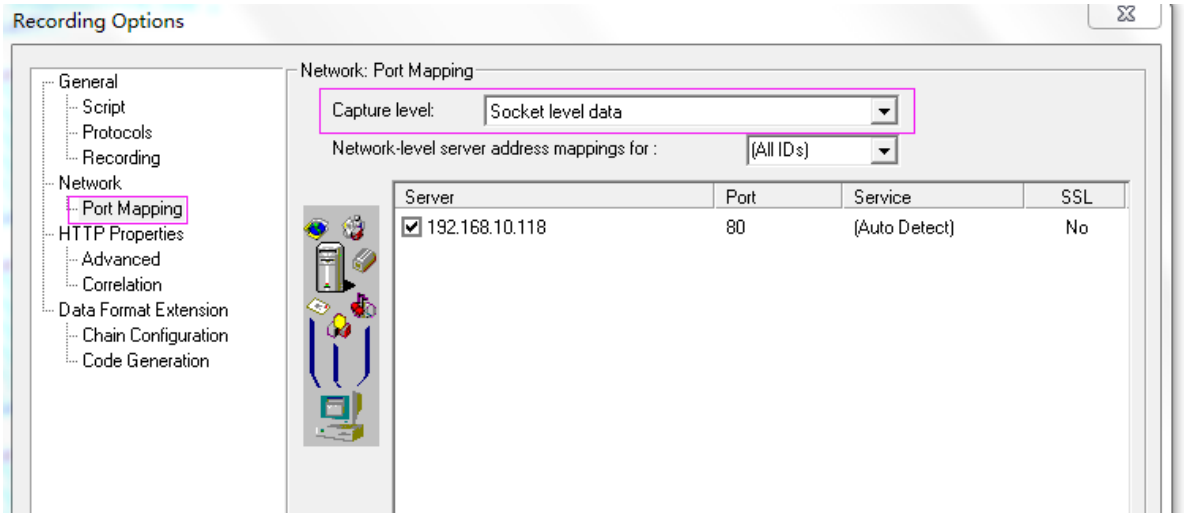




2) Programtorecord

必须填写为 LR 安装目录下 bin\wplus_init_wsock.exe 文件的绝对路径。

3) Capturelevel 的设置



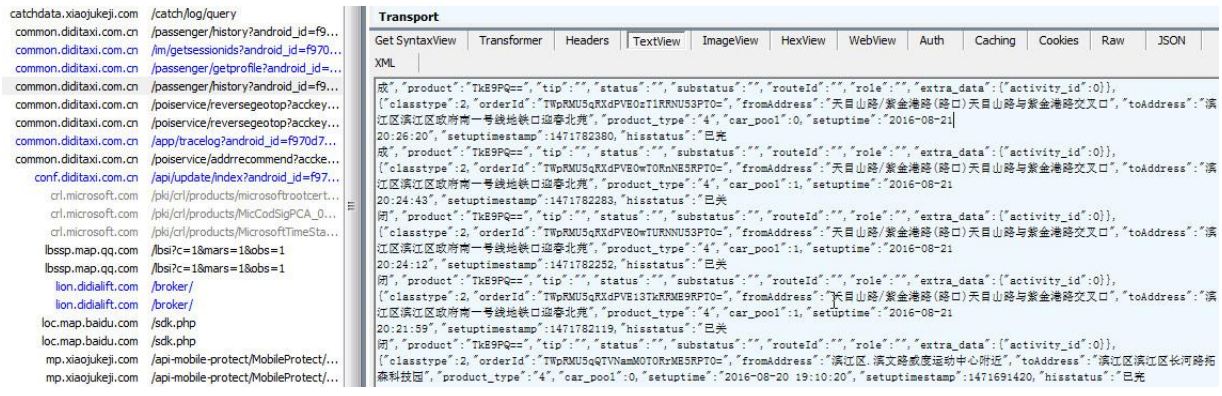
默认情况下是 Socketleveldata，如果录制完成后脚本为空，需要更改为 SocketlevelandWinNetleveldata 然后重新录制即可。

4.2.2、抓包移动端 APP 应用脚本

1：共享笔记本上的 wifi(我这里用的是猎豹的 wifi 共享小工具)。

2：设置手机端 wifi 代理

3：开启 Fiddler 抓包(下图以滴滴出行为例)



(PS：好好学下 Fiddler 这款神器哈~)

4.3、全手工开发脚本

4.3.1、C-Vuser 脚本开发

(ps: 了解了前面的 http 报文的结构，接着要乘胜追击一起来看下如何根据抓包内容手工开发性能脚本吧)

4.3.1.1、操作报文头部的函数

为下一个请求添加头部信息:

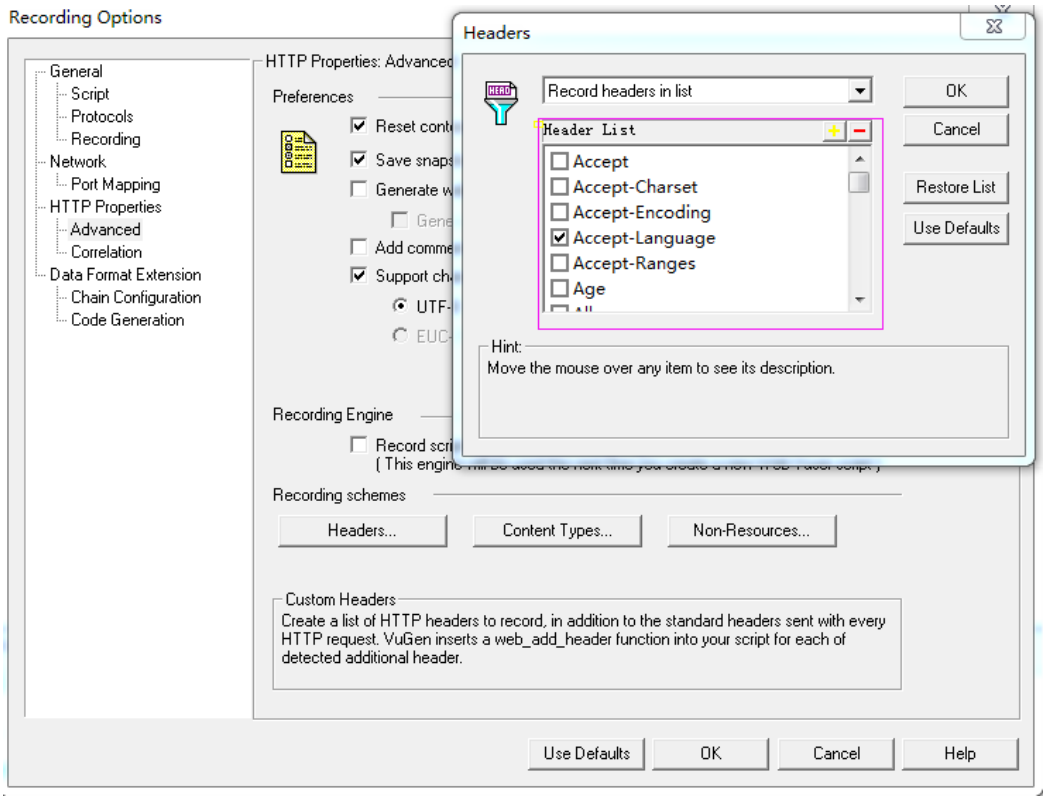
web_add_header(“Accept-Language”,“zh-CN”)

为后续的所有请求添加头部信息:

web_add_auto_header("Accept–Encoding","gzip");

我也也可以在录制脚本的时候通过设置来确定哪些请求头的信息需要保存:

RecordOptions---→Advanced---→Headers



另外：**web_cleanup_auto_headers()**停止向后面的 HTTP 请求中添加自定义标头。

web_revert_auto_header 停止向后面的 HTTP 请求中添加特定的标头，但是生成隐性标头。

4.3.1.2、过滤指定 URL 请求的函数

web_add_filter（“action=Include/Exclude”, “Attribute=xxx”,LAST）

设置下一个请求中包含或排除的 URL

web_add_auto_filter（“action=Include/Exclude”, “Attribute=xxx”,LAST）

为后续所有请求设置包含或排除的 URL

4.3.1.2、操作 cookie 的函数

web_add_cookie("lang=zh-cn;theme=default;windowWidth=1584;windowHeight=353;sid=0lab6lvctq0hh2flm48qq520h2");

增加一个新的 cookie 或修改现有的 cookie

web_remove_cookie 删除指定的 cookie(和 web_add_cookie 对应)

web_add_cookie_ex("Cookie=client_id=China127B;path=/;expires=Wednesday,09-Nov-200123:12:40GMT;domain=","Insert=Alt","AllowEmptyDomain=yes",LAST);

添加自定义的 cookie

web_cleanup_cookies()

清空当前用户保存的所有 cookie

4.3.1.4、模拟报文发送的函数

Web_url 函数格式:

```
web_url("Login",
    "URL=http://localhost/somostorage/UserInfo/Login",
    "TargetFrame=",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t7.inf",
    "Mode=HTML",
    EXTRARES,
```

"Url=../Images/Login/logo-center.png","Referer=http://localhost/somostorage/Content/themes/login/gelogin.css",ENDITEM,

"Url=../Images/Login/gebutton.png","Referer=http://localhost/somostorage/Content/themes/login/gelogin.css",ENDITEM,

"Url=../images/icomoon/icomoon.ttf","Referer=http://localhost/somostorage/Content/UIForIpadAndIphone/css/glyphs.css",ENDITEM,

"Url=../Images/Images/user_btn.png","Referer=http://localhost/somostorage/Content/UIForIpadAndIphone/css/index.css",ENDITEM,

"Url=../Content/UIForIpadAndIphone/images/mainview/jiantou.png","Referer=http://localhost/somostorage/Content/UIForIpadAndIphone/css/index.css",ENDITEM,

"Url=../Images/Buttons/page_go.png","Referer=http://localhost/somostorage/Content/themes/bluestyle/cont.css",ENDITEM,

"Url=../Content/UIForIpadAndIphone/images/mainview/bg_fileheader.png","Referer=http://localhost/somostorage/Content/UIForIpadAndIphone/css/index.css",ENDITEM,

"Url=../Content/UIForIpadAndIphone/images/mainview/4.png","Referer=http://localhost/somostorage/Content/UIForIpadAndIphone/css/index.css",ENDITEM,

"Url=../Content/UIForIpadAndIphone/images/imageview/mleft.png","Referer=http://localhost/somostorage/Content/UIForIpadAndIphone/css/index.css",ENDITEM,

"Url=../Content/UIForIpadAndIphone/images/imageview/mright.png","Referer=http://localhost/somostorage/Content/UIForIpadAndIphone/css/index.css",ENDITEM,

LAST);

Web_submit_data 函数格式:

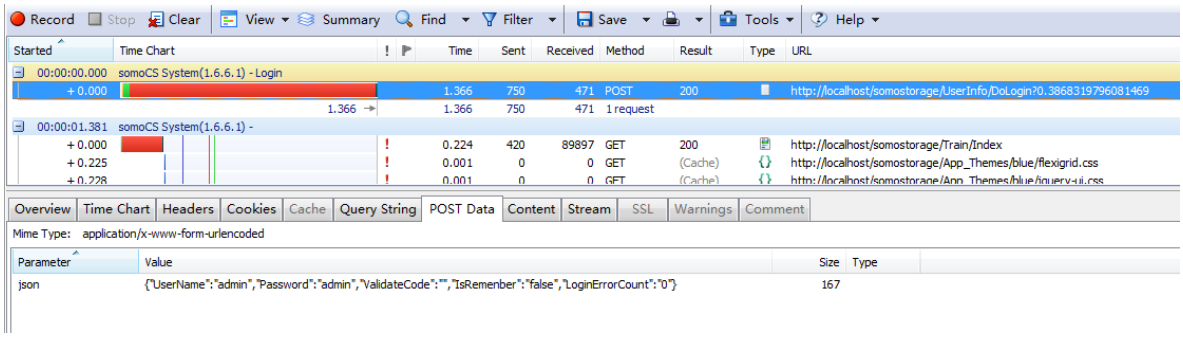
HTTP 协议 post 方法常见的有 4 种方式，所以脚本的格式也不尽相同:

1: application/x-www-form-urlencoded

这应该是最常见的 POST 提交数据的方式了。浏览器的原生<form>表单，如果不设置 enctype 属性，那么最终就会以 application/x-www-form-urlencoded 方式提交数据。

首先，Content-Type 被指定为 application/x-www-form-urlencoded；其次，提交的数

据按照 key1=val1&key2=val2 的方式进行编码，key 和 val 都进行了 URL 转码。大部分服务端语言都对这种方式有很好的支持。

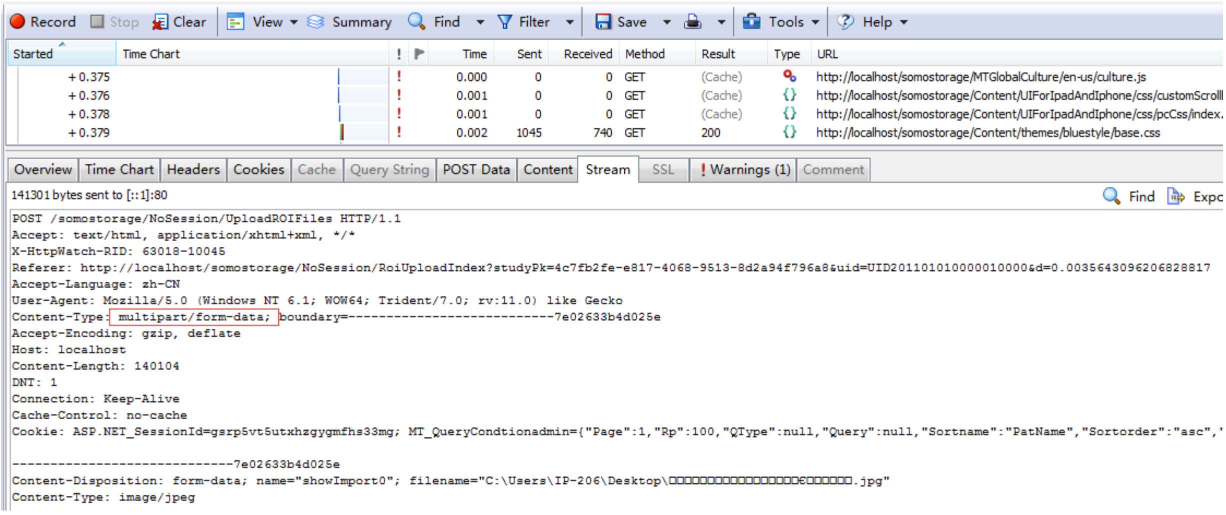


application/x-www-form-urlencoded 格式的 post 请求脚本如下:

```
web_submit_data("DoLogin",
 "Action=http://localhost/somostorage/UserInfo/DoLogin",
 "Method=POST",
 "TargetFrame=",
 "RecContentType=application/json",
 "Referer=http://localhost/somostorage/UserInfo/Login",
 "Snapshot=t8.inf",
 "Mode=HTML",
 ITEMDATA,
 "Name=json","Value={\"UserName\":\"admin\",\"Password\":\"admin\",\"ValidateCode\":\"\",\"IsRemenber\":\"false\",\"LoginErrorCount\":\"0\"}",ENDITEM,
      LAST);
```

2: multipart/form-data

这又是一个常见的 POST 数据提交的方式。我们使用表单上传文件时，必须让<form>表单的 enctyped 等于 multipart/form-data。



web_submit_data(

//stepname

"UploadROIFiles",

//请求的 URL 地址

"Action=http://localhost/somostorage/NoSession/UploadROIFiles",

//模拟的 http 请求方法

"Method=POST",

//编码类型

"EncType=multipart/form-data",

//包含当前链接资源的 frame 名称

"TargetFrame=",

"RecContentType=text/html",

//要提交该页面请求的 URL

"Referer=http://localhost/somostorage/NoSession/RoiUploadIndex?studyPk=4c7fb2fe-e817-4068-9513-8d2a94f796a8&uid=UID20110101000010000&d=0.7157794737040377",

//快照名

"Snapshot=t29.inf",

//录制模式有 html 和 http

"Mode=HTML",

//属性和数据列表的分割标记，JavaVuser 中不可用。

ITEMDATA,

"Name=showImport0","Value=前端工程师技能栈.jpg","File=Yes",ENDITEM,

//ENDITEM 是列表中每个资源的结束标识符

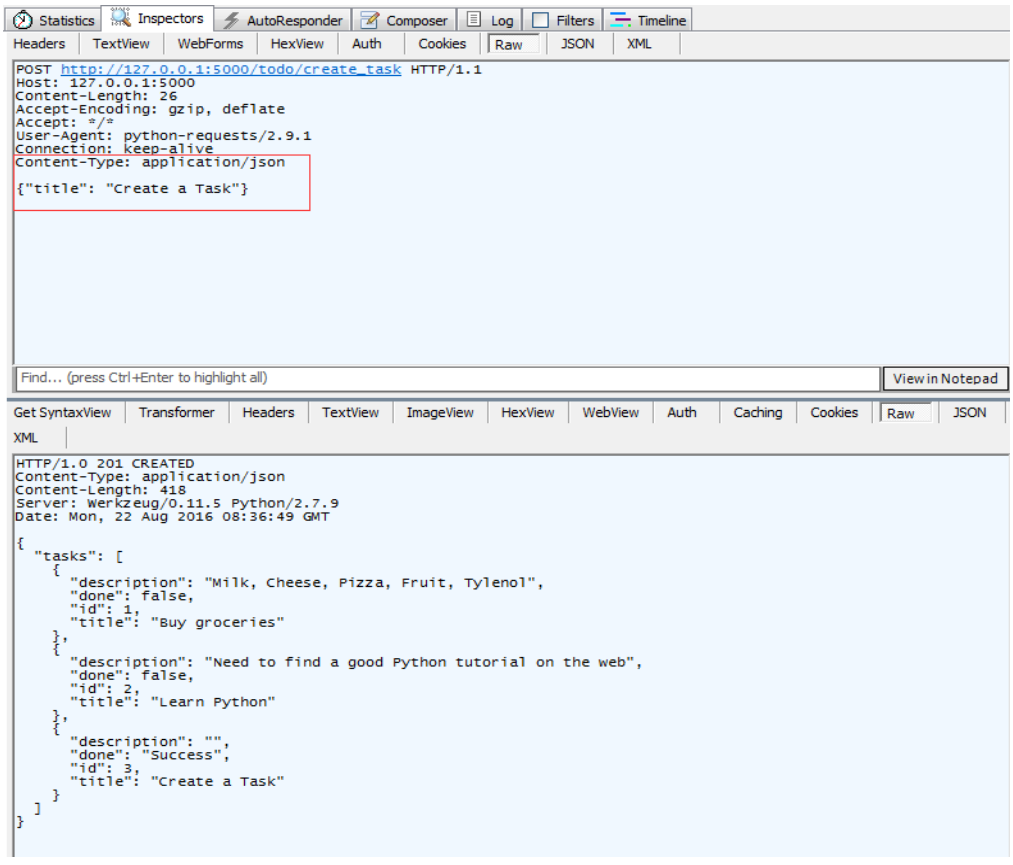
```
"Name=showImport1","Value=","File=Yes",ENDITEM,
"Name=showImport2","Value=","File=Yes",ENDITEM,
"Name=showImport3","Value=","File=Yes",ENDITEM,
"Name=showImport4","Value=","File=Yes",ENDITEM,
"Name=hidMarkerPk","Value=",ENDITEM,
"Name=hidStudyPk","Value=4c7fb2fe-e817-4068-9513-8d2a94f796a8",ENDITEM,
"Name=hidStudyUid","Value=UID201101010000010000",ENDITEM,
"Name=hidPostData","Value=",ENDITEM,
"Name=hidType","Value=",ENDITEM,
"Name=casePk","Value=",ENDITEM,
    LAST);
```

我们接着看下另一个非常常用的函数

Web_custome_request()

这个函数可以模拟 get 和 post 请求，所以如果我们手工开发脚本的时候可以用这个函数搞定所有类型的请求报文发送的模拟。

3: application/json



```
web_custom_request(
//stepname 这里可以随意命名
"restful_api",
//定义需要模拟的 http 请求的方法
    "Method=POST",
//请求的 url
"URL=http://127.0.0.1:5000/todo/create_task",
"Mode=HTTP",
//编码类型
    "EncType=application/json",
//接收的内容类型
    "RecContentType=application/json",
//请求报文的主体部分
"Body={\"title\":\"CreateaTask\"}",
"TargetFrame=",
LAST);
```

对 HTTP 或者 HTTPS 协议的报文模拟来说，使用 web_url，web_submit_data，web_custome_request 这三个函数足够了。

4.3.1.5、事务相关的函数

lr_start_transaction(“事务名称”)

事务的开始标识

lr_end_transaction("事务名称",事务状态)

事务的结束标识

(PS:这里需要注意：对于所有事务都不要使用 LR 自带的 LR_AUTO 来判断事务成功与否，为什么？

因为 LR_AUTO 只判断发送到服务器端的请求格式是否正确而不会去判断我们的业务，LR 还没智能到清楚地知道每个使用者的业务逻辑这个地步)。

4.3.1.6、其他常用的函数

检查点：web_reg_find

web_reg_find("Text=Welcome","SaveCount=Welcome_Count",LAST);

注册一个保存动态数据为参数的请求（关联）:

web_reg_save_param("outFlightVal","LB=outboundFlightvalue=","RB=checked>",LAST);

注册一个将动态数据边界值为参数的请求(这个函数是 web_reg_save_param 的升级):

web_reg_save_param_ex("When_Txt","LB=Whereand","RB=do",LAST)

字符串转换函数:

atoi()函数用来将字符串转换成整数(int)

atoi 读取字符串的起始部分，通过在第一个非数值字符停止

返回脚本中的一个参数当前的值:

lr_eval_string

将程序中的常量或变量保存为 lr 中的参数:

lr_save_string

将两个 char 类型字符串连接:

strcat

字符串编码转换

lr_convert_string_encoding

(PS: LR 的函数还有很多其他的函数，各位同学可通过 F1 来查看使用方法)

4.3.1.7、业务逻辑的重要性

讲个自己在之前开发脚本的过程中遇到一个小故事吧~~

测试环境配置:

LoadRunner11(patch3+patch4)

HttpWatch9.3

Web 服务器: IIS6.0

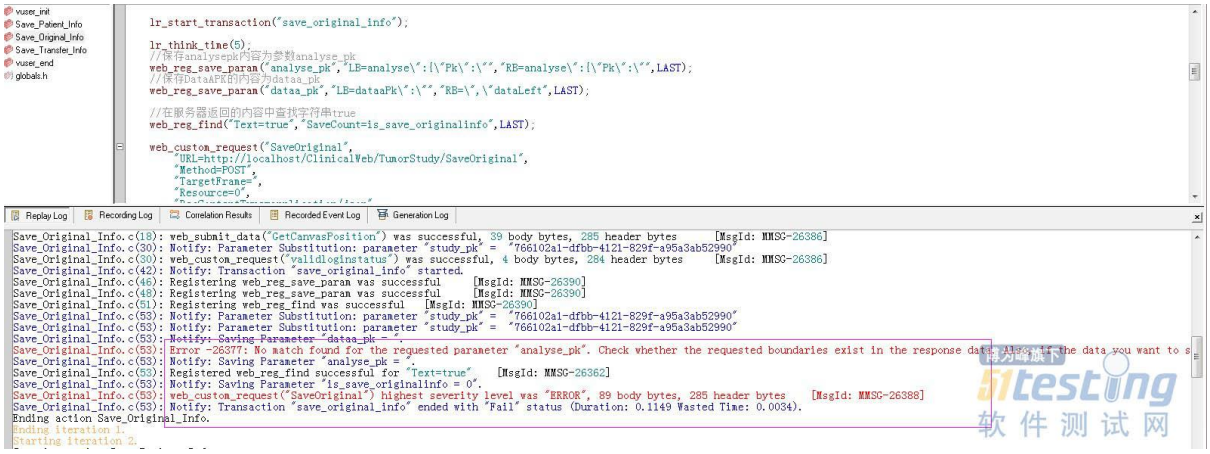
OS:win764 位

浏览器：火狐 28，IE11

业务操作：添加病理评估检查(分三步：1：保存病人基本信息 2：填写保存原发灶信息 3：填写保存转移灶信息)

初始版脚本：通过 FireFox28 用 LoadRunner 进行录制。

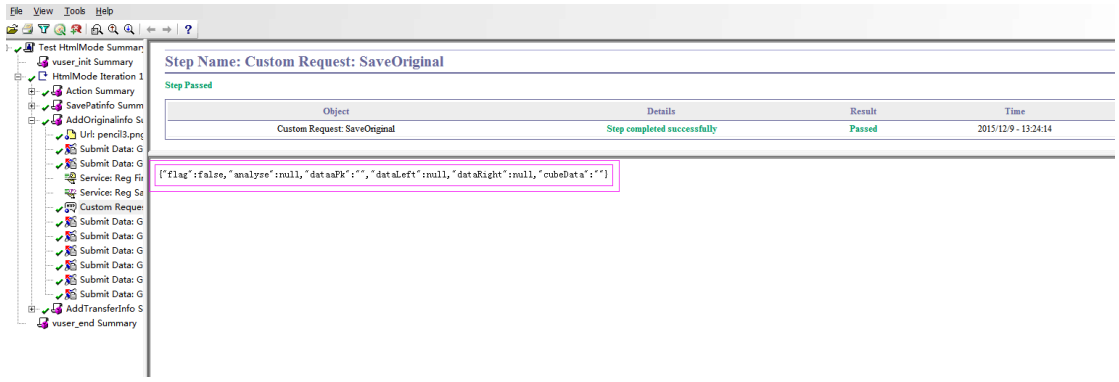
回放问题：进行参数化，关联动态数据后，卡在保存原发灶这一步，脚本截图如下：



回放日志信息：需要进行关联的字段 analysepk 不存在

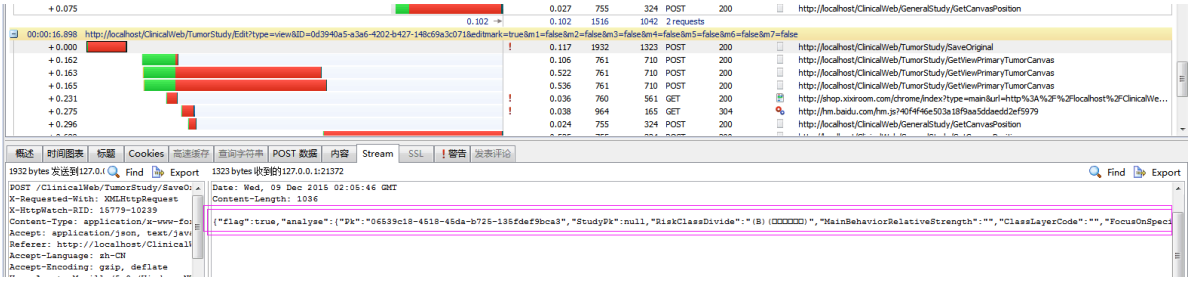
第一想法就是把这一步骤从服务器返回的信息打印出来。

修改这一步之前的注册函数 web_reg_save_param 中左右边界分别为“{“和”}”然后 lr_output_message 输出服务器返回的信息，结果如下：



看到 flag 字段对应的值为 false 个人猜测很可能是业务失败。

使用 HttpWatch 抓包查看保存原发灶时服务器正确的返回格式，如下图：



看到 flag 字段对应值为 true 追问开发同学后确定：flag=false 是业务没处理成功。

再三检查 Body 数据(经历了 URL 解码，随机字符串生成俩坑)确认无误~~在此 Run 脚本错误依然存在。

思虑再三，决定找开发同学要源码看看**这一步操作调用哪个方法？接收几个参数？参数的类型是什么？对接收的参数做什么判断处理？**

为方便开发跟进，我把脚本里的服务地址改到开发同学的机器上有 ta 打断点步步跟进。

开发定位第一步：发现报出异常信息，数据库里某个 pk 字段违反了唯一性约束

我检查自己的脚本发现问题所在

```
Body=jsonData={\"StudyPk\":\"{study_pk}\",
\"GSHistoryInfo\":{\"Pk\":\"{random_pk}\",\"BackgroundDisease\":\"/\",\"MarkStudy\":2},
\"TSMicroscopeInfo\":{\"Pk\":\"{random_pk}\",\"StoveSpread\":\"/\",\"PathologySubtype\":\"/\"},
\"TSBUltrasonicList\":[{\"Pk\":\"{random_pk}\",\"StovePosition\":\"/\",\"StoveNum\":\"/\",
\"StoveMeanDiameter\":\"/\",\"MarkAlign\":\"1\"},{\"Pk\":\"{random_pk}\",\"StovePosition\":\"/\",\"StoveNum\":\"/\",
\"StoveMeanDiameter\":\"/\",\"MarkAlign\":\"2\"},{\"Pk\":\"{random_pk}\",\"StovePosition\":\"/\",\"StoveNum\":\"/\",
\"StoveMeanDiameter\":\"/\",\"MarkAlign\":\"3\"},{\"Pk\":\"{random_pk}\",
\"StovePosition\":\"/\",\"StoveNum\":\"/\",\"StoveMeanDiameter\":\"/\",\"MarkAlign\":\"4\"}]}",
```

我在 httpwatch 里查找这些 pk 值只有保存原发灶的请求时才出现，个人臆测为可以传入为相同的随机 32 位字符串。

追着开发继续跟进这些 pk 不是从服务器端返回的是不是可以随机生成？可不可以不写？

开发定位第二步：最终确定这些 pk 值不传也是可以的，于是将该请求中这些 pk 字段值置为空，重新 Run 脚本。Bingo~~~!搞定！

```
    LAST);
web_custom_request("SaveOriginal",
    "URL=http://192.168.10.100/syf1.1/TumorStudy/SaveOriginal",
    "Method=POST",
    "Resource=0",
    "RecContentType=application/json",
    "Referer=http://192.168.10.100/syf1.1/TumorStudy/Edit?type=view&ID={study_pk}&editmark=true&m1=false&m2=f
    "Snapshot=t92.inf",
    "Mode=HTTP",
    "EncType=application/x-www-form-urlencoded; charset=UTF-8",
    "Body=jsonData={\"StudyPk\":\"{study_pk}\",\"GSHistoryInfo\":{\"Pk\":\"\",\"BackgroundDisease\":\"/\",\"M
    LAST);
lr_output_message("服务器返回内容是：%s",lr_eval_string("{ContentData}"));

web_submit_data("GetViewPrimaryTumorCanvas_2",
    "Action=http://192.168.10.100/syf1.1/TumorStudy/GetViewPrimaryTumorCanvas",
    "Method=POST",
    "RecContentType=application/json",
    "Referer=http://192.168.10.100/syf1.1/TumorStudy/Edit?type=view&ID={study_pk}&editmark=true&m1=false&m2=f
    "Snapshot=t93.inf",
```

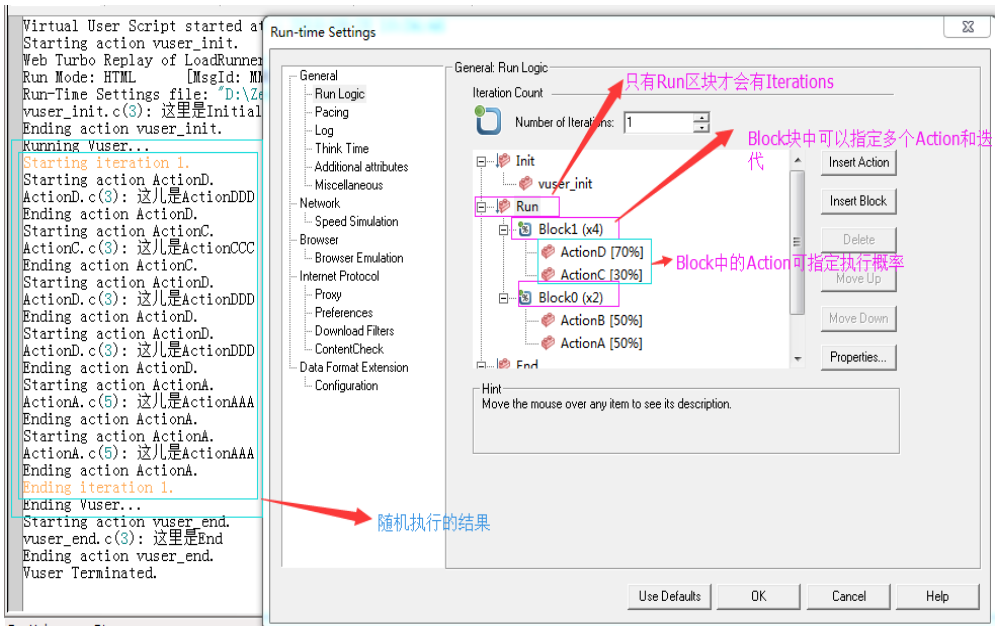
Recording Log | Correlation Results | Recorded Event Log | Generation Log

```
32): Registered web_reg_find successful for "Text=true" (count=1)     [MsgId: MMSG-26364]
32): web_custom_request("AddPatContent") was successful, 532 body bytes, 286 header bytes     [MsgId: MMSG-26386]
45): Notify: Transaction "111111" ended with "Pass" status (Duration: 0.5096 Wasted Time: 0.0249).
53): 随机字符串是：fs41rgnla11uuyh5aajl4c5heqzolg2b
55): Notify: Transaction "222222" started.
56): web_submit_data("GetViewPrimaryTumorCanvas") was successful, 432 body bytes, 286 header bytes     [MsgId: MMSG-26386]
57): web_submit_data("GetCanvasPosition") was successful, 39 body bytes, 285 header bytes     [MsgId: MMSG-26386]
79): Registering web_reg_save_param_ex was successful     [MsgId: MMSG-26390]
86): web_custom_request("SaveOriginal") was successful, 1029 body bytes, 287 header bytes     [MsgId: MMSG-26386]
98): 服务器返回内容是："Pk":"7dd699c7-0a97-4d95-a223-5fe6d4d44d44","StudyPk":null,"RiskClassDivide":"(浸为主)","MainBehavi
00): web_submit_data("GetViewPrimaryTumorCanvas_2") was successful, 424 body bytes, 286 header bytes     [MsgId: MMSG-26
```

也希望各位同学在性能测试脚本开发的过程中把理解业务这个点作为重中之重！

4.4、运行时设置&调试

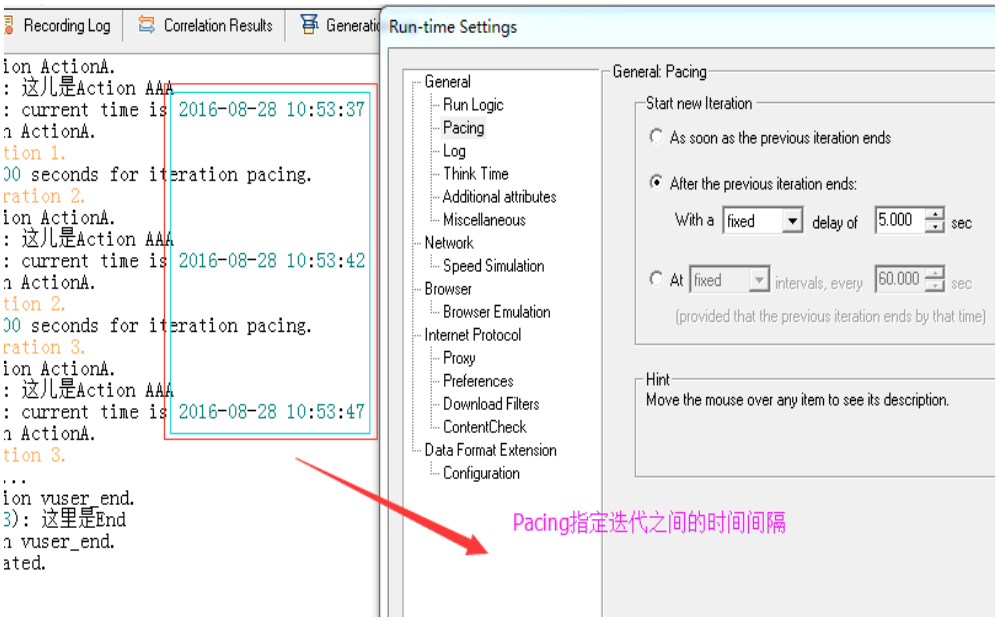
4.4.1、Run-Logic 规则



Run 区块中可指定整个 Run 逻辑的迭代次数，这里设置的迭代次数不会对 Initial 和 End 生效。

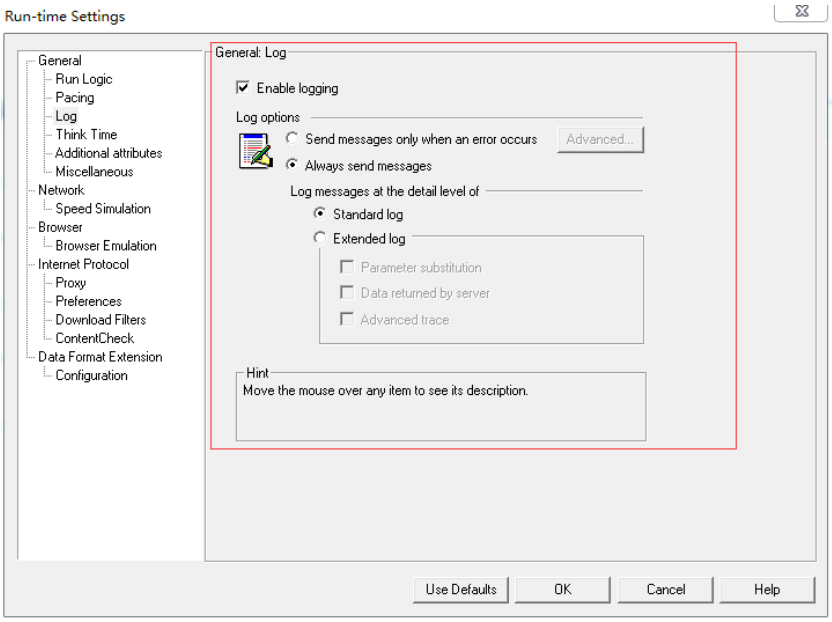
Block 中可包含多个 Action，并且可设置 Action 的执行概率。Block 中可包含 Block。

4.4.2、Pacing 设置



Pacing: 可设置迭代间的时间间隔。

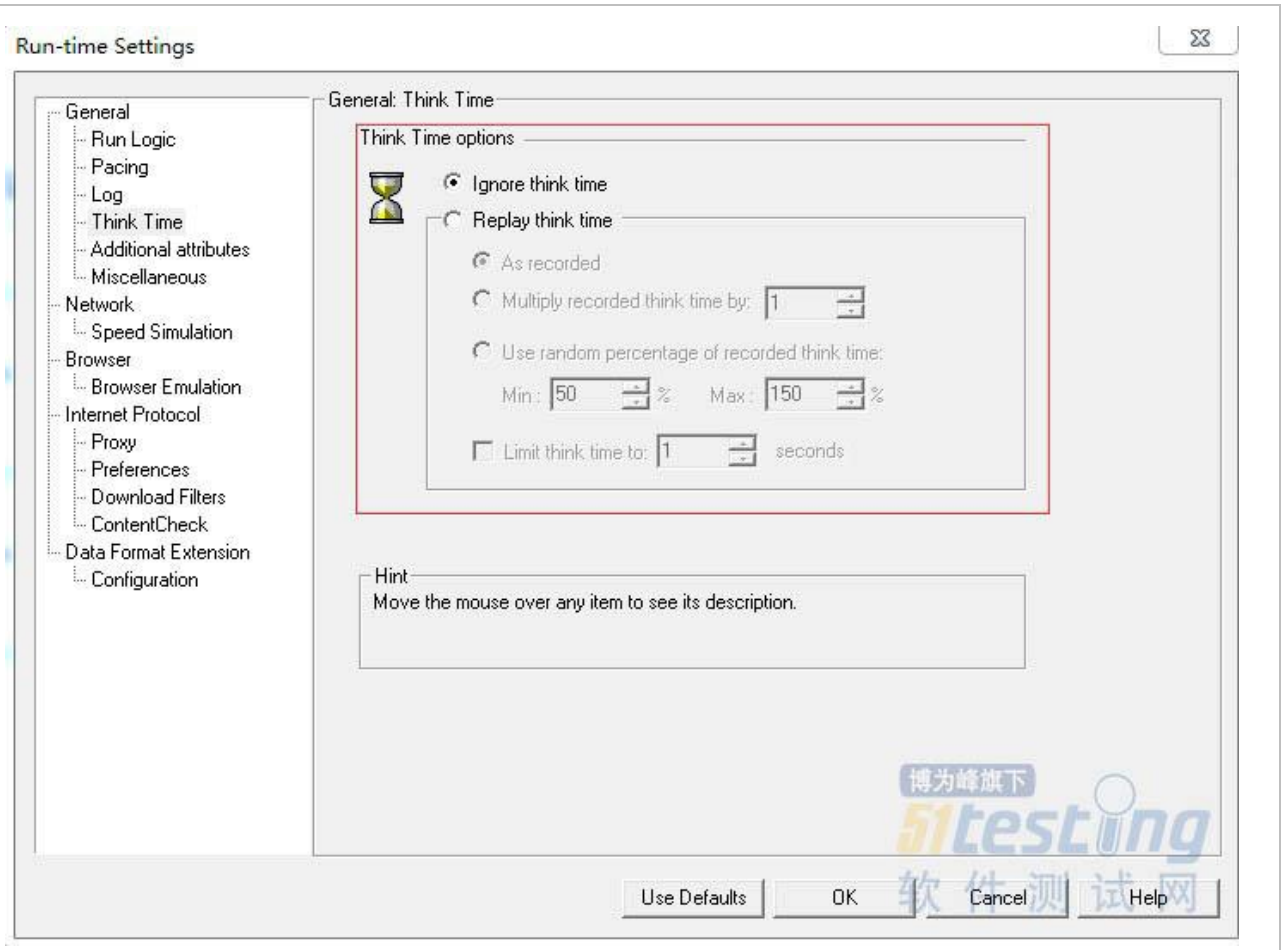
4.4.3、输出日志



(PS: 调试的时候可以勾选上 Enablelogging,若需要查看服务端返回的信息我们可以选择 Extendedlog，用到的最多场景是勾选 Parametersubstitution)

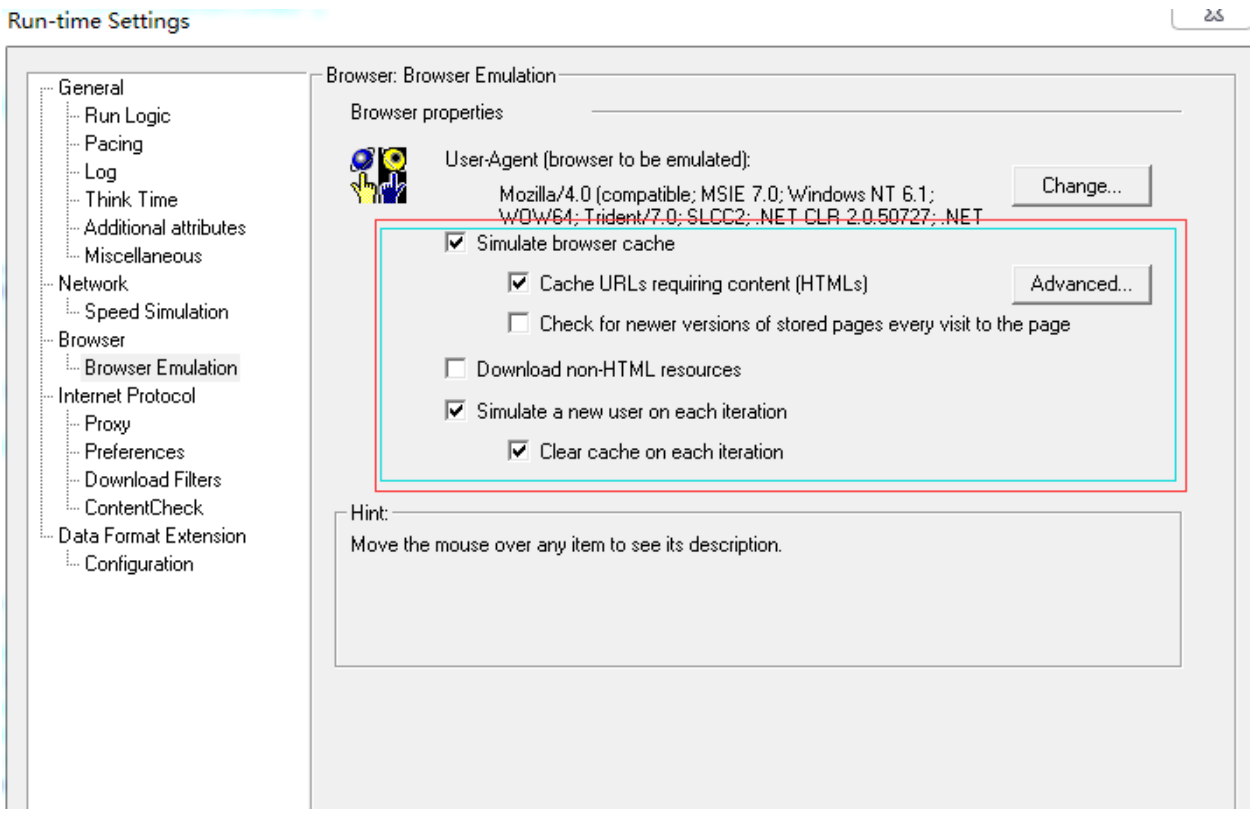
切记执行测试的时候把 log 关掉，否则硬盘里被塞满日志文件。

4.4.4、思考时间 Thinktime



(PS：思考时间用来模拟实际操作过程中的等待时间；如果只是想评估在一定环境配置下服务器能承受多大并发则可以忽略思考时间；另：思考时间一定不要放到事务内)

4.4.5、BrowserEmulation 浏览器模拟



simulatebrowsercache 选项

这个选项是指虚拟用户使用缓存模拟浏览器，缓存是用来将经常使用的文件保存到本地，这样可以减少网络连接时间，默认情况下，缓存模拟是启用的，当缓存被设置为禁用后，虚拟用户将忽略所有的缓存功能并且在每一次请求的时候下载所有的资源。(PS: 即使设置了禁用缓存模拟，对于页面上的每个资源仅被下载一次，即使该资源出现了多次。一种资源可以是一张图片、一个框架或者其他类型的脚本文件)。

当使用 LoadRunner 进行并发测试的时候，每个用户都使用自己的缓存并且从缓存中检索图片。如果禁用了缓存选项，所有的虚拟用户都不会使用缓存来模拟浏览器。

CacheURLsrequiringcontent(HTML)选项

这个选项是指 Vugen 仅缓存网页的一些必要信息，这些信息可以是一些必须的验证信息、分析数据或者关联数据，当你勾选了这项后，这些信息自动被缓存（默认是启用）。

Checkfornewerversionsofstoredpageseveryvisittothepage 选项

这个选项是指浏览器会将存储在 cache 中的网页信息和最新浏览的页面进行比较，当勾选此项时，vugen 会增加"If-modified-since"到 HTTP 包头，在场景执行过程中这个

选项可以显示最新的网页信息，但是也增加了更多的网络流量，通常配置这个选项是用来匹配浏览器设置来达到模拟浏览器的目的。

Downloadnon-HTMLresources 选项

这个选项是指虚拟用户在回放期间访问网站时加载图片的过程，这里图片是指随着页面录制的图片和那些没有随页面录制下来的图片。
（PS：禁用此选项后，可能会遇到图片验证失败，因为在访问网站的时候有些图片是会发生变化的）。

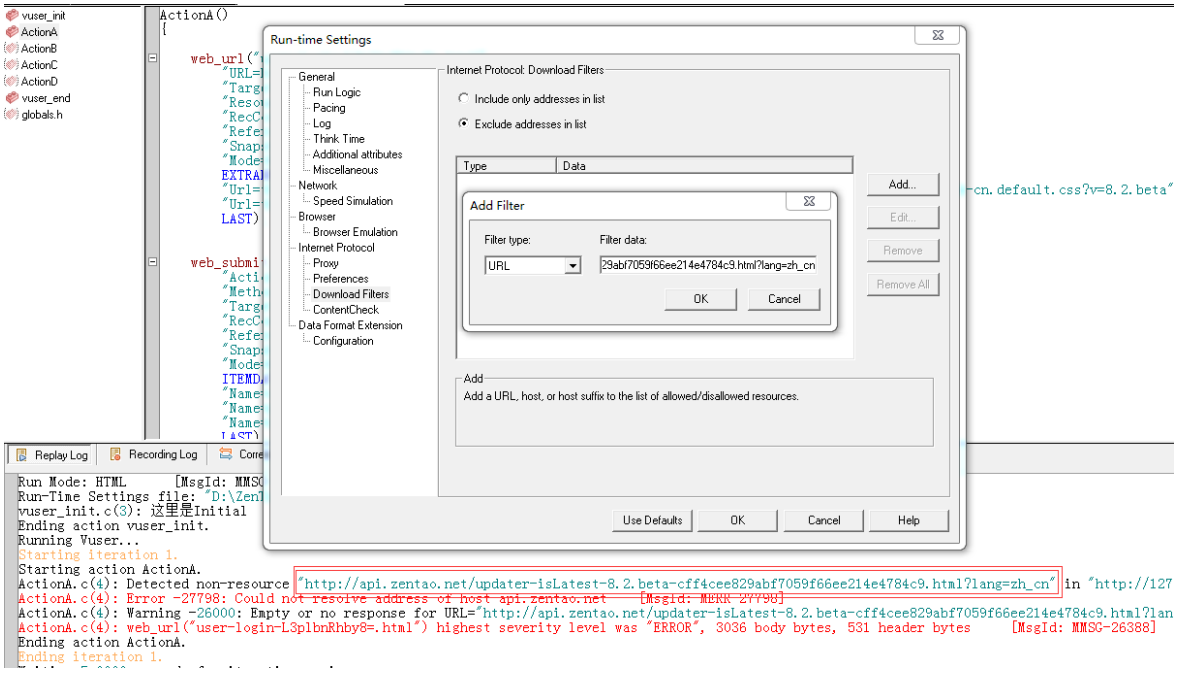
Simulateanewusereachiteration 选项

这个选项是指 VuGen 在迭代过程中重置了所有的 HTTP 内容，此设置允许虚拟用户能够更准确的模拟用户开始进行新的会话，它删除了所有的 Cookie，关闭了所有的 TCP 连接（包含保活包），清除了模拟浏览器的缓存，重置了 HTML 框架，并且清除了用户名和密码。

Clearcacheoneachiteration 选项

在每次迭代过程中清除浏览器中缓存来达到模拟一个真实用户第一次访问网页，清除该复选框以禁用此选项，允许虚拟用户使用缓存来存储用户信息，来模拟一个已经访问过网页的用户。

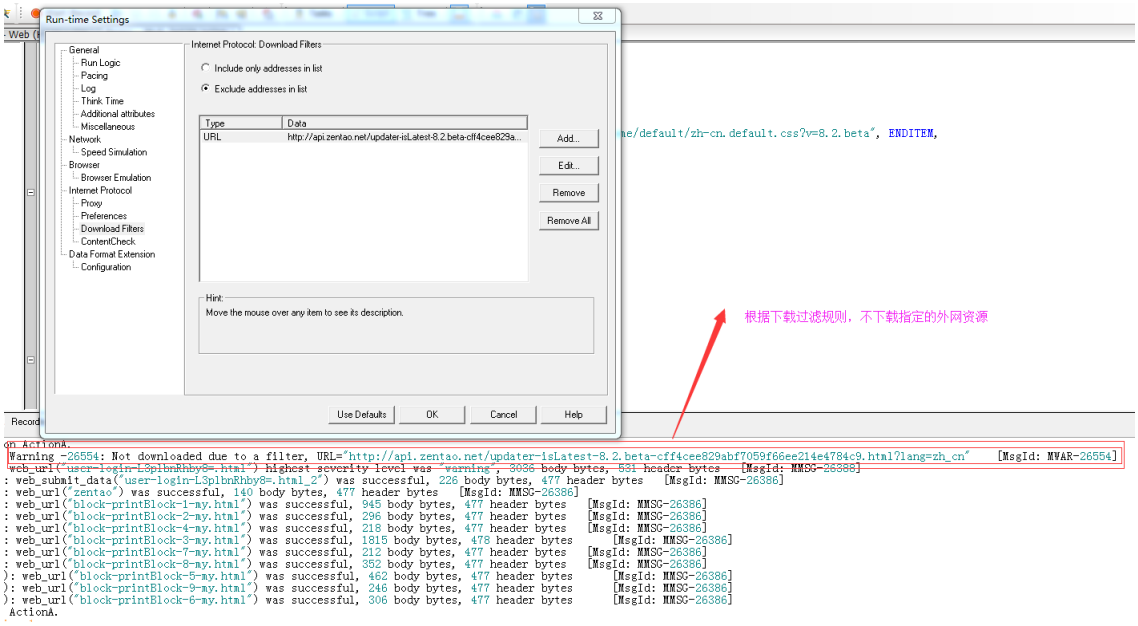
4.4.6、过滤外网的下载请求



(PS：在运行时设置中切换到 DownloadFilters 页，选择 Excludeaddressinlist，添加外网 URL

即可不下载外网资源)

设置后再次运行脚本日志中无报错信息，结果如下：



4.4.7、参数化及其取值规则

规则组合 1： Sequential+Eachiteration+Continuewithlastvalue

SelectNextValue	UpdateValueOn	WhenOutOfValues
Sequential	Eachiteration	Continuewithlastvalue

取值结果如下图在一次迭代中取同一个值；在不同迭代中按列表顺序取值；当参数不足时，进行下一轮循环参数列表。

```
ActionA()
{

    lr_output_message("取到的参数值是: %s",lr_eval_string("{Parameters}"));
    lr_output_message("一次迭代中第二次取到的参数值是: %s",lr_eval_string("{Parameters}"));
    return 0;
}
```

ay Log | Recording Log | Correlation Results | Generation Log

```
g action vuser_init.
ng Vuser...
ing iteration 1.
ing action ActionA.
nA.c(5): 取到的参数值是: 111
nA.c(7): 一次迭代中第二次取到的参数值是: 111
g action ActionA.
g iteration 1.
ing iteration 2.
ing action ActionA.
nA.c(5): 取到的参数值是: 222
nA.c(7): 一次迭代中第二次取到的参数值是: 222
g action ActionA.
g iteration 2.
```

规则组合 2： Sequential+Eachoccurence+Continuewithlastvalue

SelectNextValue	UpdateValueOn	WhenOutOfValues
Sequential	Eachoccurence	Continuewithlastvalue

取值结果：在一次迭代中，参数每出现一次则按顺序取一个新的参数值；不同迭代中则按顺序取新的值；参数不足时则循环参数列表顺序取值。

```
ActionA()
{

    lr_output_message("取到的参数值是: %s",lr_eval_string("{Parameters}"));
    lr_output_message("一次迭代中第二次取到的参数值是: %s",lr_eval_string("{Parameters}"));
    return 0;
}
```

```
y Log | Recording Log | Correlation Results | Generation Log
.ng action ActionA.
iA.c(5): 取到的参数值是: 111
iA.c(7): 一次迭代中第二次取到的参数值是: 222
; action ActionA.
; iteration 1.
.ng iteration 2.
.ng action ActionA.
iA.c(5): 取到的参数值是: 333
iA.c(7): 一次迭代中第二次取到的参数值是: 111
; action ActionA.
; iteration 2.
```

规则组合 3： Sequential+Once+Continuewithlastvalue

SelectNextValue	UpdateValueOn	WhenOutOfValues
Sequential	Once	Continuewithlastvalue

取值结果：无论多少次迭代，无论多少个虚拟用户进行并发，只取一个值。

```
    lr_output_message("取到的参数值是: %s",lr_eval_string("{Parameters}"));
    lr_output_message("一次迭代中第二次取到的参数值是: %s",lr_eval_string("{Parameters}"));
    return 0;
}
```

```
Log | Recording Log | Correlation Results | Generation Log
ng action ActionA.
A.c(5): 取到的参数值是: 111
A.c(7): 一次迭代中第二次取到的参数值是: 111
 action ActionA.
 iteration 1.
ng iteration 2.
ng action ActionA.
A.c(5): 取到的参数值是: 111
A.c(7): 一次迭代中第二次取到的参数值是: 111
 action ActionA.
 iteration 2.
```

规则组合 4： Random+Eachiteration+Continuewithlastvalue

SelectNextValue	UpdateValueOn	WhenOutOfValues
Random	Eachiteration	Continuewithlastvalue

取值结果：在一次迭代中，无论参数出现几次都根据随机取同一个值；不同迭代中则取新的随机值。

规则组合 5： Random+Eachoccurence+Continuewithlastvalue

SelectNextValue	UpdateValueOn	WhenOutOfValues
Random	Eachoccurence	Continuewithlastvalue

取值结果：在 Actions 中每次出现参数时都会随机取值。

规则组合 6： Random+Once+Continuewithlastvalue

SelectNextValue	UpdateValueOn	WhenOutOfValues
Random	Once	Continuewithlastvalue

取值结果：在 Actions 中只会随机取一个值。

规则组合 7： Unique+Eachiteration+Continuewithlastvalue

SelectNextValue	UpdateValueOn	WhenOutOfValues
Unique	Eachiteration	Continuewithlastvalue

取值结果：每次迭代都会取唯一的一个参数值；当参数不足时会取最后一个，但是 log 中会报错。

规则组合 8： Unique+Eachiteration+Continueinacyclicmanner

SelectNextValue	UpdateValueOn	WhenOutOfValues
Unique	Eachiteration	Continueinacyclicmanner

取值结果：每次迭代会取唯一的一个参数值；当参数不足时循环参数列表。

规则组合 9： Unique+Eachiteration+abortVuser

SelectNextValue	UpdateValueOn	WhenOutOfValues
Unique	Eachiteration	abortVuser

取值结果：每次迭代会取唯一的一个参数值；当参数不足时中止 Vuser 执行。

规则组合 10： Unique+Eachoccurence+Continuewithlastvalue

SelectNextValue	UpdateValueOn	WhenOutOfValues
Unique	Eachoccurence	Continuewithlastvalue

取值结果：每次参数出现会取唯一的一个参数值；当参数不足时日志中报错

规则组合 11： Unique+Eachoccurence+Continueinacyclicmanner

SelectNextValue	UpdateValueOn	WhenOutOfValues
Unique	Eachoccurence	Continueinacyclicmanner

取值结果：每次参数出现会取唯一的一个参数值；当参数不足时循环参数列表。

规则组合 12： Unique+Eachoccurence+abortVuser

SelectNextValue	UpdateValueOn	WhenOutOfValues
Unique	Eachoccurence	abortVuser

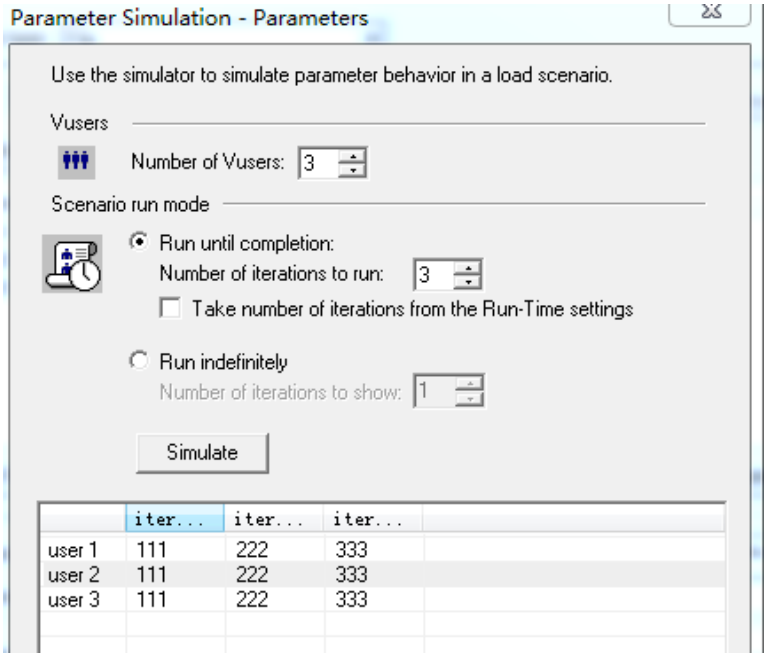
取值结果：每次参数出现会取唯一的一个参数值；当参数不足时中止 Vuser 执行。

规则组合 13： Random+Once+Continuewithlastvalue

SelectNextValue	UpdateValueOn	WhenOutOfValues
Random	Once	Continuewithlastvalue

取值结果：每个 Vuser 只会取唯一的一个参数值。

万法归一：可以通过 simulateparameters 来查看取参的结果。



4.4.8、关联的使用场景

关联是指获取服务器端返回的动态数据，将客户端的数据与服务器端的数据建立联系。

什么时候使用关联：

当我们需要使用从服务器端返回的动态数据时，我们需要使用 web_reg_save_param

这个注册函数将服务器端返回的数据保存到参数中。

下表列出 web_reg_save_param 可用的属性。

- **NotFound**

找不到边界并且生成了空字符串时的处理方法。默认值“ERROR”表示找不到边界时 LoadRunner 应发出错误消息。如果设置为“EMPTY”，则不会发出错误消息，并且脚本的执行将继续进行。注意，如果为脚本启用了“出现错误时仍继续”，则即使将 NOTFOUND 设置为“ERROR”，在找不到边界时脚本将仍然会继续执行，但会将错误消息写入扩展日志文件中

- **LB**

参数或动态数据的左边界。此参数必须为非空的、以 null 结尾的字符串。边界参数区分大小写；要忽略大小写，请在边界之后添加“/IC”。如果在边界之后指定“/BIN”，则指定为二进制数据。

- **RB**

参数或动态数据的右边界。此参数必须为非空的、以 null 结尾的字符串。其他规则呃 LB 一样。

- **RelFrameID**

与请求的 URL 相关的 HTML 页的层次结构级别。可能的值为 ALL 或数字。

- **Search**

搜索的范围-搜索已分隔的数据的位置。可能的值为 Headers（仅搜索页眉）、Body（仅搜索正文数据，而不包括页眉）或 ALL（搜索正文和页眉）。默认值为 ALL

- **ORD**

此可选参数表示匹配的序号或出现的次数。默认序号为 1。

如果指定“All”，则会将参数值保存在数组中,是将找到的动态变量保存到数组中。默认是 ord=1.如果搜索到的字符是多个，并想将他保存在数组里，则 ord=all;他们分别保存到 pr_1pr_2.....。其中 pr_count 为内部函数，统计数组的个数。

- **SaveOffset**

找到的值的子字符串偏移量，将保存到参数。默认值为 0。

偏移量值必须为非负数,偏移量。从搜索到的字符串中，取子串。默认 saveoffset=0.

- **Savelen**

找到的值的子字符串的长度（在指定的偏移量中），将保存到参数。默认值为 ñ1，表示直到字符串的末尾。

- **Convert**

要应用于数据的转换方法：HTML_TO_URL：将 HTML 编码数据转换为 URL 编码数据格式

HTML_TO_TEXT：将 HTML 编码数据转换为纯文本格式。

函数的一些使用技巧:

1、web_reg_save_param 必须在获取返回值的操作前面注册，在获取返回值的操作后面使用。

2、保存参数最大不能超过 256 字节，如果超过 256 字节请使用 intweb_set_max_html_param_len(constchar*length)函数扩大参数保存范围

例如：web_set_max_html_param_len("1024");//扩大参数最大保存范围为 1024 字节

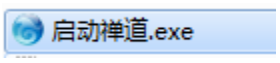
3、LB 和 RB 后面跟着"/ic",则边界大小写都匹配(不加，也就是默认是大小写敏感的)

例如：web_reg_save_param("IsRight","LB/ic=cache-control:private\r\n\r\n","RB/ic=|",LAST);

五、测试执行

5.1、搭建禅道环境

搭建 windows 下的禅道环境很简单，官网下载压缩包，解压后启动

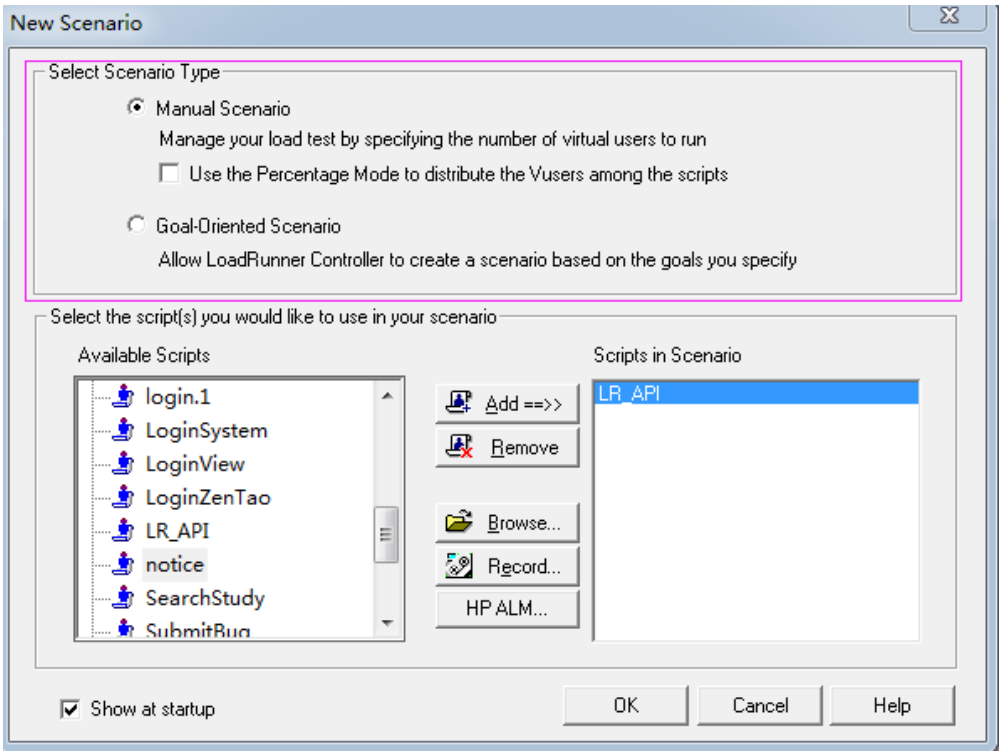


即可。

(ps：禅道集成环境试用 mysql 数据库，并集成了 phpmyadmin 来通过浏览器对数据库进行方便的管理)

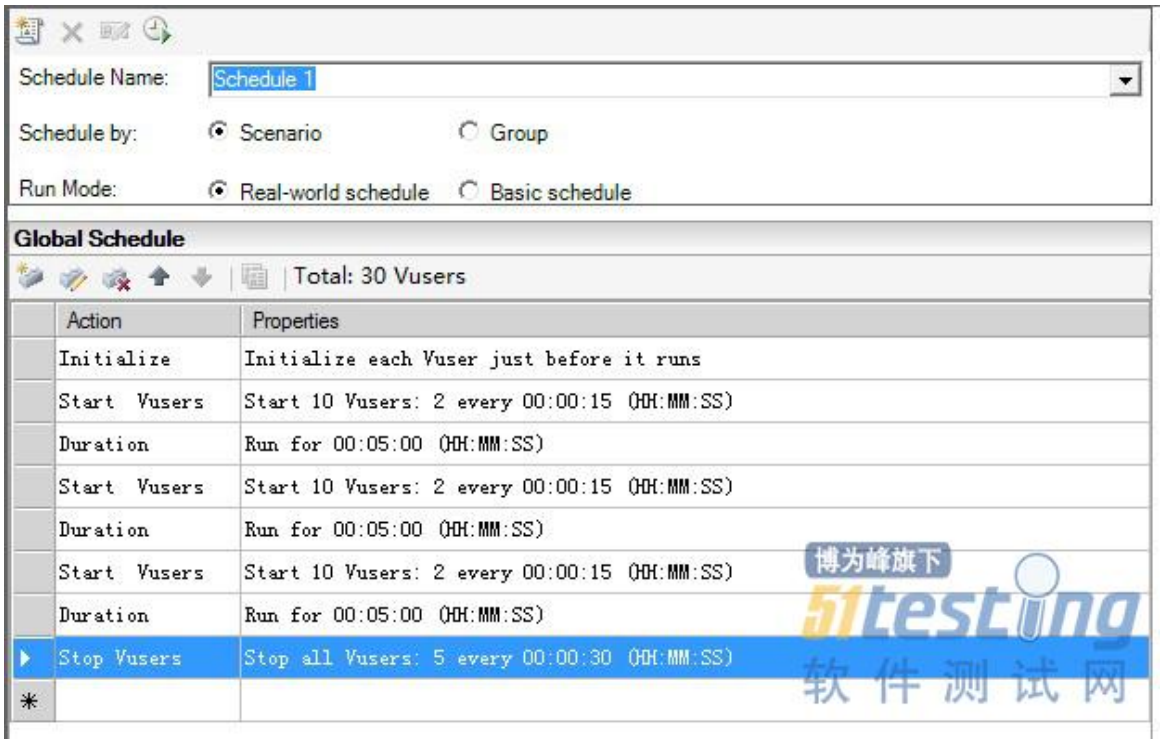
5.2、设置 LR 场景

5.2.1、手工场景



手动场景设计可以通过建立组并指定脚本、负载生成器和每个组中包括的 Vuser 数来建立手动场景；

还可以通过百分比模式建立手动场景，使用此方法建立场景可以指定场景中将使用的 Vuser 的总数，并为每个脚本分配负载生成器和占总数一定百分比的 Vuser。



脚本执行模式：

1) Real-worldschedule：按照场景设计进行加压，可实现阶梯加压和峰值压测。

（PS：在 real-world 模式中，Duration 的优先级高于 run-timesetting 的设置）

2) Basicschedule：按照脚本设计的迭代方式执行，只能进行峰值压测。

3) Scheduleby:

- Scenario(场景)：多个脚本组合场景的时候根据设计的场景运行方式，所有脚本按统一方式执行。
- Group(组)：多个脚本组合场景的时候，单独设计每个脚本的执行方式。

(PS：我们常用的场景设置对应我们的性能测试需求时如下：

1)基准测试：

采用一个虚拟用户迭代执行 N 次，取测试结果的平均值作为基准参考值。

2)单交易负载

采用多个用户并发的方式对单交易进行负载测试，分析性能指标是否满足需求。使用峰值加压，可以测试特定峰值下系统性能情况。使用梯队加压，可以测试系统的最大并发用户数和最佳并发用户数。

建议：在单交易负载测试时，采用峰值加压和梯度加压相结合的方式进行测试。

3)综合场景负载

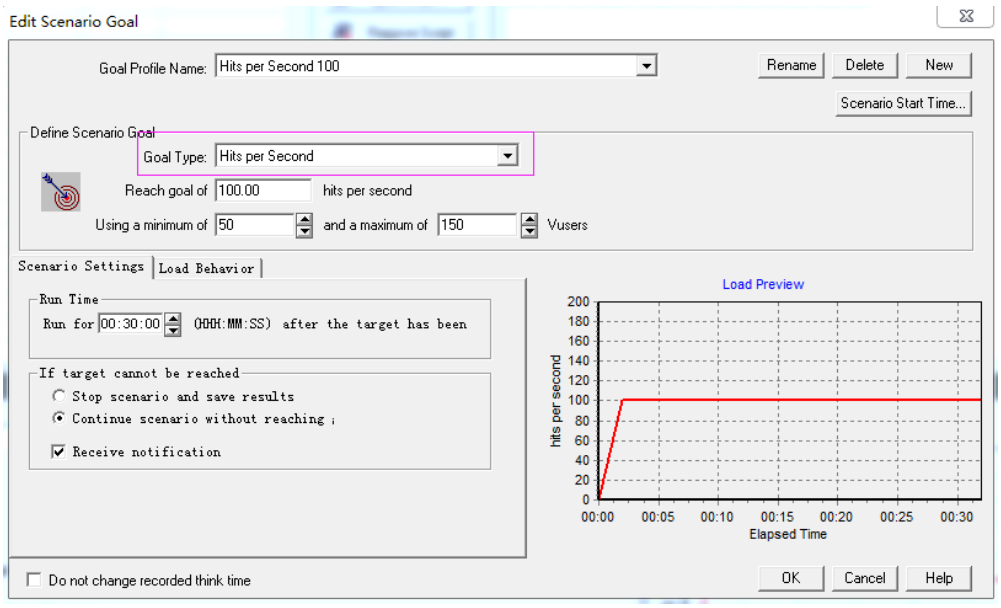
模拟真实交易场景，分析正常业务交易量，选取日交易量最多的几个交易，合理分配并发用户数，持续运行一段时间，分析测试结果是否满足需求。

4)稳定性测试

根据需求选取几个常用交易，按照峰值压力的百分比进行加压，稳定运行一段时间，查看系统运行是否正常，系统资源利用率是否在最佳状态。

)

5.2.2、面向目标的场景



在面向目标的场景中，可以定义要实现的测试目标，LoadRunner 会根据这些目标自动构建场景。

可以在一个面向目标的场景中定义希望场景达到的下列 5 种类型的目标：虚拟用户数、每秒单击次数（仅 WebVuser）、每秒事务数、每分钟页面数（仅 WebVuser）或事务响应时间。可以通过单击【EditScenarioGoal...】设置，如目标类型、最小用户数、最大用户数、运行时间等。

(PS：当我们的性能测试需求非常明确的时候，我们可以采用面向目标的模式来验证系统服务的级别；也可来验证我们的优化结果是否满足要求)

5.3、多 PC 联机负载

(PS：当我们需要模拟大量 Vusers 时需要多台负载机来生成负载，避免负载机成为瓶颈，从而更精确地模拟并发场景)

1、安装，在需要添加为负载机的计算机上安装 loadrunner11

2 启动 agentHPloadrunner-advancedsetting-loadrunneragaentprocess 启动 LRagaent。

3、关闭负载机的防火墙

4、添加负载机，在场景所在的机器添加负载机为当前场景压力机，录入负载机的 ip、操作系统等信息（操作系统和临时目录可以不录入采用默认），点击 more：出现‘负载生成器设置选项卡’

5、连接负载机，点击 connect 按钮连接负载机，status 列变为 ready 表示负载机可用，列头表示的是负载机的资源使用情况，如果表示为绿色表示有空余的资源，红色表

示服务器忙碌。

6、为脚本配置对应压力机，在 group 里面可以为每一个脚本配置对应的压力机，配置方法是点击 loadgenerator 选择压力机。

5.4、服务器监控

5.4.1、监控数据库

(PS: 极力推荐 spotlight 系列的工具，高端专业监控 Oracle/MySQL 数据库。
当然我们也可以自己写 shell 或者 python 脚本来监控数据库)。

我们一起来看下 Oracle 数据库的常用性能监控指标:

--监控事例的等待--

```
selectevent,
sum(decode(wait_Time,0,0,1))"Prev",
sum(decode(wait_Time,0,1,0))"Curr",
count(*)"Tot"
fromv$session_Wait
groupbyevent
orderby4;
```

--回滚段的争用情况--

```
selectname,waits,gets,waits/gets"Ratio"
fromv$rollstata,v$rollnameb
wherea.usn=b.usn;
```

--监控表空间的 I/O 比例--

```
selectdf.tablespace_namename,
df.file_name"file",
f.phyrdspyr,
f.phyblkrdpbr,
f.phywrtspyw,
f.phyblkwrtpbw
fromv$filestatf,dba_data_filesdf
```

```
wheref.file#=df.file_id
orderbydf.tablespace_name;

--监控文件系统的 I/O 比例--
selectsubstr(a.file#,1,2)"#",
substr(a.name,1,30)"Name",
a.status,
a.bytes,
b.phyrds,
b.phywrts
fromv$datafilea,v$filestatb
wherea.file#=b.file#;

--在某个用户下找所有的索引--
selectuser_indexes.table_name,
user_indexes.index_name,
uniqueness,
column_name
fromuser_ind_columns,user_indexes
whereuser_ind_columns.index_name=user_indexes.index_name
anduser_ind_columns.table_name=user_indexes.table_name
orderbyuser_indexes.table_type,
user_indexes.table_name,
user_indexes.index_name,
column_position;

--监控 SGA 的命中率--
selecta.value+b.value"logical_reads",
c.value"phys_reads",
round(100*((a.value+b.value)-c.value)/(a.value+b.value))"BUFFERHITRATIO"
fromv$sysstata,v$sysstatb,v$sysstatc
wherea.statistic#=38
```

```
andb.statistic#=39
andc.statistic#=40;

 --监控 SGA 中字典缓冲区的命中率--
selectparameter,
gets,
Getmisses,
getmisses/(gets+getmisses)*100"missratio",
(1-(sum(getmisses)/(sum(gets)+sum(getmisses))))*100"Hitratio"
fromv$rowcache
wheregets+getmisses<>0
groupbyparameter,gets,getmisses;

 --监控 SGA 中共享缓存区的命中率，应该小于 1%--
selectsum(pins)"TotalPins",
sum(reloads)"TotalReloads",
sum(reloads)/sum(pins)*100libcache
fromv$librarycache;

selectsum(pinhits-reloads)/sum(pins)"hitradio",
sum(reloads)/sum(pins)"reloadpercent"
fromv$librarycache;

 --监控 SGA 中重做日志缓存区的命中率，应该小于 1%--
SELECTname,
gets,
misses,
immediate_gets,
immediate_misses,
Decode(gets,0,0,misses/gets*100)ratio1,
Decode(immediate_gets+immediate_misses,
0,
```

```
0,
immediate_misses/(immediate_gets+immediate_misses)*100)ratio2
FROMv$latch
WHEREnameIN('redoallocation','redocopy');

--监控内存和硬盘的排序比率增加 sort_area_size--
SELECTname,value
FROMv$sysstat
WHEREnameIN('sorts(memory)','sorts(disk)');

--监控当前数据库谁在运行什么 SQL 语句--
SELECTosuser,username,sql_text
fromv$sessiona,v$sqltextb
wherea.sql_address=b.address
orderbyaddress,piece;

--监控字典缓冲区--
SELECT(SUM(PINS-RELOADS))/SUM(PINS)"LIBCACHE"FROMV$LIBRARYCACHE;
SELECT(SUM(GETS-GETMISSES-USAGE-
FIXED))/SUM(GETS)"ROWCACHE"FROMV$ROWCACHE;
SELECTSUM(PINS)"EXECUTIONS",SUM(RELOADS)"CACHEMISSESWHILEEXECUTING
"FROMV$LIBRARYCACHE;

----此值大于 0.5 时，参数需加大--
selectsum(wait)/sum(totalq)"dispatcherwaits"fromv$queuewheretype='dispatcher';
selectcount(*)fromv$dispatcher;
selectservers_highwaterfromv$mts;
servers_highwater 接近 mts_max_servers 时，参数需加大

---碎片程度--
selecttablespace_name,count(tablespace_name)
```

```
fromdba_free_space
groupbytablespace_name
havingcount(tablespace_name)>10;
altertablespacenamecoalesce;
altertablenamedeallocateunused;
createorreplaceviewts_blocks_vas
selecttablespace_name,block_id,bytes,blocks,'freespace'segment_namefromdba_free_space
unionall
selecttablespace_name,block_id,bytes,blocks,segment_namefromdba_extents;
select*fromts_blocks_v;
selecttablespace_name,sum(bytes),max(bytes),count(block_id)
fromdba_free_space
groupbytablespace_name;

--查看碎片程度高的表--
SELECTsegment_nametable_name,COUNT(*)extents
FROMdba_segments
WHEREownerNOTIN('SYS','SYSTEM')
GROUPBYsegment_name
HAVINGCOUNT(*)=(SELECTMAX(COUNT(*))
FROMdba_segments
GROUPBYsegment_name);

---表、索引的存储情况检查--
selectsegment_name,sum(bytes)space,count(*)ext_quan
fromdba_extents
wheretablespace_name='&tablespace_name'
andsegment_type='TABLE'
groupbytablespace_name,segment_name;

selectsegment_name,count(*)
fromdba_extents
```

```
wheresegment_type='INDEX'
andowner='&owner'
groupbysegment_name;

--检测数据库中的事件和等待--
SELECTevent,total_waits,total_timeouts,time_waited,average_waitFROMv$system_event

--查询会话中的事件和等待时间--
selectsid,event,total_waits,average_waitfromv$session_eventwheresid=10;

--查询等待进程--
SELECTsid,seq#,event,wait_time,stateFROMv$session_wait;

---当前 sql 语句--
selectsql_text,users_executing,executions,loadsfromv$sqlarea;

--查询高速缓存中的命中率--
selectsum(pins)"Executions",
sum(reloads)"CacheMisses",
sum(reloads)/sum(pins)
fromv$librarycache;

--cpuusedbythissession--
selecta.sid,
spid,
status,
substr(a.program,1,40)prog,
a.terminal,
oSUSEr,
value/60/100value
```

```
fromv$sessiona,v$processb,v$sesstatc
wherec.statistic#=12
andc.sid=a.sid
anda.paddr=b.addr
orderbyvaluedesc;

--查看表锁--
select*fromsys.v_$sqlareawheredisk_reads>100

--监控事例的等待--
selectevent,
sum(decode(wait_Time,0,0,1))"Prev",
sum(decode(wait_Time,0,1,0))"Curr",
count(*)"Tot"
fromv$session_Wait
groupbyevent
orderby4

--查看前台正在发出的 SQL 语句--
selectuser_name,sql_text
fromv$open_cursor
wheresidin(selectsid
from(selectsid,serial#,username,program
fromv$session
wherestatus='ACTIVE'))

--数据表占用空间大小情况--
selectsegment_name,tablespace_name,bytes,blocks
fromuser_segments
wheresegment_type='TABLE'
ORDERBYbytesDESC,blocksDESC
```

```
--查看表空间碎片大小--
selecttablespace_name,
round(sqrt(max(blocks)/sum(blocks))*
(100/sqrt(sqrt(count(blocks)))),
2)FSFI
fromdba_free_space
groupbytablespace_name
orderby1

--查看表空间占用磁盘情况--
selectb.file_id 文件 ID 号,
b.tablespace_name 表空间名,
b.bytes 字节数,
(b.bytes-sum(nvl(a.bytes,0)))已使用,
sum(nvl(a.bytes,0))剩余空间,
sum(nvl(a.bytes,0))/(b.bytes)*100 剩余百分比
fromdba_free_spacea,dba_data_filesb
wherea.file_id=b.file_id
groupbyb.tablespace_name,b.file_id,b.bytes
orderbyb.file_id

--查看 session 使用回滚段--
SELECTr.name 回滚段名,
s.sid,
s.serial#,
s.username 用户名,
t.status,
t.cr_get,
t.phy_io,
t.used_ublk,
t.noundo,
```

```
substr(s.program,1,78)操作程序
FROMsys.v_$sessions,sys.v_$transactiont,sys.v_$rollnamer
WHEREt.addr=s.taddrandt.xidusn=r.usn
ORDERBYt.cr_get,t.phy_io

--查看 SGA 区剩余可用内存--
selectname,
sgasize/1024/1024"Allocated(M)",
bytes/1024"**空间(K)",
round(bytes/sgasize*100,2)"**空间百分比(%)"
from(selectsum(bytes)sgasizefromsys.v_$sgastat)s,sys.v_$sgastatf
wheref.name='freememory'

---非系统用户建在 SYSTEM 表空间中的表--
SELECTowner,table_name
FROMDBA_TABLES
WHEREtablespace_namein('SYSTEM','USER_DATA')AND
ownerNOTIN('SYSTEM','SYS','OUTLN','ORDSYS','MDSYS','SCOTT','HOSTEAC')

---性能最差的 SQL--

SELECT*FROM(SELECTPARSING_USER_IDEXECUTIONS,SORTS,COMMAND_TYPE,DISK_READS,sql_text
FROMv$sqlarea
ORDERBYdisk_readsDESC)
WHEREROWNUM<100

---读磁盘数超 100 次的 sql--
select*fromsys.v_$sqlareawheredisk_reads>100

---最频繁执行的 sql--
```

```
select*fromsys.v_$sqlareawhereexecutions>100

---查询使用 CPU 多的用户 session--
selecta.sid,spid,status,substr(a.program,1,40)prog,a.terminal,osuser,value/60/100value
fromv$sessiona,v$processb,v$sesstatc
wherec.statistic#=12and
c.sid=a.sidand
a.paddr=b.addr
orderbyvaluedesc
```

5.4.2、监控 OS

Windows 上的可以使用 python 脚本来监控，Linux 上的极力推荐 nmon 谁用谁知道哈~

至于 OS 的性能指标，大家自行谷歌即可~

强调一点：我们没必要非要用 LoadRunner 进行 OS 的监控~方法很多自行摸索

LinuxOS 的常用性能指标:

度量	描述
Averageload	上一分钟同时处于“就绪”状态的平均进程数
Collisionrate	每秒钟在以太网上检测到的冲突数
Contextswitchesrate	每秒钟在进程或线程之间的切换次数
CPUutilization	CPU 的使用时间百分比
Diskrate	磁盘传输速率
Incomingpacketserrorrate	接收以太网数据包时每秒钟接收到的错误数
Incomingpacketsrate	每秒钟传入的以太网数据包数
Interruptrate	每秒内的设备中断数
Outgoingpacketserrorsrate	发送以太网数据包时每秒钟发送的错误数
Outgoingpacketsrate	每秒钟传出的以太网数据包数
Page-inrate	每秒钟读入到物理内存中的页数
Page-outrate	每秒钟写入页面文件和从物理内存中删除的页数
Pagingrate	每秒钟读入物理内存或写入页面文件的页数
Swap-inrate	正在交换的换入进程数
Swap-outrate	正在交换的换出进程数
SystemmodeCPUutilization	在系统模式下使用 CPU 的时间百分比
UsermodeCPUutilization	在用户模式下使用 CPU 的时间百分比

WindowsOS 常用的性能指标:

对象	度量	描述
System	%TotalProcessorTime	系统上所有处理器都忙于执行非空闲线程的时间的平均百分比。 在多处理器系统上，如果所有处理器始终繁忙，此值为 100%，如 果所有处理器为 50%繁忙，此值为 50%，而如果这些处理器中的四分 之一是 100%繁忙的，则此值为 25%。 它反映了用于有用作业上的时间 的比率。每个处理器将分配给空闲进程 中的一个空闲线程，它将消耗 所有其他线程不使用的那些非生产性处理器周期
Processor	%ProcessorTime （Windows2000）	处理器执行非空闲线程的时间百分比。 该计数器设计为处理器活动的 一个主要指示器。它是通过测量处理器 在每个采样间隔中执行空闲进 程的线程所花费的时间，然后从 100% 中减去此时间值来进行计算的 （每个处理器都有一个空闲线程，它在 没有其他线程准备运行时消 耗处理器周期）。它可以反映有用作业占 用的采样间隔的百分比。 该计数器显示在采样期间所观察到的繁忙时间的平均百分比。 它是通过监控服务处于非活动状态的时间值， 然后从 100%中减去此值来进行计算的

对象	度量	描述
System	FileDataOperations/sec	计算机对文件系统设备执行读取和写入操作的速率。 这不包括文件控制操作
System	ProcessorQueueLength	以线程数计的处理器队列的即时长度。 如果不同时监控线 程计数，则此计数始终为 0。所有处理器都使用一个队列， 而线程在该队列中等待处理器进行循环调用。此长度不包括 当前正在执行的线程。一般情况下，如果处理器队列的长度一 直超过 2，则可能表示处理器堵塞。此值为即时计数，不是一段时间的平均值
Memory	PageFaults/sec	此值为处理器中的页面错误的计数。当

		进程引用特定的虚拟内存页，该页不在主内存的工作集当中时，将出现页面错误。如果某页位于待机列表中（因此它已经位于主内存中），或者它正在被共享该页的其他进程所使用，则页面错误不会导致该页从磁盘中提取出
PhysicalDisk	%DiskTime	选定的磁盘驱动器对读写请求提供服务的已用时间所占百分比
Memory	PoolNonpagedBytes	非分页池中的字节数，指可供操作系统组件完成指定任务后从其中获得空间的系统内存区域。非分页池页面不可以退出到分页文件中。它们自分配以来就始终位于主内存中
Memory	Pages/sec	为解决引用时不在内存中的页面的内存引用，从磁盘读取的或写入磁盘的页数。这是 PagesInput/sec 和 PagesOutput/sec 的和。此计数器中包括的页面流量代表着用于访问应用程序的文件数据的系统缓存。此值还包括存入/取自非缓存映射内存文件的页数。如果关心内存压力过大问题（即系统失效）和可能产生的过多分页，则这是值得观察的主要计数器
System	TotalInterrupts/sec	计算机接收并处理硬件中断的速度。可能生成中断的设备有系统时钟、鼠标、数据通信线路、网络接口卡和其他外围设备。此计数指示这些设备在计算机上所处的繁忙程度
Objects	Threads	计算机在收集数据时的线程数。注意，这是一个即时计数，不是一段时间的平均值。线程是能够执行处理器指令的基础可执行实体
Process	PrivateBytes	专为此进程分配，无法与其他进程共享的当前字节数

5.4.3、监控中间件

5.4.3.1、监控 tomcat

使用 LoadRunner 自带的 lr_user_data_point 监控 tomcat

1、配置 Tomcat 登录用户，找到 tomcat 安装目录下的/conf/tomcat-users.xml，添加配置如下:

```
<tomcat-users>
<rolerolename="manager-gui"/>
<userusername="用户名"password="密码"roles="manager-gui"/>
</tomcat-users>
```

（配置 Tomcat 登录用户后，建议测试一下配置的用户登录能否登录进入 Tomcat 管理页面）

2、在 Action 脚本中，使用 web_set_user("用户名","密码","tomcat 服务器所在的 IP 地址:端口");

3、脚本中编写 web_url();模拟访问 Tomcat 的 url 并登录

4、利用关联函数 web_reg_save_parm()动态地捕获想要的数据

5、最后利用打点函数 lr_user_data_point("监控指标名","监控指标值");记录用户自定义的数据样本

VuGen 脚本代码如下:

```
lr_start_transaction("");
lr_end_transaction();
web_reg_save_param("JVMFreeMemory",
"LB=Freememory:",
"RB=MB",
"Ord=1",
LAST);
web_reg_save_param("JVMTotalMemory",
"LB=Totalmemory:",
"RB=MB",
"Ord=1",
LAST);
web_reg_save_param("JVMMaxMemory",
"LB=Maxmemory:",
"RB=MB",
"Ord=1",
LAST);
```

```
web_url("status",
"URL=http://{ServerName}/manager/status",
"Resource=0",
"RecContentType=text/html",
"Referer=",
"Snapshot=t1.inf",
"Mode=HTTP",
LAST);

lr_user_data_point("TomcatJVMFreememory",atof(lr_eval_string("{JVMFreeMemory}")));
lr_user_data_point("TomcatJVMTotalmemory",atof(lr_eval_string("{JVMTotalMemory}")));
lr_user_data_point("TomcatJVMMaxmemory",atof(lr_eval_string("{JVMMaxMemory}")));
```

(PS:需要注意的是，我们有两种方法来使用 lr_user_data_point，1：将自定义时间和我们的业务请求放到一个 Action 中；2：将自定义时间单独放在一个 Action 中，并将这个 action 放到业务处理的 Action 之后)

5.4.3.2、监控 Nginx

1) 使用 ngxtop 监控 Nginx

ngxtop 默认会从其配置文件(/etc/nginx/nginx.conf)中查找 Nginx 日志的地址。所以，监控 Nginx，运行以下命令即可：

```
1        $ngxtop
```

将会列出 10 个 Nginx 服务，按请求数量排序。

显示前 20 个最频繁的请求：

```
1        $ngxtop-n20
```

```
running for 516 seconds, 7549 records processed: 14.63 req/sec

Summary:
|   count |   avg_bytes_sent |   2xx |   3xx |   4xx |   5xx |
|---------+------------------+-------+-------+-------+-------|
|    7549 |         8503.166 |  7517 |     0 |    32 |     0 |

Detailed:
| request_path                 |   count |   avg_bytes_sent |   2xx |   3xx |   4xx |   5xx |
|------------------------------+---------+------------------+-------+-------+-------+-------|
| /                            |    2201 |        15423.605 |  2192 |     0 |     9 |     0 |
| /archives/25                 |    1484 |         3593.412 |  1478 |     0 |     6 |     0 |
| /archives/10                 |     914 |         3801.291 |   910 |     0 |     4 |     0 |
| /archives/1                  |     744 |        15343.065 |   740 |     0 |     4 |     0 |
| /archives/21                 |     735 |         3598.182 |   733 |     0 |     2 |     0 |
| /archives/8                  |     559 |         3663.435 |   558 |     0 |     1 |     0 |
| /archives/23                 |     303 |         3605.046 |   301 |     0 |     2 |     0 |
| /archives/12                 |     301 |         4539.867 |   300 |     0 |     1 |     0 |
| /archives/category/news      |     148 |        14536.115 |   147 |     0 |     1 |     0 |
| /archives/date/2014/06       |     137 |         4336.117 |   136 |     0 |     1 |     0 |
| /wp-admin/admin-ajax.php     |      14 |           56.143 |    14 |     0 |     0 |     0 |
| /wp-admin/edit.php           |       3 |        12599.000 |     3 |     0 |     0 |     0 |
| /wp-admin/                   |       2 |        12675.000 |     2 |     0 |     0 |     0 |
| /aaa                         |       1 |         4157.000 |     0 |     0 |     1 |     0 |
| /wp-admin/index.php          |       1 |        12679.000 |     1 |     0 |     0 |     0 |
| /wp-admin/options-general.php |       1 |        12487.000 |     1 |     0 |     0 |     0 |
| /wp-admin/post.php           |       1 |        30797.000 |     1 |     0 |     0 |     0 |
```

2) 获取 Nginx 基本信息：

```
1    $ngxtopinfo
```

```
dev@ubuntu:~$ ngxtop info
nginx configuration file:
  /etc/nginx/nginx.conf
access log file:
  /var/log/nginx/access.log
access log format:
  $remote_addr - $remote_user [$time_local] "$request" $status $body_bytes_sent
"$http_referer" "$http_user_agent"
available variables:
  body_bytes_sent, http_referer, http_user_agent, remote_addr, remote_user, requ
est, status, time_local
dev@ubuntu:~$
```

你可以自定义显示的变量，简单列出需要显示的变量。使用“print”命令显示自定义请求。

```
1    $ngxtopprintrequesthttp_user_agentremote_addr
```

```
running for 14 seconds, 219 records processed: 15.53 req/sec

request, http_user_agent, remote_addr:
| request                          | http_user_agent                                | remote_addr     |
|----------------------------------+------------------------------------------------+-----------------|
| GET / HTTP/1.1                   | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.128  |
| GET / HTTP/1.1                   | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.138  |
| GET / HTTP/1.1                   | Mozilla/5.0 (pc-x86_64-linux-gnu) Siege/3.0.5  | 172.16.253.1    |
| GET /archives/1 HTTP/1.1         | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.128  |
| GET /archives/1 HTTP/1.1         | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.138  |
| GET /archives/1 HTTP/1.1         | Mozilla/5.0 (pc-x86_64-linux-gnu) Siege/3.0.5  | 172.16.253.1    |
| GET /archives/10 HTTP/1.1        | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.128  |
| GET /archives/10 HTTP/1.1        | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.138  |
| GET /archives/10 HTTP/1.1        | Mozilla/5.0 (pc-x86_64-linux-gnu) Siege/3.0.5  | 172.16.253.1    |
| GET /archives/12 HTTP/1.1        | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.128  |
| GET /archives/12 HTTP/1.1        | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.138  |
| GET /archives/12 HTTP/1.1        | Mozilla/5.0 (pc-x86_64-linux-gnu) Siege/3.0.5  | 172.16.253.1    |
| GET /archives/21 HTTP/1.1        | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.128  |
| GET /archives/21 HTTP/1.1        | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.138  |
| GET /archives/21 HTTP/1.1        | Mozilla/5.0 (pc-x86_64-linux-gnu) Siege/3.0.5  | 172.16.253.1    |
| GET /archives/23 HTTP/1.1        | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.138  |
| GET /archives/23 HTTP/1.1        | Mozilla/5.0 (pc-x86_64-linux-gnu) Siege/3.0.5  | 172.16.253.1    |
| GET /archives/25 HTTP/1.1        | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.128  |
| GET /archives/25 HTTP/1.1        | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.138  |
| GET /archives/25 HTTP/1.1        | Mozilla/5.0 (pc-x86_64-linux-gnu) Siege/3.0.5  | 172.16.253.1    |
| GET /archives/8 HTTP/1.1         | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.128  |
| GET /archives/8 HTTP/1.1         | JoeDog/1.00 [en] (X11; I; Siege 3.0.1)         | 172.16.253.138  |
```

显示请求最多的客户端 IP 地址

```
1    $ngxtoptopremote_addr
```

```
running for 432 seconds, 5714 records processed: 13.22 req/sec

top remote_addr
| remote_addr     |   count |
|-----------------+---------|
| 172.16.253.1    |    3074 |
| 172.16.253.138  |    1589 |
| 172.16.253.128  |    1051 |
```

显示状态码是 404 的请求

```
1    $ngxtop-i'status==404'printrequeststatus
```

```
running for 84 seconds, 4 records processed: 0.05 req/sec

request, status:
| request                                |   status |
|----------------------------------------+----------|
| GET /archive/101 HTTP/1.1              |      404 |
| GET /archive/120 HTTP/1.1              |      404 |
| GET /archives/15555 HTTP/1.1           |      404 |
| GET /archives/category/ubuntu HTTP/1.1 |      404 |
```

除了 Nginx，ngtop 还可以处理其他的日志文件，比如 Apache 的访问文件。使用以下命令监控 Apache 服务器

```
1    $tail-f/var/log/apache2/access.log|ngxtop-fcommon
```

5.4.4、监控网络

个人推荐 Zabbix

zabbix 是一个基于 WEB 界面的提供分布式系统监视以及网络监视功能的企业级的开源解决方案。它能监视各种网络参数，保证服务器系统的安全运营；并提供灵活的通知机制以让系统管理员快速定位/解决存在的各种问题。

zabbix 由两部分构成，zabbixserver 与可选组件 zabbixagent。

zabbixserver 可以通过 SNMP，zabbixagent，ping，端口监视等方法提供对远程服务器/网络状态的监视，数据收集等功能，它可以运行在 Linux，Solaris，HP-UX，AIX，FreeBSD，OpenBSD，OSX 等平台上。

Zabbix 主要功能:

-CPU 负荷

-内存使用

-磁盘使用

-网络状况

-端口监视

-日志监视。

Zabbix 的使用啥的这儿不多说了谁用谁知道~~

六、测试报告编写

(ps: 测试报告不仅仅是给领导看的尽量把图表做的漂亮些)

6.1、测试目标

1.1.1: 评估禅道(开源版 8.0.1)是否满足公司使用。

1.1.2: 评估禅道(开源版 8.0.1)最大负载。

6.2、参考文档

《禅道开源版操作手册》

6.3、测试环境说明

6.3.1、硬件配置

服务器名称	配置/详细信息	数量	IP
Web&数据库服务器	Cpu:i5 Disk:500G Memory: 8G	1	192.168.10.206
负载机	Cpu:i3 Disk:500G Memory: 8G DB:MySQL5.5	1	192.168.10.188
负载机	Cpu:i5 Disk:500G Memory:8G Net:100Mbps 局域网	1	192.168.10.118

6.3.2、软件配置

序号	软件名称	Web 服务器	数据库服务器	负载机
1	操作系统	Win7 旗舰版 SP1	Win7 旗舰版 SP1	Win7 旗舰版 SP1
2	数据库	--	MySQL5.5	--
3	浏览器	--	--	IE11Chrome45
4	WebServer	ApacheApache2.4	--	--
5	Application	禅道(开源版 8.0.1)	--	--
6	其他工具			HttpWatch9.4 LoadRunner11

6.4、测试策略

6.4.1、人力资源

测试轮次	测试时间		测试人员
	起始时间	结束时间	
第 1 轮测试	2016-01-15	2016-01-29	姜林斌
第 2 轮测试	2016-02-14	2016-02-19	姜林斌
第 3 轮测试	2016-02-22	2016-02-28	姜林斌

6.4.2、测试方案

测试过程按两个步骤进行，即单独场景压力测试、负载测试。

单独场景压力测试：针对某个功能点进行压力测试，分析测试结果是否满足用户要求的指标;

负载测试：选择某些业务场景对系统持续增压测力，持续运行一段时间，根据并发

量或系统监控等来观察系统的最大负载(响应时间<5s&CPU 使用率<80%&内存使用率<80%)。

6.4.3、测试场景

具体请参考《禅道性能测试方案》

6.4.4、测试用例

6.4.4.1、团队需求符合性验证基准测试用例

- **登录禅道(LoginZentao)**

用例名称	登录禅道	用例编号	001
测试步骤	1: 部署性能测试环境 2: 用 loadrunner 录制脚本使用 PortMapping 策略录制脚本 3: 设置登录事件为事务“登录禅道”		
场景设计	1、设计用户数量为 10 2、StartUsers: 每隔 30 秒自动增加 1 个用户登录系统，直到增加至 10 个 3、Duration 方案: 在 10Vusers 负载下持续压测 10Mins 4、StartUsers: 每隔 30 秒自动增加 1 个用户登录系统，直到增加至 20 个 5、Duration 方案: 在 20Vusers 负载下持续压测 10Mins 6、StopUsers: 每隔 30 秒停止 1 个用户，直到停止所有用户		
执行时间	38 分钟		
预期结果	1、事务响应时间平均值不能超过 5 秒 2、CPU 使用率平均值不能高于 75% 3、物理内存使用率不超过 70%		

- **添加用例(AddTestCase)**

用例名称	添加测试用例	用例编号	002
测试步骤	1: 部署性能测试环境 2: 用 loadrunner 录制脚本使用 PortMapping 策略录制脚本 3: 设置登录事件为事务“添加测试用例”		
场景设计	1、设计用户数量为 10 2、StartUsers: 每隔 30 秒自动增加 1 个用户，直到增加至 10 个 3、Duration 方案: 在 10Vusers 负载下持续压测 10Mins 4、StartUsers: 每隔 30 秒自动增加 1 个用户，直到增加至 20 个 5、Duration 方案: 在 20Vusers 负载下持续压测 10Mins 6、StopUsers: 每隔 30 秒停止 1 个用户，直到停止所有用户		
执行时间	38 分钟		

预期结果	1、事务响应时间平均值不能超过 5 秒 2、CPU 使用率平均值不能高于 75% 3、物理内存使用率不超过 70%

- **执行用例(ExecuateCase)**

用例名称	执行用例	用例编号	003
测试步骤	1: 部署性能测试环境 2: 用 loadrunner 录制脚本使用 PortMapping 策略录制脚本 3: 设置测试执行用例的事件为事务“执行用例”		
场景设计	1、设计用户数量为 10 2、StartUsers: 每隔 30 秒自动增加 1 个用户，直到增加至 10 个 3、Duration 方案: 在 10Vusers 负载下持续压测 10Mins 4、StartUsers: 每隔 30 秒自动增加 1 个用户，直到增加至 20 个 5、Duration 方案: 在 20Vusers 负载下持续压测 10Mins 6、StopUsers: 每隔 30 秒停止 1 个用户，直到停止所有用户		
执行时间	38 分钟		
预期结果	1、事务响应时间平均值不能超过 5 秒 2、CPU 使用率平均值不能高于 75% 3、物理内存使用率不超过 70%		

- **提交 Bug(提交 Bug)**

用例名称	提交 Bug	用例编号	004
测试步骤	1: 部署性能测试环境 2: 用 loadrunner 录制脚本使用 PortMapping 策略录制脚本 3: 设置测试新建 bug 的事件为事务“提交 Bug”		
场景设计	1、设计用户数量为 10 2、StartUsers: 每隔 30 秒自动增加 1 个用户登录系统，直到增加至 10 个 3、Duration 方案: 在 10Vusers 负载下持续压测 10Mins 4、StartUsers: 每隔 30 秒自动增加 1 个用户登录系统，直到增加至 20 个 5、Duration 方案: 在 20Vusers 负载下持续压测 10Mins 6、StopUsers: 每隔 30 秒停止 1 个用户，直到停止所有用户		
执行时间	38 分钟		
预期结果	1、事务响应时间平均值不能超过 5 秒 2、CPU 使用率平均值不能高于 75% 3、物理内存使用率不超过 70%		

- **解决 Bug(FixBug)**

用例名称	解决 Bug	用例编号	005
测试步骤	1: 部署性能测试环境 2: 用 loadrunner 录制脚本使用 PortMapping 策略录制脚本 3: 设置开发更改 bug 为 fixed 状态的事件为事务“解决 Bug”		
场景设计	1、设计用户数量为 10 2、StartUsers: 每隔 30 秒自动增加 1 个用户登录系统，直到增加		

	至 10 个 3、Duration 方案：在 10Vusers 负载下持续压测 10Mins 4、StartUsers：每隔 30 秒自动增加 1 个用户登录系统，直到增加至 20 个 5、Duration 方案：在 20Vusers 负载下持续压测 10Mins 6、StopUsers：每隔 30 秒停止 1 个用户，直到停止所有用户
执行时间	38 分钟
预期结果	1、事务响应时间平均值不能超过 5 秒 2、CPU 使用率平均值不能高于 75% 3、物理内存使用率不超过 70%

- **关闭 Bug(CloseCase)**

用例名称	关闭 Bug	用例编号	006
测试步骤	1：部署性能测试环境 2：用 loadrunner 录制脚本使用 PortMapping 策略录制脚本 3：设置测试更改 bug 为 close 状态的事件为事务“关闭 Bug”		
场景设计	1、设计用户数量为 10 2、StartUsers：每隔 30 秒自动增加 1 个用户登录系统，直到增加至 10 个 3、Duration 方案：在 10Vusers 负载下持续压测 10Mins 4、StartUsers：每隔 30 秒自动增加 1 个用户登录系统，直到增加至 20 个 5、Duration 方案：在 20Vusers 负载下持续压测 10Mins 6、StopUsers：每隔 30 秒停止 1 个用户，直到停止所有用户		
执行时间	38 分钟		
预期结果	1、事务响应时间平均值不能超过 5 秒 2、CPU 使用率平均值不能高于 75% 3、物理内存使用率不超过 70%		

- **确认 Bug**

用例名称	查看待办事项	用例编号	007
测试步骤	1：部署性能测试环境 2：用 loadrunner 录制脚本使用 PortMapping 策略录制脚本		
场景设计	1、设计每组虚拟用户数量为 20 2、Duration：持续压测 20 分钟 3、StopUsers：每隔 30 秒停止 1 个用户，直到停止所有用户		
执行时间	38 分钟		
预期结果	1、事务响应时间平均值不能超过 5 秒 2、CPU 使用率平均值不能高于 75% 3、物理内存使用率不超过 70%		

6.4.4.2、负载测试

负载测试选取的业务场景如下：

1. 登录系统

用例名称	登录禅道	用例编号	008
测试步骤	1: 部署性能测试环境 2: 用 loadrunner 录制脚本使用 PortMapping 策略录制脚本		
场景设计	场景模式：使用百分比模式设置每台负载机承受 50%的负载(Real-World 模式)。 第一轮测试: 1 初始化 100 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 120 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 140 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第二轮测试: 1 初始化 140 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 160 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 180 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第三轮测试: 1 初始化 180 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 200 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 220 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第四轮测试: 1 初始化 220 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 240 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 260 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。		

2. 添加用例

用例名称	添加测试用例	用例编号	009
测试步骤	1: 部署性能测试环境 2: 用 loadrunner 录制脚本使用 PortMapping 策略录制脚本		
场景设计	场景模式：使用百分比模式设置每台负载机承受 50%的负载(Real-World 模式)。 第一轮测试: 1 初始化 100 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 120 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 140 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第二轮测试: 1 初始化 140 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 160 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 180 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第三轮测试:		

	1 初始化 180 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 200 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 220 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。

3. 执行用例

用例名称	执行测试用例	用例编号	010
测试步骤	1: 部署性能测试环境 2: 用 loadrunner 录制脚本使用 PortMapping 策略录制脚本		
场景设计	场景模式: 使用百分比模式设置每台负载机承受 50%的负载(Real-World 模式)。 第一轮测试: 1 初始化 100 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 120 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 140 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第二轮测试: 1 初始化 140 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 160 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 180 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第三轮测试: 1 初始化 180 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 200 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 220 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第四轮测试: 1 初始化 220 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 240 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 260 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。		

4. 新建 Bug

用例名称	新建 Bug	用例编号	011
测试步骤	1: 部署性能测试环境。 2: 用 loadrunner 录制脚本使用 PortMapping 策略录制脚本。		
场景设计	场景模式: 使用百分比模式设置每台负载机承受 50%的负载(Real-World 模式)。 第一轮测试: 1 初始化 100 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 120 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 140 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第二轮测试: 1 初始化 140 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟;		

	2 初始化至 160 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 180 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第三轮测试: 1 初始化 180 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 200 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 220 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第四轮测试: 1 初始化 220 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 240 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 260 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。

5. 修复 Bug

用例名称	修复 Bug	用例编号	009
测试步骤	1: 部署性能测试环境 2: 用 loadrunner 录制脚本使用 PortMapping 策略录制脚本		
场景设计	场景模式: 使用百分比模式设置每台负载机承受 50%的负载(Real-World 模式)。 第一轮测试: 1 初始化 100 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 120 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 140 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第二轮测试: 1 初始化 140 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 160 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 180 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 4StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。 第三轮测试: 1 初始化 180 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 2 初始化至 200 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; 3 初始化至 220 个 Vusers(每 5 秒增加 2 个)，Duration 持续压测 10 分钟; StopUsers: 每隔 5 秒停止 2 个用户，直到停止所有用户。		

6.5、测试结果及其分析

{PS: 性能分析相对于脚本开发和场景设计来说反而简单许多，为什么？别急，听愚兄为你们娓娓道来。

上面的篇幅中有介绍到 OS，DB，网络，中间件的监控；我们可以把 responsetime 原子化分割成独立的部分，比如我们执行一次压测的时候监控 OS，DB，网络，中间件，并在代码里作埋点处理(埋点其实很好理解，最简单的就是函数开始时计时，函数执行完后再次计时，从而得出函数执行时间)，这样一来，我们清楚地知道每一层的性能状况，从而分析性能问题上会事半功倍。这里不得不提一点：运维同学对监控的专业程度远非我们性能测试同学能想象的哈哈~别

说我没告诉过你“性能测试的实施不仅仅只要有测试工程师就够了”。

另：再次强调一下不要以为掌握了 LR 的使用就表示会性能测试了，这不扯淡么！

LR 它本身就只是一个工具，只不过我们可以用它很方便地来实施性能测试。

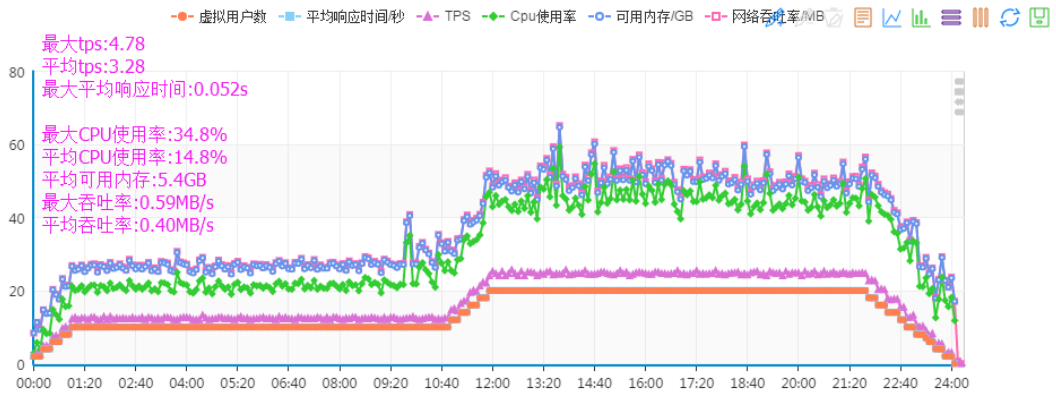
如果我们不用 LR，而选择自己写脚本来达到模拟 N 多用户的并发请求这不更好么？

But：做事情要考虑成本和回报！已经有这些很好用的工具了(LR，Jmeter，Gatling)就不要重复造轮子了，辛辛苦苦造出来，说不定到最后推行时阻力重重会被废掉。

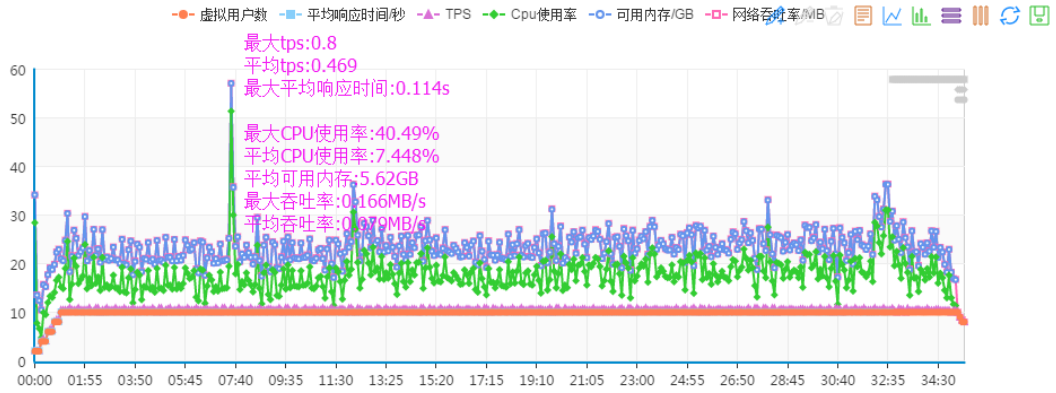
当然，BAT 这种怪兽团队除外，不差钱，不差人，不差时间，天时地利人和均得那就没啥好说的了。咋整？playyourselves!!!

}

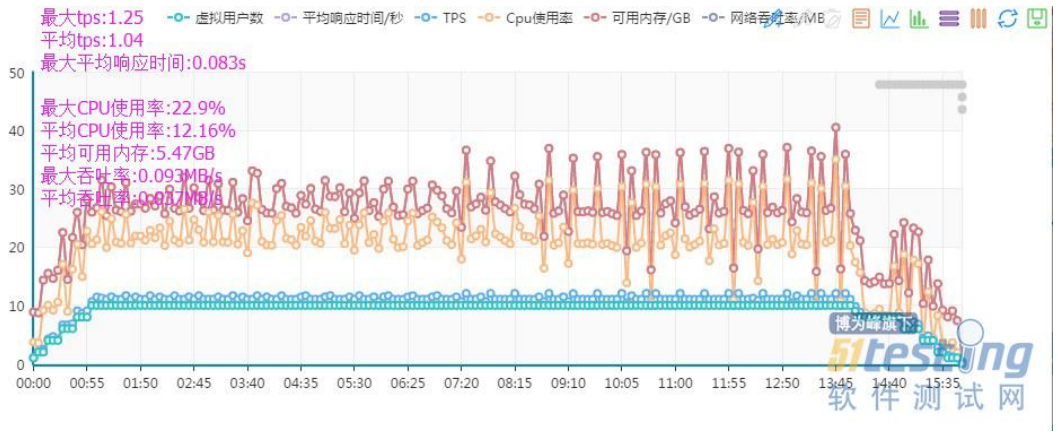
6.5.1、登录场景结果分析



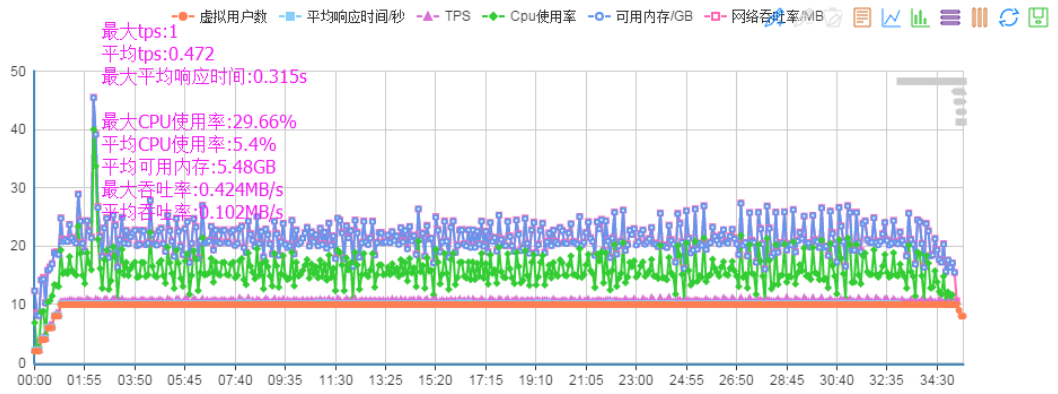
6.5.2、添加用例场景结果分析



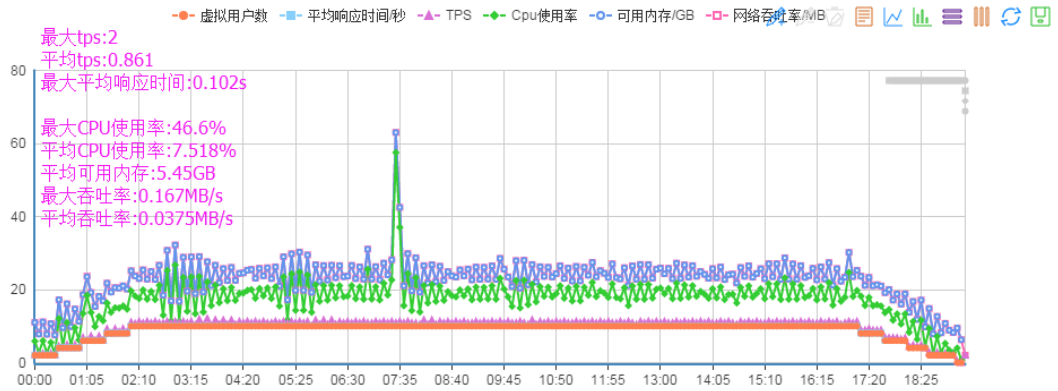
6.5.3、执行用例场景结果分析



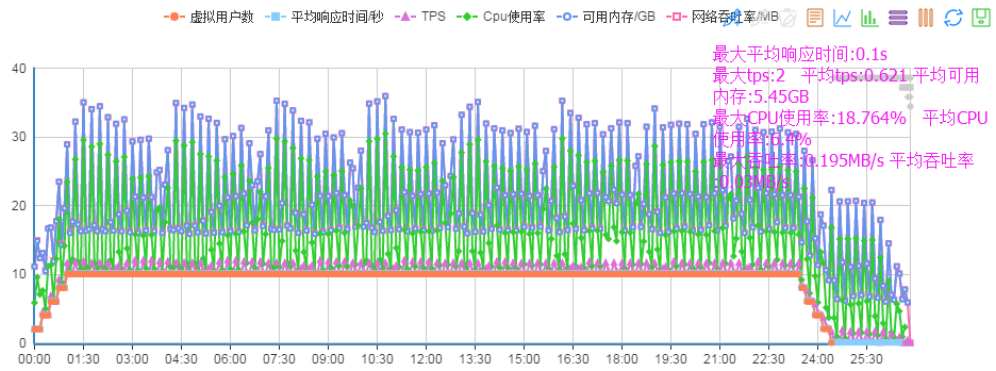
6.5.4、提交 Bug 场景结果分析



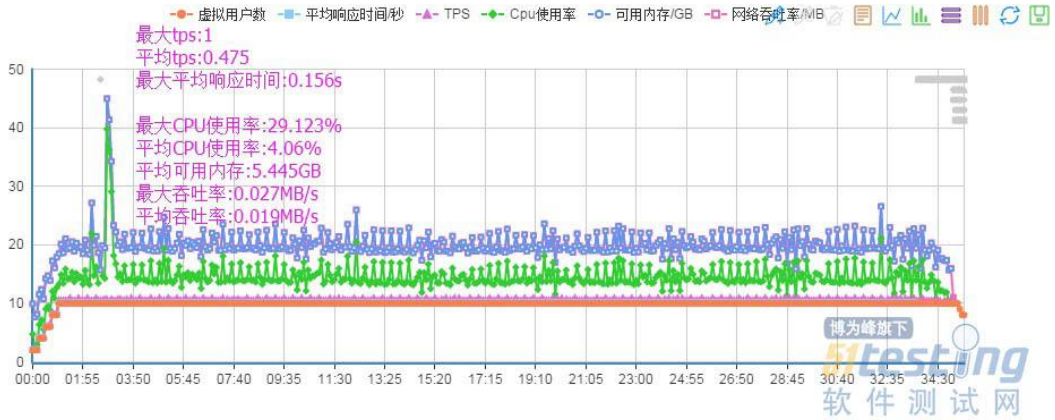
6.5.5、解决 Bug 场景结果分析



6.5.6、关闭 Bug 场景结果分析



6.5.7、确认 Bug 场景测试结果分析



6.6、测试结论

6.6.1、团队符合性评估

场景名称	成功事务数	事务成功率	最大tps	平均tps	最大平均响应时间	最大CPU使用率	平均CPU使用率	平均可用内存	最大吞吐率	平均吞吐率
登录	4792笔	100%	4.78笔/s	3.28笔/s	0.052s	34.80%	14.80%	5.4GB	0.59MB/s	0.40MB/s
添加用例	2000笔	100%	0.80笔/s	0.46笔/s	0.114s	40.49%	7.45%	5.62GB	0.166MB/s	0.079MB/s
执行用例	2000笔	100%	1.25笔/s	1.04笔/s	0.083s	22.90%	12.16%	5.47GB	0.093MB/s	0.037MB/s
新建bug	1000笔	100%	1.00笔/s	0.47笔/s	0.315s	29.66%	5.40%	5.48GB	0.424MB/s	0.102MB/s
确认bug	1000笔	100%	1.00笔/s	0.47笔/s	0.156s	29.12%	4.06%	5.44GB	0.027MB/s	0.019MB/s
修复bug	1000笔	100%	2.00笔/s	0.86笔/s	0.102s	46.60%	7.51%	5.45GB	0.167MB/s	0.0375MB/s
关闭bug	1000笔	100%	2.00笔/s	0.62笔/s	0.100s	18.76%	6.40%	5.45GB	0.195MB/s	0.030MB/s

分析结论:

测试过程中 CPU 使用率小于 50%，平均响应时间远小于 5s，且 tps 均大于目前团队业务量。

禅道(开源版 v8.0.1)满足团队目前及未来 5 年内使用。

6.6.2、负载测试

登录场景负载测试结果:

测试次别	最大TPS	平均TPS	最大响应时间	最大CPU使用率	平均CPU使用率	平均可用内存	最大磁盘读写繁忙时间	最大网络吞吐率
第一轮测试	33.938	25.559	0.052s	72.48%	42.92%	5.545GB	24.03%	0.249MB/s
第二轮测试	43.844	32.869	0.093s	77.34%	54.42%	5.359GB	17.78%	0.347MB/s
第三轮测试	53.188	39.92	0.117s	92.99%	65.59%	5.378GB	17.25%	0.404MB/s
	PS：最大支持并发用户数200Vusers							

添加用例场景负载测试结果:

测试次别	最大TPS	平均TPS	最大响应时间	最大CPU使用率	平均CPU使用率	平均可用内存	最大磁盘读写繁忙时间	最大网络吞吐率
第一轮测试	6.875	5.151	0.173s	72.99%	46.19%	5.371GB	24.82%	1.169MB/s
第二轮测试	8.391	6.447	0.955s	100.00%	62.68%	5.265GB	115.02%	1.439MB/s
第三轮测试	未执行							
	PS：最大支持并发用户数V140users							

执行用例场景负载测试结果:

测试次别	最大TPS	平均TPS	最大响应时间	最大CPU使用率	平均CPU使用率	平均可用内存	最大磁盘读写繁忙时间	最大网络吞吐率
第一轮测试	8.734	6.703	0.254	100.00%	73.11%	5.243GB	53.91%	0.319MB/s
第二轮测试	未执行							
第三轮测试	未执行							
	PS：最大支持并发用户数V120users							

提交 Bug 场景负载测试结果:

测试次别	最大TPS	平均TPS	最大响应时间	最大CPU使用率	平均CPU使用率	平均可用内存	最大磁盘读写繁忙时间	最大网络吞吐率
第一轮测试	3.578	2.703	0.155	63.70%	34.03%	5.4GB	15.58%	0.162MB/s
第二轮测试	4.531	3.493	0.231	91.65%	46.06%	5.345GB	52.42%	0.22MB/s
第三轮测试	未执行							
	PS：最大支持并发用户数V160users							

修复 Bug 场景负载测试结果:

测试次别	最大TPS	平均TPS	最大响应时间	最大CPU使用率	平均CPU使用率	平均可用内存	最大磁盘读写繁忙时间	最大网络吞吐率
第一轮测试	2.813	2.153	0.108s	82.54%	49.68%	5.302GB	52.62%	0.599MB/s
第二轮测试	3.594	2.799	0.201s	94.47%	63.40%	5.304GB	38.15%	0.198MB/s
第三轮测试	未执行							
	PS：最大支持并发用户数V160users							

关闭 Bug 场景负载测试结果:

测试次别	最大TPS	平均TPS	最大响应时间	最大CPU使用率	平均CPU使用率	平均可用内存	最大磁盘读写繁忙时间	最大网络吞吐率
第一轮测试	3.484	2.565	0.401	99.92%	73.98%	5.275GB	94.64%	0.209MB/s
第二轮测试	未执行							
第三轮测试	未执行							
	PS：最大支持并发用户数V60users							

负载测试结果汇总:

场景名	支持的最大并发用户数
登录禅道	200Vusers
添加用例	140Vusers
执行用例	120Vusers
提交bug	160Vusers
解决Bug	160Vusers
关闭Bug	60Vusers

分析结论:

各场景测试过程中响应时间和 CPU 使用率和磁盘读写速率密切相关;CPU 使用率为主要性能瓶颈点。

禅道部署优化建议:

为了禅道系统能达到更优的服务，建议将集成环境配置改为 web 服务和 mysql 数据库分开部署在不同的 PC 上而不是使用现有的集成环境。

七、写给测试路上一起前行的同学们

他山之石可以攻玉

性能测试涉及到的知识域非常广，我们需要了解网络，OS，系统架构，业务逻辑，协议报文，脚本开发，服务和系统的监控等等更方面的知识，这就要求我们在实施性能测试的时候需要和多方进行跨部门沟通协调~

其实，我们完全没必要在所有领域亲自操刀，数据库的监控和调优我们肯定比不了 DBA;

服务和系统的监控我们也绝对的没运维熟悉；而代码级别的性能问题定位我们更是没开发同学清楚；而配置的调优更是运维同学们的拿手好戏~

Sowhat？我们完全可以调动在专项领域里更专业的资源来做正确的事情。

最后，希望有机会看到此文的同学不要眼高手低光说不练，那样只会成为理论派的纯把式~

诸君共勉！

原创测试文章系列（四十三） 精彩推荐

- 测试中的"望闻问切"法
- LoadRunner 脚本编写介绍之 HTTP 篇
- 我的自动化历程-我对自动化测试的一些理解和认识
- 测试女巫之控制鼠标键盘篇
- 场景法--基于用户行为的测试方法
- 无线测试之 H5 测试方法（二）：性能测试
- 3 个月的测试生涯
- 如何给产品需求做"体检"
- 你是如何成为一名捉虫师的-测试校招漫谈
- 你心中好的测试用例是怎样的？
- 服务端测试笔记
- 移动设备的配置测试
- 用 Pageobject 的方式使用 Webdriver

马上阅读