# LoadRunner压力测试

## 概述

LoadRunner是一款工业级性能测试软件，通过模拟真实的用户行为，通过负载、并发和性能实时监控以及完成后的测试报告，分析系统可能存在的瓶颈，支持的语言脚本为C语言。

## 安装

### 安装步骤

### 常见问题

#### 安装时卡主不动

##### 问题描述

在安装LoadRunner时，页面提示“正在执行产品配置，此过程需要几分钟的时间…”

##### 解决方案

因为后台有2个或多个“msiexec.exe”进程，打开任务管理器，若用Administrator安装，则找到用户为“SYSTEM”的“msiexec.exe”进程，若使用User用户，则找到用户为“USER”的“msiexec.exe”进程。kill掉占用大小较小的那个进程。

## LR模块

### 模块组成

Virtual User Generator：录制脚本

Controller：创建、运行和监控场景

Analysis：分析测试结果

### Virtual User Generator

可以监视并记录客户端与服务器之间的通话，让虚拟用户模拟实际的业务流程，记录真正用户操作行为，并转化为特定的测试脚本语言集合。

### Controller

创建测试场景，实现真实的负载，让一台或多台机器模拟多个用户，同时执行脚本，对被测应用进行操作或发起请求，每个LoadRunner的场景都相当于一套负载测试方案，因为它包含了反映业务操作流程的测试脚本，反映系统承受能力的虚拟用户数量，用于平衡测试机自身压力的Load Generator机器及脚本之间执行顺序等，执行场景的过程就是对系统施压的过程。

实时监控场景，在场景执行过程中，用户可以根据需要选择一个或多个监视窗口对关心的数据进行动态监控，可监控的信息包括系统资源、网络设备、web服务器和数据库等交易数据。

### Analysis

能够打开结果数据，帮助用户查找性能问题并追溯原由。



## 脚本录制流程

### 脚本录制协议

#### 创建协议

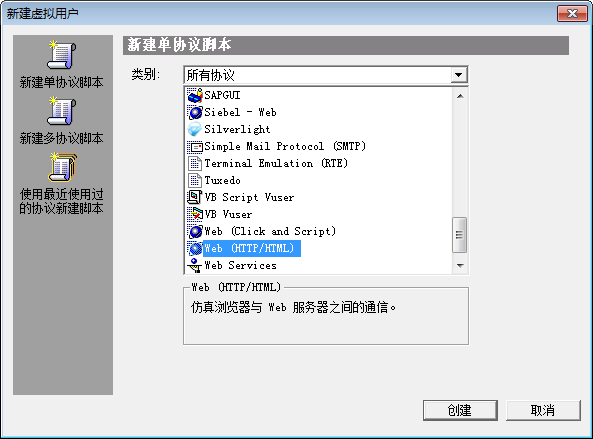
Single Protocol：使用单协议的Vuser脚本

Multiple Protocol：使用多个协议的Vuser脚本

Recent：最近创建脚本所用到的协议

Mobile：移动应用脚本所使用到的协议（新版本）

Popular：常用协议（新版本）



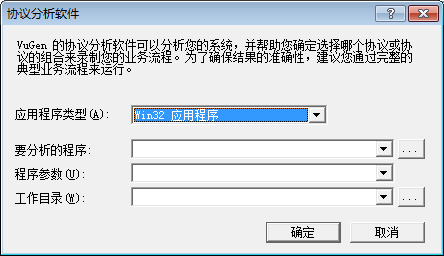
#### 选择协议

* 根据应用是B/S还是C/S选择协议：

B/S：一般选择Web（HTTP/HTML）协议

C/S：根据后端数据库类型选择

* 使用Protocol Advisor进行检测



* 其他第三方协议检测工具

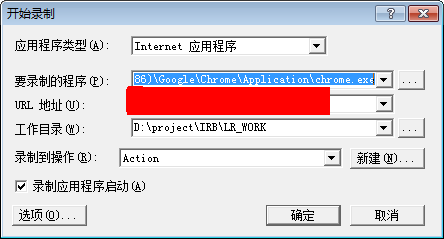
### 脚本录制概述

#### 录制文件说明

* vuser\_init：存放应用程序初始化脚本（只执行一次）
* Action：存放实际的操作脚本（可多次执行）
* vuser\_end：存放应用程序注销和关闭时的脚本（只执行一次）

#### 开启脚本录制

选择应用程序类型、URL地址（要录制程序地址），开始录制



#### 脚本录制控制



自左向右依次为：

开始录制、运行、停止、暂停、编译、选择录制位置、新建Action、插入开始事务、插入结束事务、插入集合点、插入注释、编辑录制选项、插入文本检查

#### 脚本常用函数

web\_url()：页面访问函数，无需页面支持，使用频率高。

web\_submit\_data()：页面数据提交，无需页面支持，使用频率高。

lr\_think\_time()：思考时间。

lr\_save\_string()：记录字符串，如lr\_save\_string(lr\_decrypt("5e54d369802848923"), "Password")，将字符串加密保存到Password中。

#### 脚本操作

脚本回放：用以检查脚本录制的内容和运行是否符合要求。

### 脚本增强

#### 脚本增强的作用

录制的测试脚本达不到预期的测试目的

缺乏针对性

无法体现自动化测试

#### 事务管理

##### 事务概述

性能测试中，除了衡量整个脚本的性能外，还需要脚本中某段或某几段操作的性能，以更详细的字段具体用户哪些动作对系统性能的影响较大。

事务是局部操作的汇总，是TPS的基础。定义事务时，首先在事务的开始和结束位置，分别插入一个事务起始标记，在脚本运行结束后，系统会在结果信息中单独反映每个事务的运行结果。

阅读运行日志，可以分析和查看事务的运行状态（Passed/Failed），事务可以嵌套，更加详细的统计事务。

##### 事务函数

开始事务：lr\_start\_transaction("事务名称");

结束事务：lr\_end\_transaction("事务名称",LR\_PASS);

#### 集合点

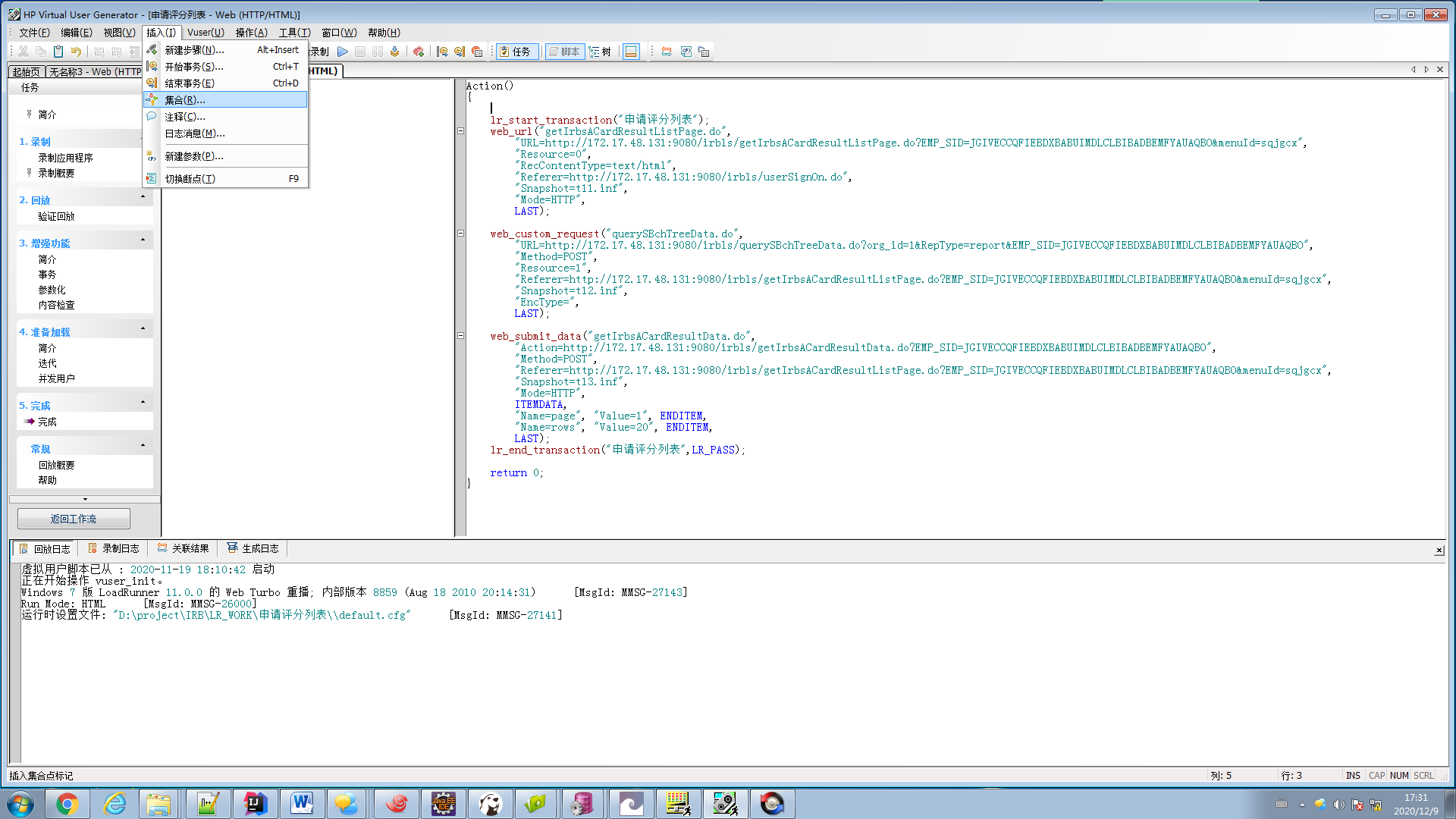
##### 集合点概述

并发过程仅仅体现在开始执行的一刹那，随着服务器对请求的响应时间的不一致或系统环境条件限制，用户执行速度不一致，在运行中能够集合到一点的可能性很小，这样并不是真正意义的并发，而系统的压力情况是这样用户集合到系统瓶颈的某个点进行操作，通过集合点，实现真正意义的并发。

##### 集合点函数

集合点只能放在Action中，且应该放在事务之前。

lr\_rendezvous("集合点");



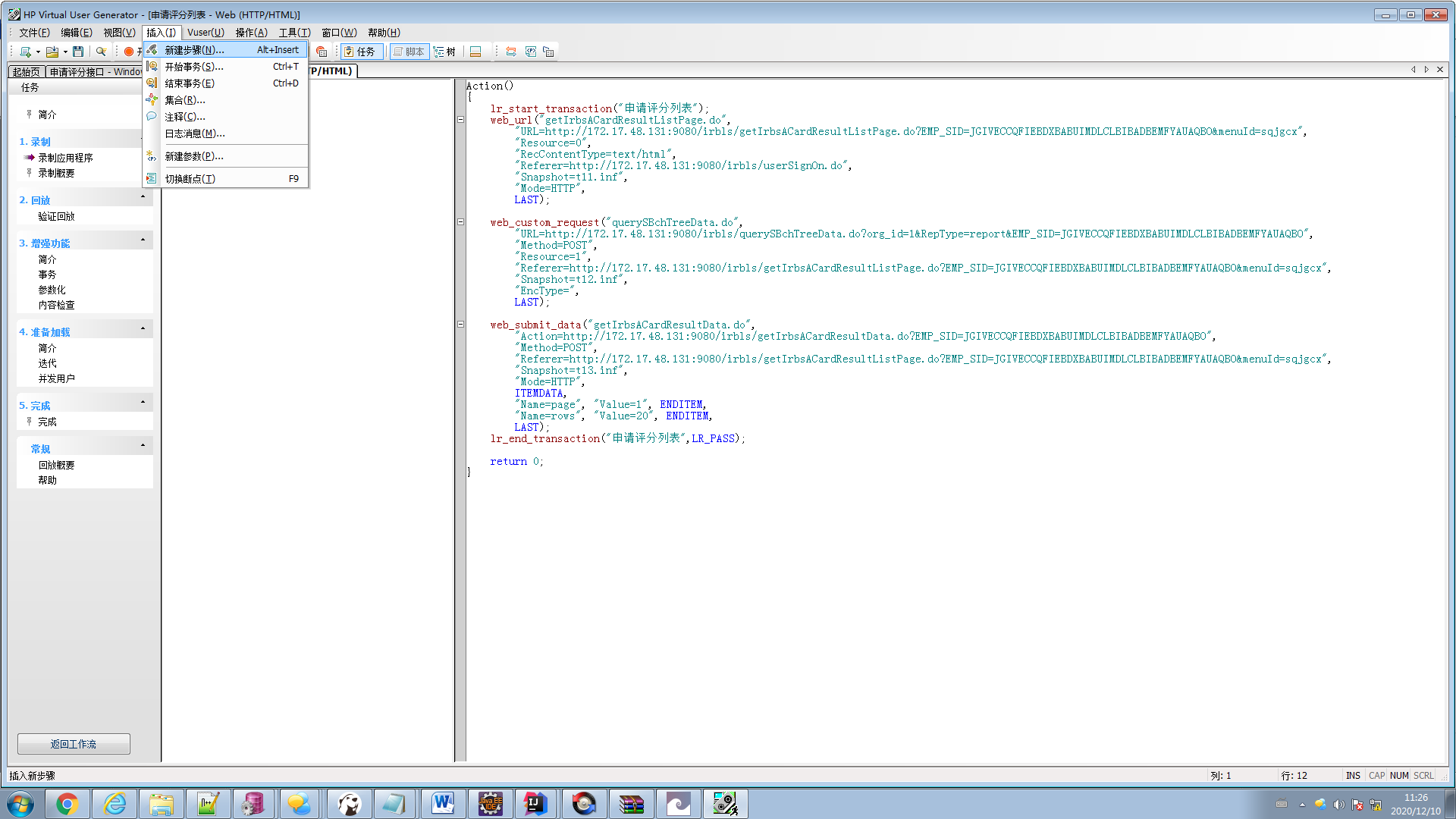
#### 检查点

##### 检查点概述

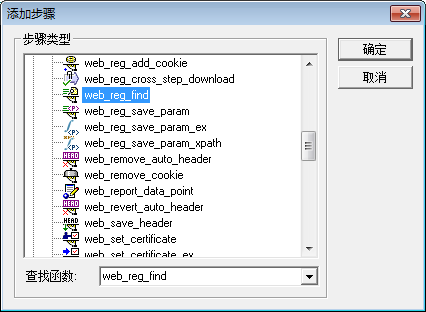
主要验证某个界面是否存在指定的Text对象，可以通过检查压力较大时web服务器是否返回正常页面。需要在页面请求前设置。可以通过运行日志查看检查点是否正确。

##### 插入检查点

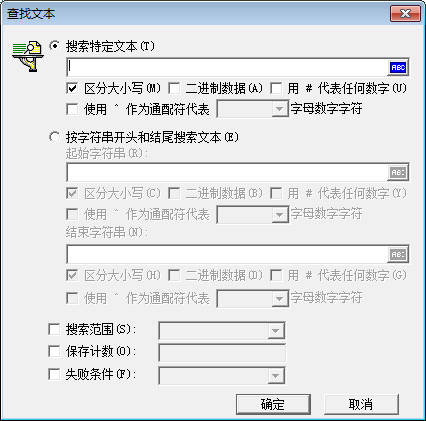
* 插入->新建步骤



* 搜索web\_reg\_find函数



* 编辑检查点内容



##### 检查点函数

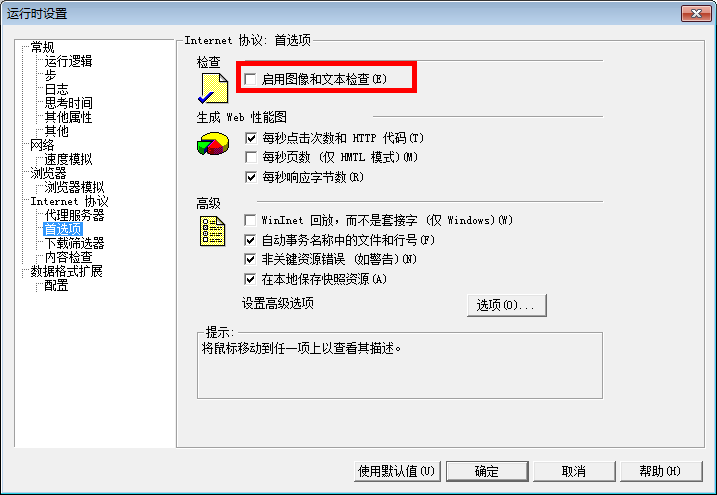
web\_reg\_find();

web\_reg\_find(" Search=Body", "Text=welcome", LAST);

web\_reg\_find("Fail=NotFound", " Search=Body", "Text=customer", LAST);

##### 开启检查点

打开运行时设置，点击首选项，启动图像和文本检查。



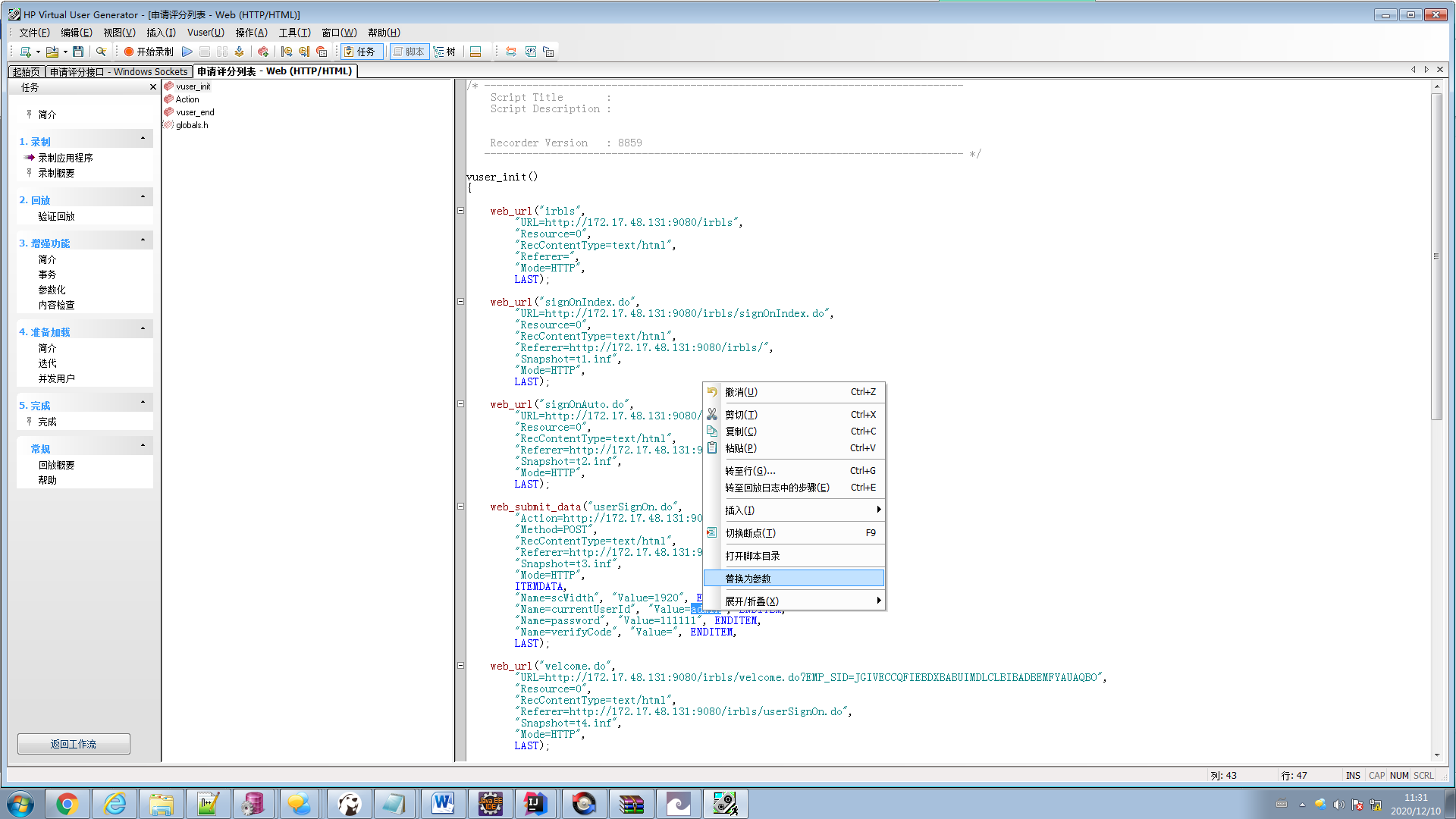
#### 参数化运行

##### 参数化概述

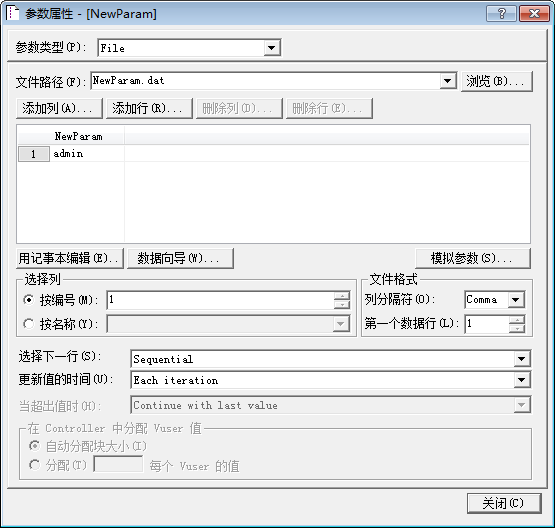
脚本参数化，是针对脚本中的某些常量，定义一个或多个包含数据源的参数来取代，让场景中不同的虚拟用户在执行相同脚本时，分别使用参数数据源中的不同数据替代这些常量，从而达到模拟多用户真实使用系统的目的。

##### 创建参数

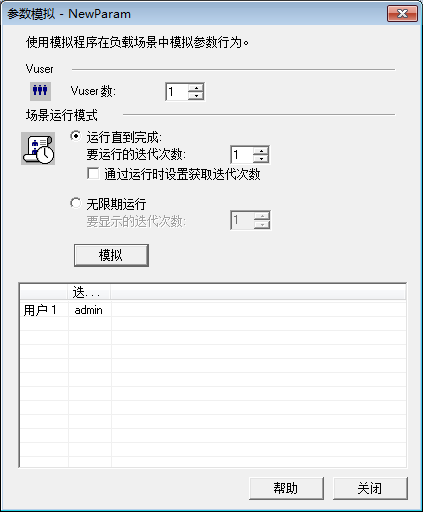
* 选择要替换的内容，右键选择替换为参数



* 编辑参数属性



##### 模拟参数



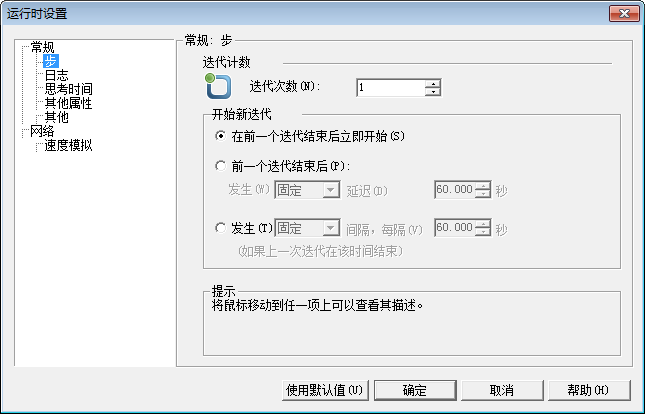
#### 脚本运行时设置

##### 运行次数设置

用于设置脚本的循环运行次数，其中init和end仅运行一次，其他脚本可以运行多次。

迭代设定：

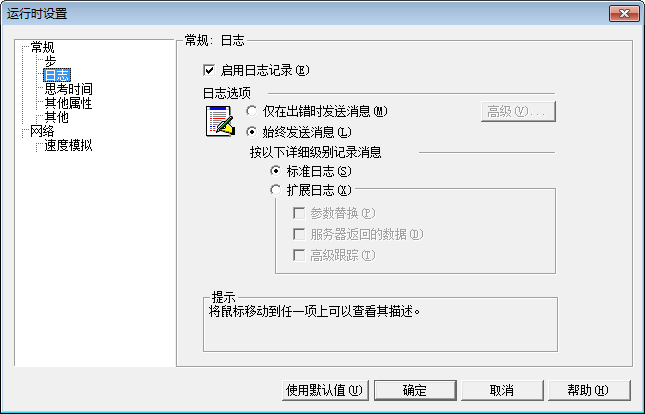
* 在前一个迭代结束后立即开始。
* 前一个迭代结束后：可以是指定时间，也可能是两个时间间隔的随机值。
* 新的迭代在指定或随机时间后，直接开始。



##### 日志

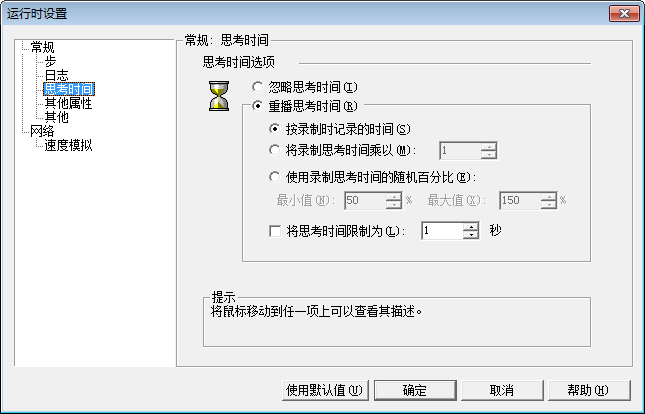
提供给测试人员查看脚本的运行状态，日志功能只有开启才能使用，可以通过扩展日志，查看参数的使用情况、服务器返回次数等信息。

日志路径在脚本目录下output.txt



##### 思考时间

思考时间是指两次请求之间的时间间隔，一般不会选择忽略思考时间，思考时间函数为lr\_think\_time(5);



##### 错误处理

脚本运行错误处理方式：

* 出现错误仍继续运行
* 当调用lr\_error\_message时，将打开的事务置为失败
* 对错误进行快照



##### 多线程设置

一个进程包含多个线程，在一个进程中的多个线程，共享进程分配到的资源，线程是软件能够支持并发的一个重要运行机制，开发过程中，大多数系统会使用支持多线程的开发语言。

* 按进程运行Vuser：如果按进程运行每个Vuser，对于每个Vuser实例都将启动一个mdrv进程，都将反复启动同一驱动程序并将加载到内存中，从而会占用大量的RAM及其他系统资源。
* 按线程运行Vuser：这些线程Vuser将共享父驱动进程的内存段，这就消除了多次重新加载驱动程序/进程的需要，节省大量内存空间，从而可以在一个负载生成器上运行更多Vuser。

## Controller

### 场景设计

谁？：配置Vuser数量

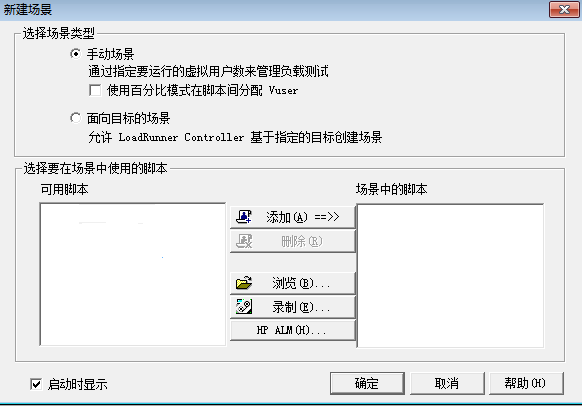
时间？：配置场景开始时间

地点？：配置Load Generator

做什么？：选择脚本

怎么做？：设计脚本加压、持续时间

### 场景分类



#### 手工场景

概述：通过手工设置虚拟用户量、加压、减压、持续时间和其他内容。

特点：手工、灵活

作用：整体了解系统处理能力和性能问题，从而确定系统瓶颈。

使用场合：需要对系统多个方面的性能指标进行验证。

#### 目标导向场景

概述：根据设定的目标进行运行，只要设计好目标，用户量、时间、并发等都不要单独设置，因为其自发形成负载测试，因此无法设置集合点。

特点：自动、简易

作用：验证系统是否到了预期目标，从而确定系统瓶颈。

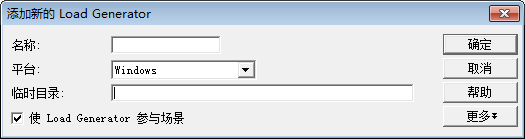
使用场合：有明确的测试目标。

### 场景测试流程

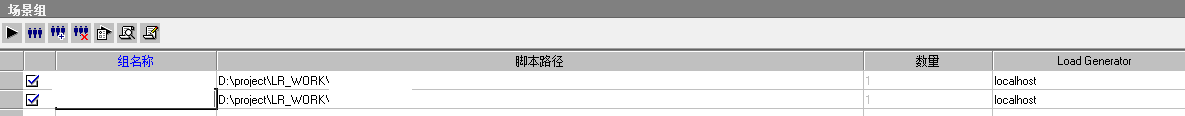
#### 设置Load Generator（负载机）





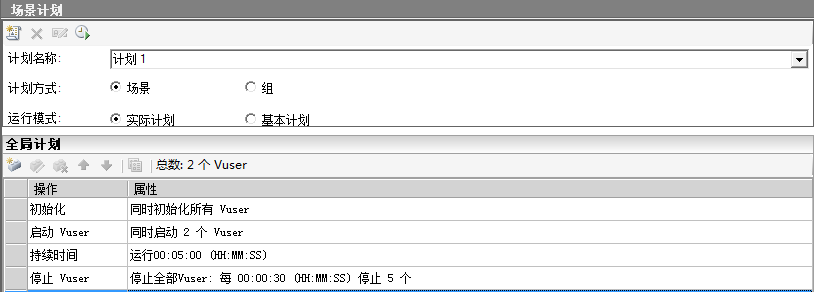


#### 设置脚本及虚拟用户



#### 设置计划方式（加压、减压策略）

##### 场景

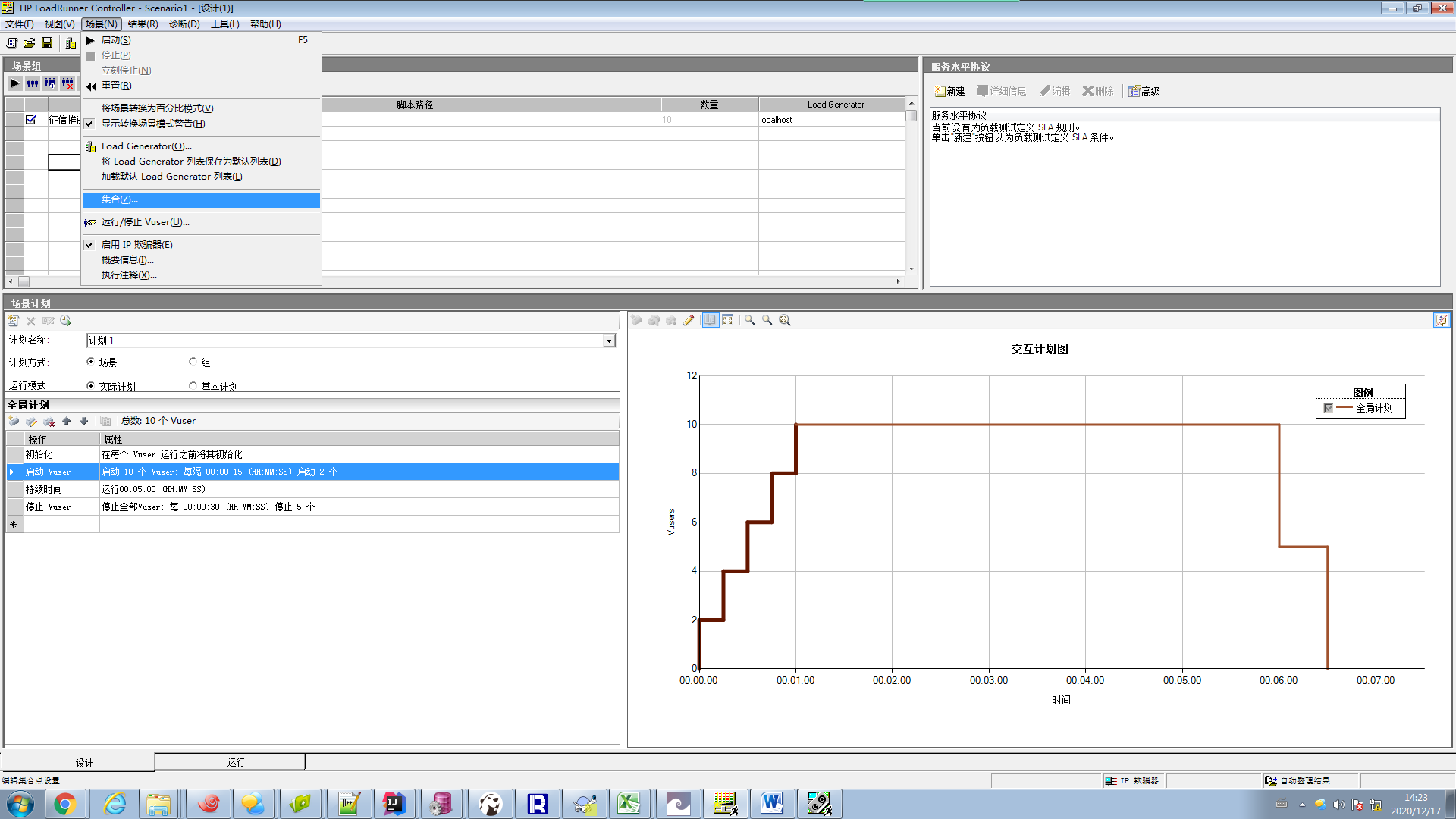


##### 组



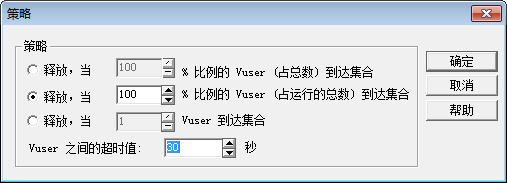
### 集合点配置

点击场景->集合



点击集合信息的策略按钮





百分比虚拟用户都到集合点位置

百分比运行的用户到集合点位置

固定数目的虚拟用户到集合点位置

虚拟用户的超时设置

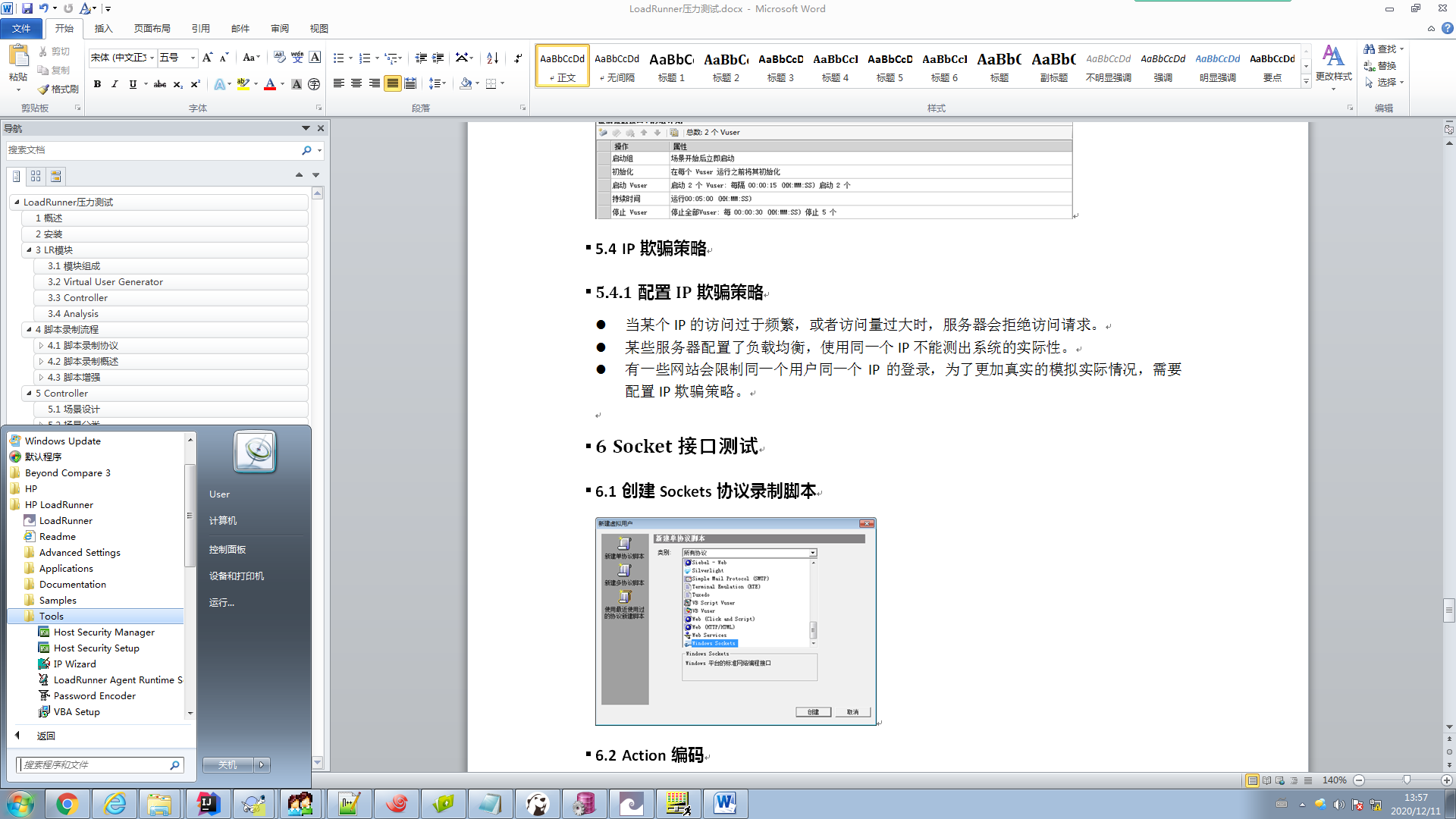
### IP欺骗策略

#### IP欺骗策略概述

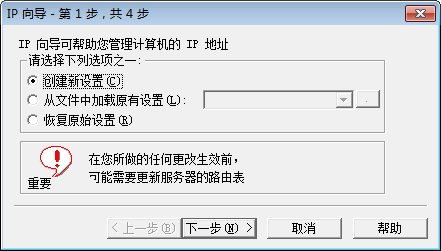
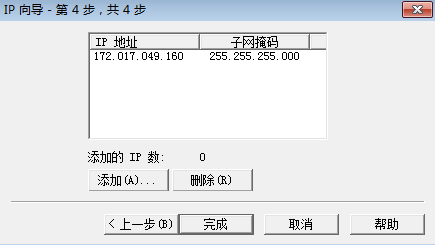
* 当某个IP的访问过于频繁，或者访问量过大时，服务器会拒绝访问请求。
* 某些服务器配置了负载均衡，使用同一个IP不能测出系统的实际性。
* 有一些网站会限制同一个用户同一个IP的登录，为了更加真实的模拟实际情况，需要配置IP欺骗策略。

#### 配置IP欺骗策略

##### 打开IP Wizard

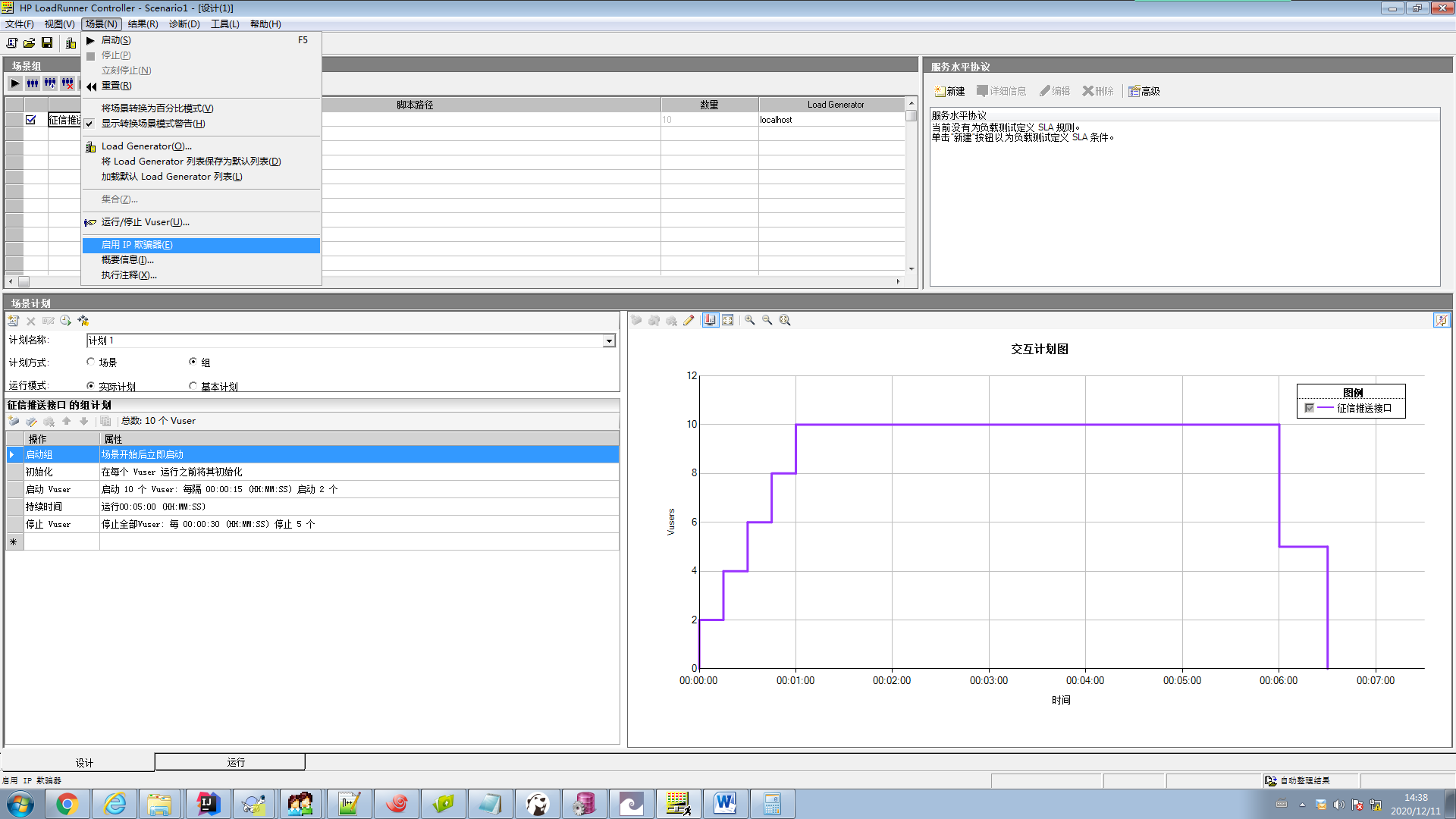


##### 创建IP欺骗策略

#### 开启IP欺骗策略

##### 启用IP欺骗器



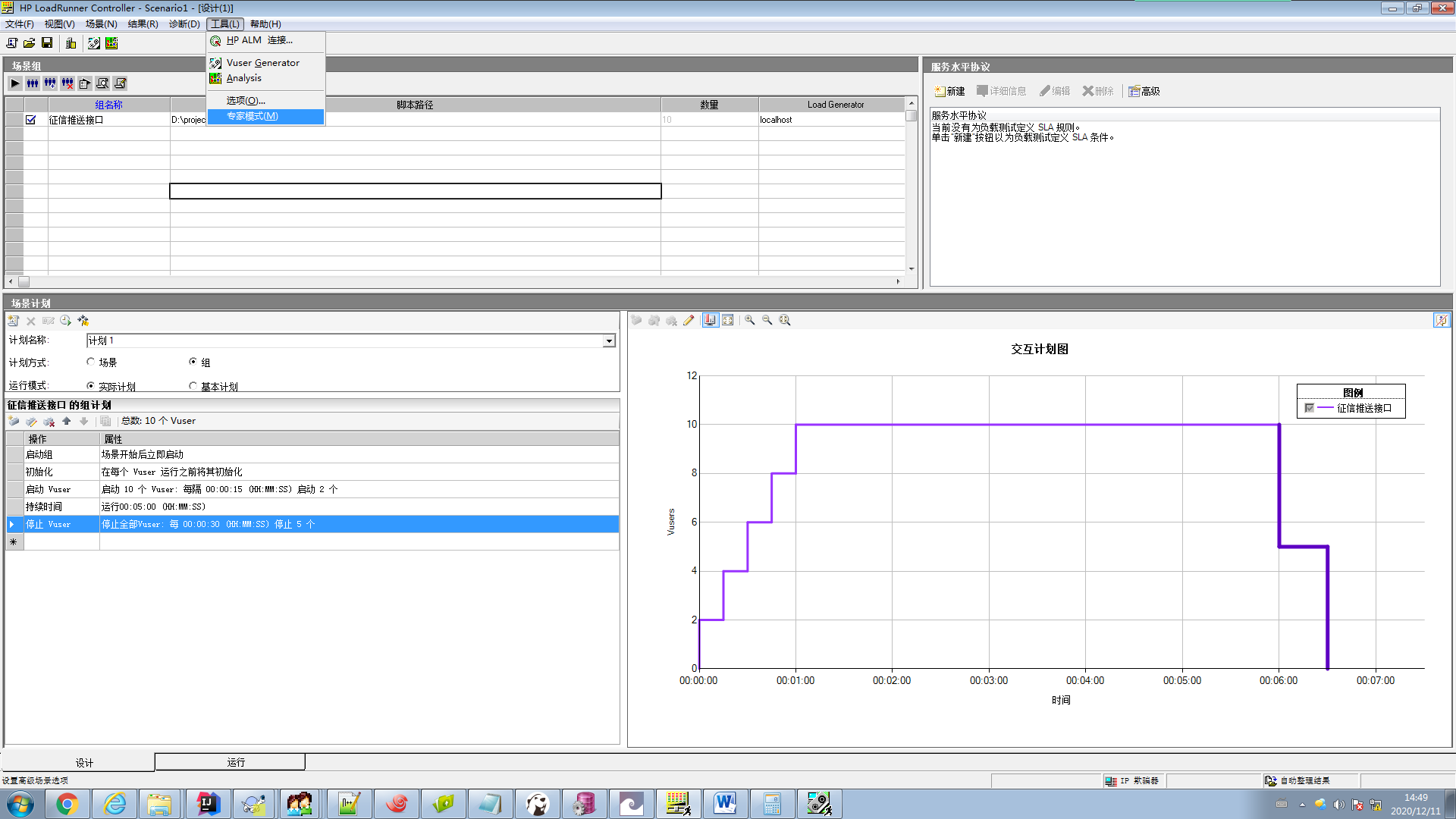
##### 修改虚拟用户为线程

在Controller打开运行时设置，选择按线程运行Vuser

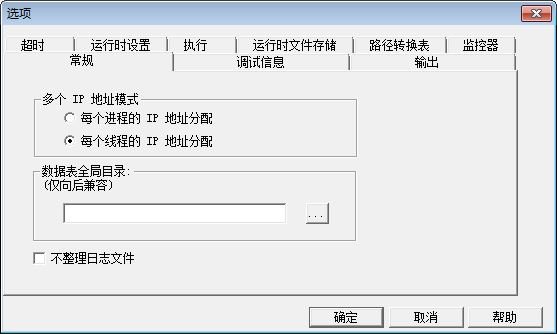


##### 设置线程与IP地址的关系

在Controller选择专家模式

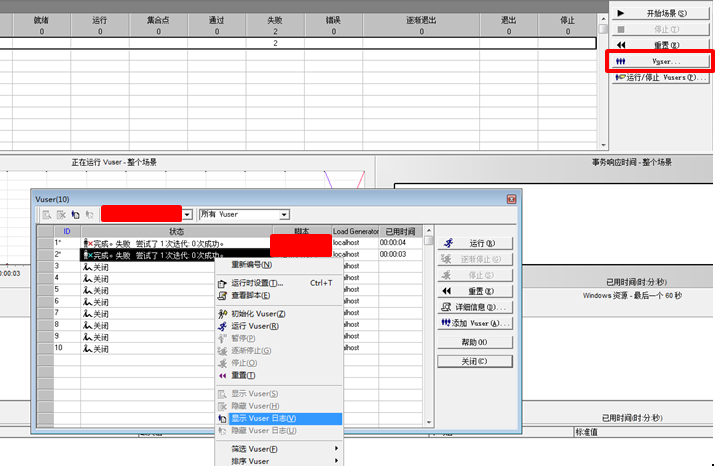


选择选项按钮（只有在专家模式才可以），选择常规选项卡，点击每个线程的IP地址分配



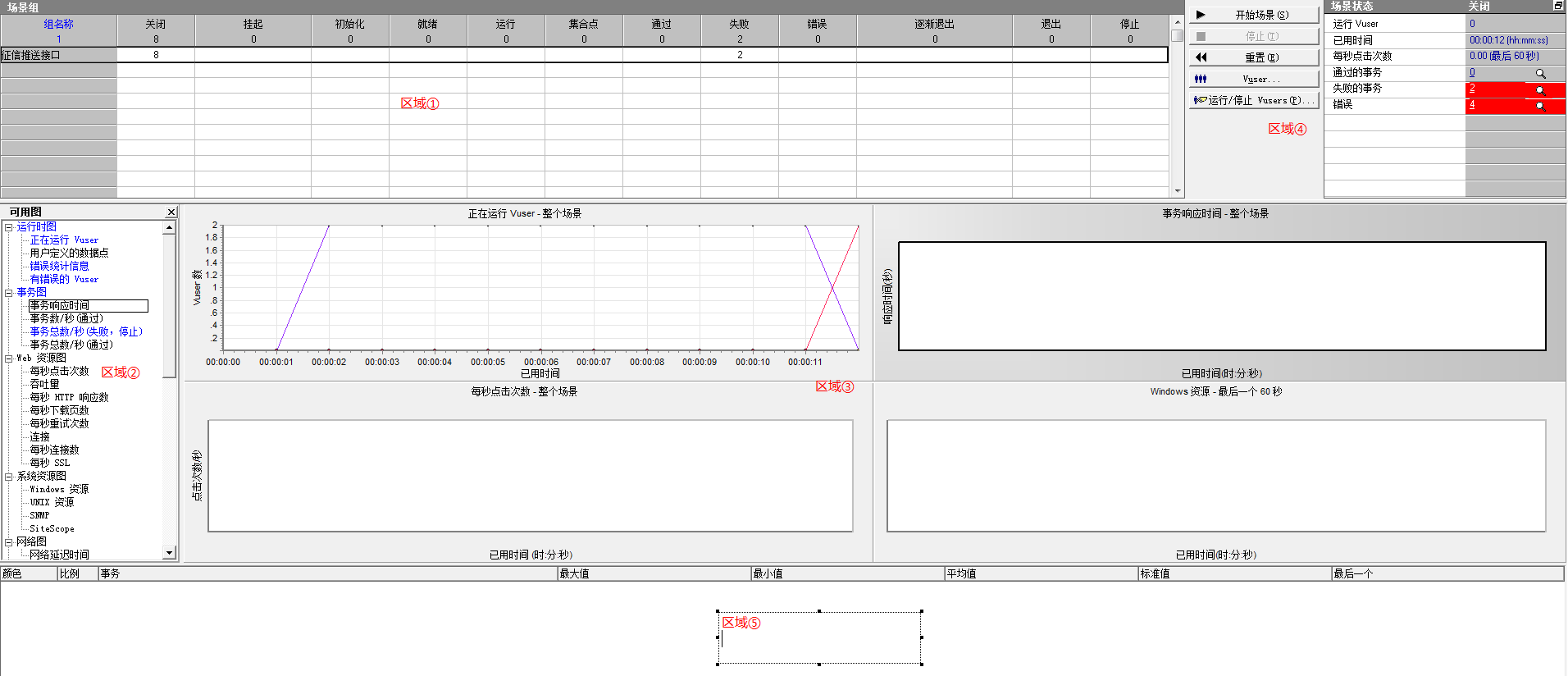
### 查看运行日志

选择Vuser按钮，右键某一条，点击显示Vuser日志，可查看运行日志



### 运行时监控

#### 运行时监控布局



区域①：显示场景中每一个脚本组的虚拟用户运行状态。

区域②：显示当前可以查看的监控数据，蓝色为可查看，其它为不可查看。

区域③：展示区域②中可供查看的资源的实时动态变化，最多可以设置16个资源图的变化。

区域④：控制场景运行，显示当前场景的运行状态。

区域⑤：资源图的详细信息及内容。

#### 场景信息

##### 虚拟用户运行状态

关闭、挂起、初始化、就绪、运行、集合点、通过、失败、错误、逐渐退出、退出、停止。

##### 场景控制

开始场景、停止、重置、Vuser（查看当前虚拟用户状态）、运行/停止Vuser。

##### 场景运行状态

运行Vuser：当前正在运行的虚拟用户数量

已用时间

每秒点击次数

通过的事务

失败的事务

错误

#### 图信息

运行时图：主要表现运行过程中，虚拟用户的运行情况。

事务图：表现运行过程中，场景中事务的运行状态，事务响应时间：每秒钟事务数。

网页资源图：主要表现web测试中几个重要资源的数据图，如：每秒钟点击数、吞吐量、每秒钟HTTP响应、连接数、每秒SSL数量。

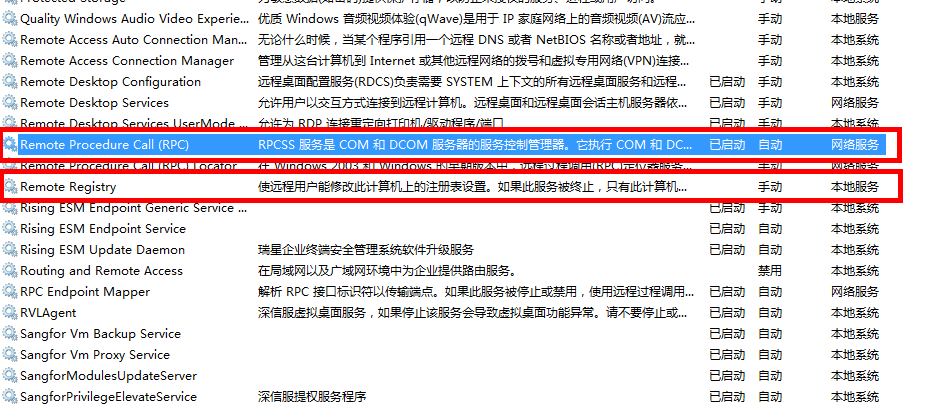
系统资源图：操作系统资源消耗图，需要单独设置并需要连接服务。

网络图：主要监控网络延迟。

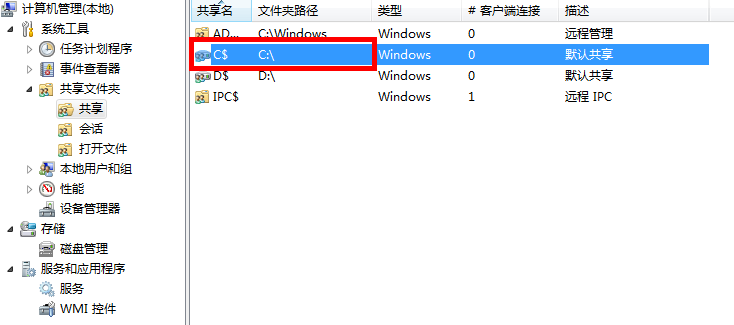
Web服务器图：监控web服务器的资源情况

#### 监控Windows资源

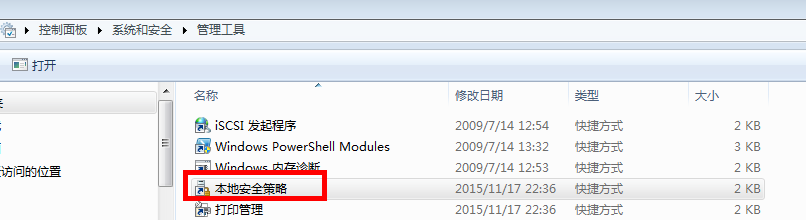
1）启动Windows RPC服务

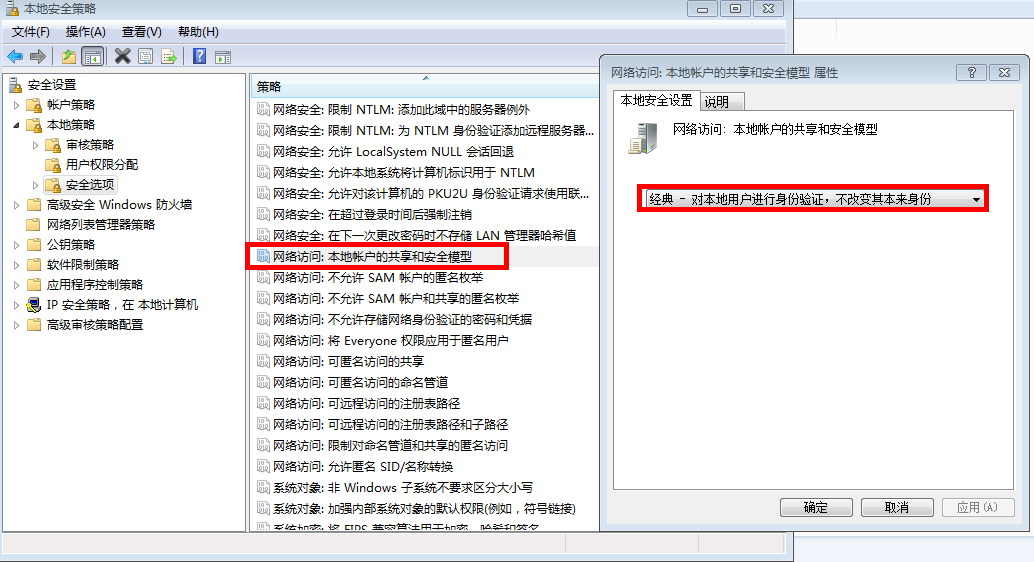


2）创建虚拟文件C盘

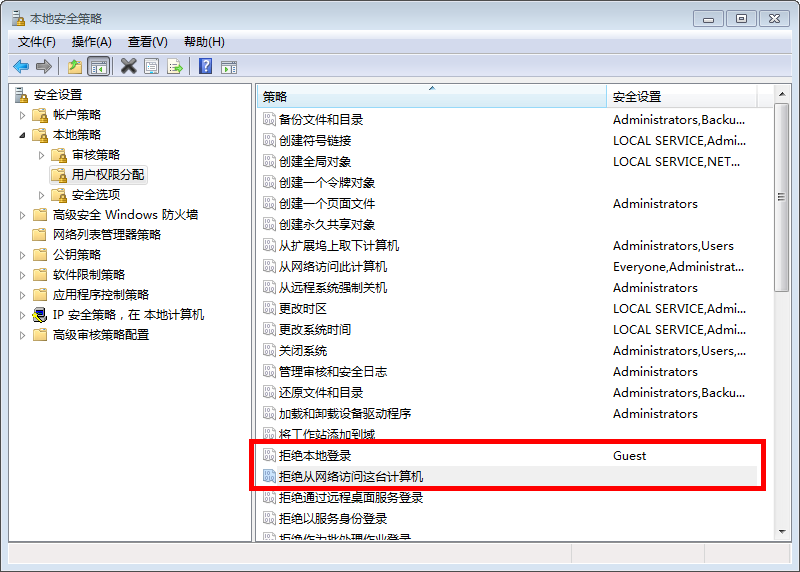


3）配置网络访问：本地账户的共享和安全模型为经典，访问[\\IP\c$](file:///\\IP\c$)为可用

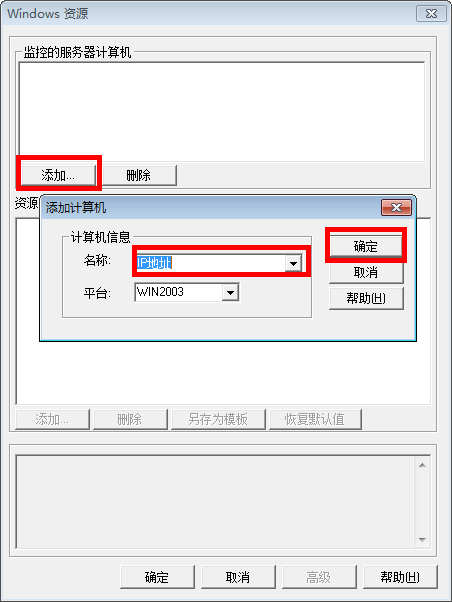




4）设置计算机不被拒绝网络访问



5）添加度量：右键Windows资源图->添加度量->添加被监控的服务器计算机，输入IP地址



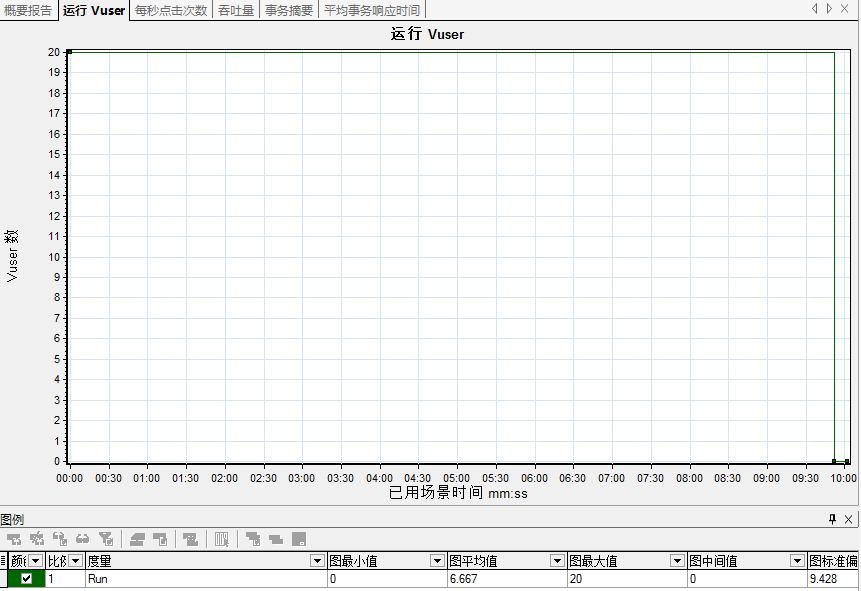
## Analysis

### 分析结果-概要报告

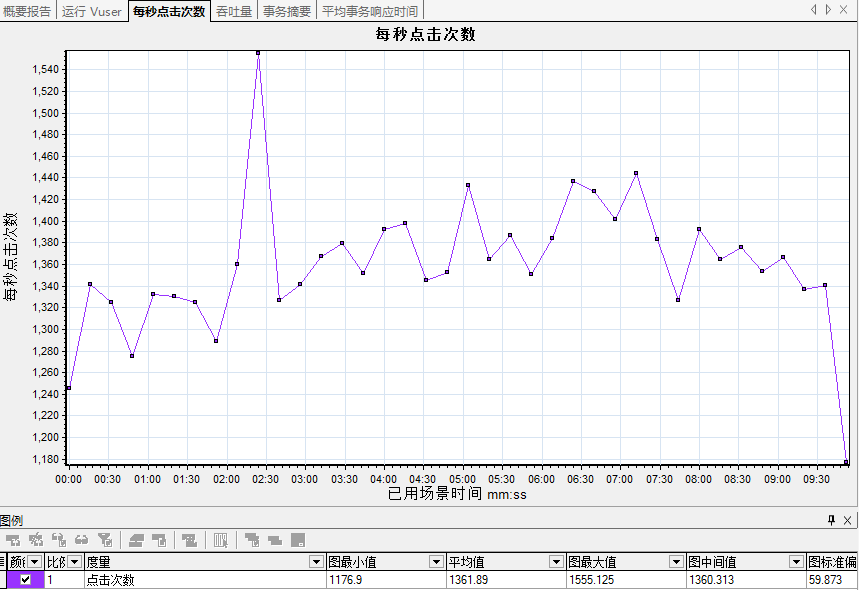


* 分析概要
  + 场景名称
  + 运行结果的存储位置
  + 持续时间：场景运行时间
* 统计信息概要
  + 运行Vuser的最大数目：运行的虚拟用户量
  + 总吞吐量
  + 平均吞吐量：每秒吞吐量
  + 总点击次数
  + 平均每秒点击次数
* 事务概要
  + init、end自带的事务
  + 用户自定义的事务
  + 脚本文件产生的事务，一个脚本代表一个事务
* HTTP响应概要
  + 响应状态码
  + 响应总次数和平均每秒次数

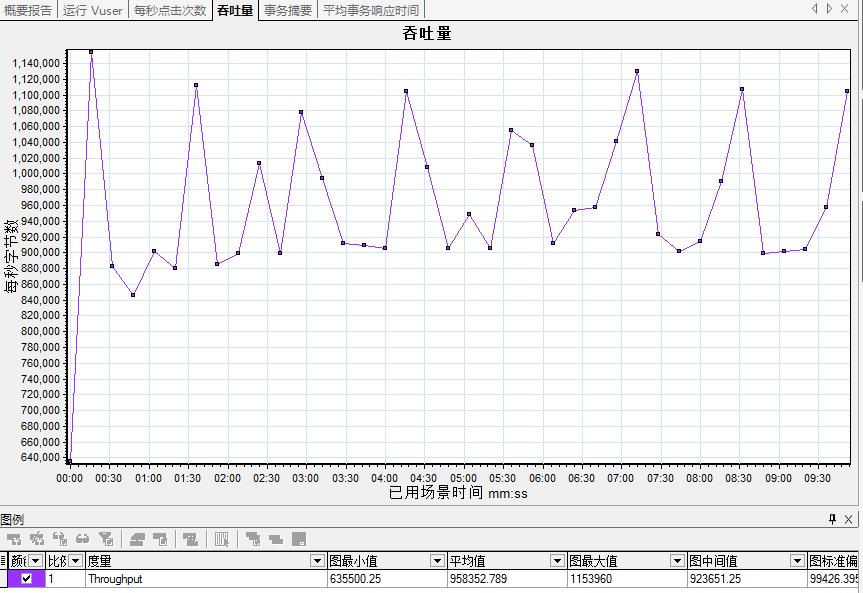
### 分析结果-运行Vuser



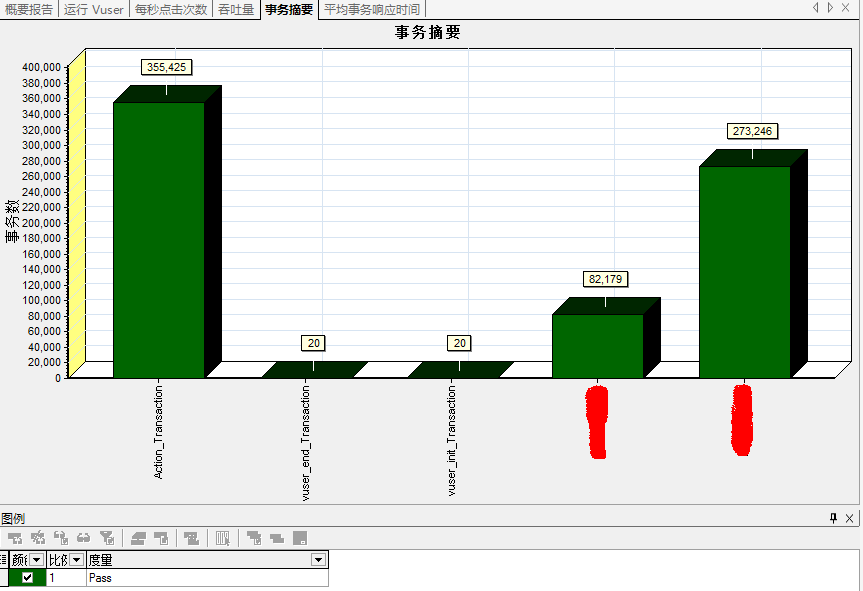
### 分析结果-每秒点击次数



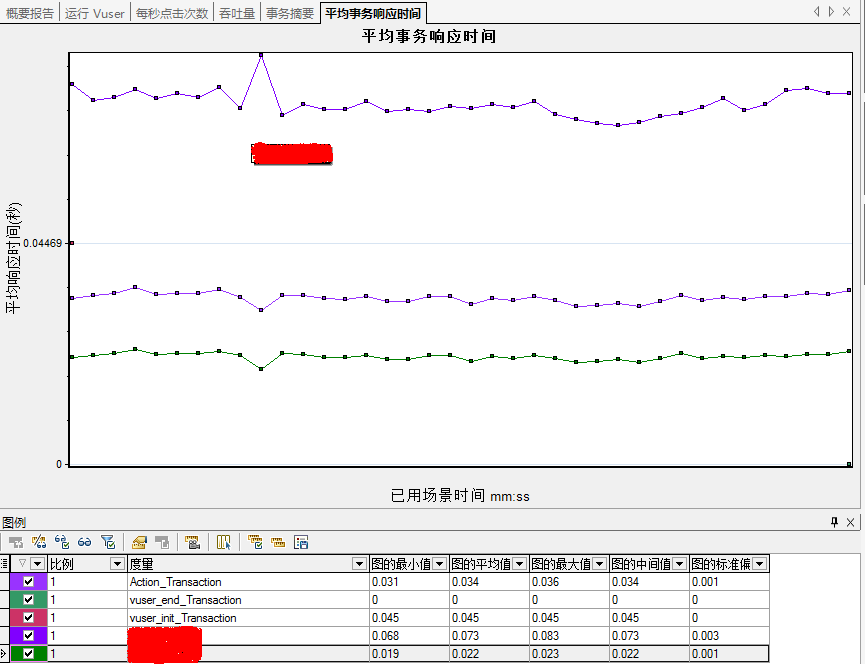
### 分析结果-吞吐量



### 分析结果-事务摘要

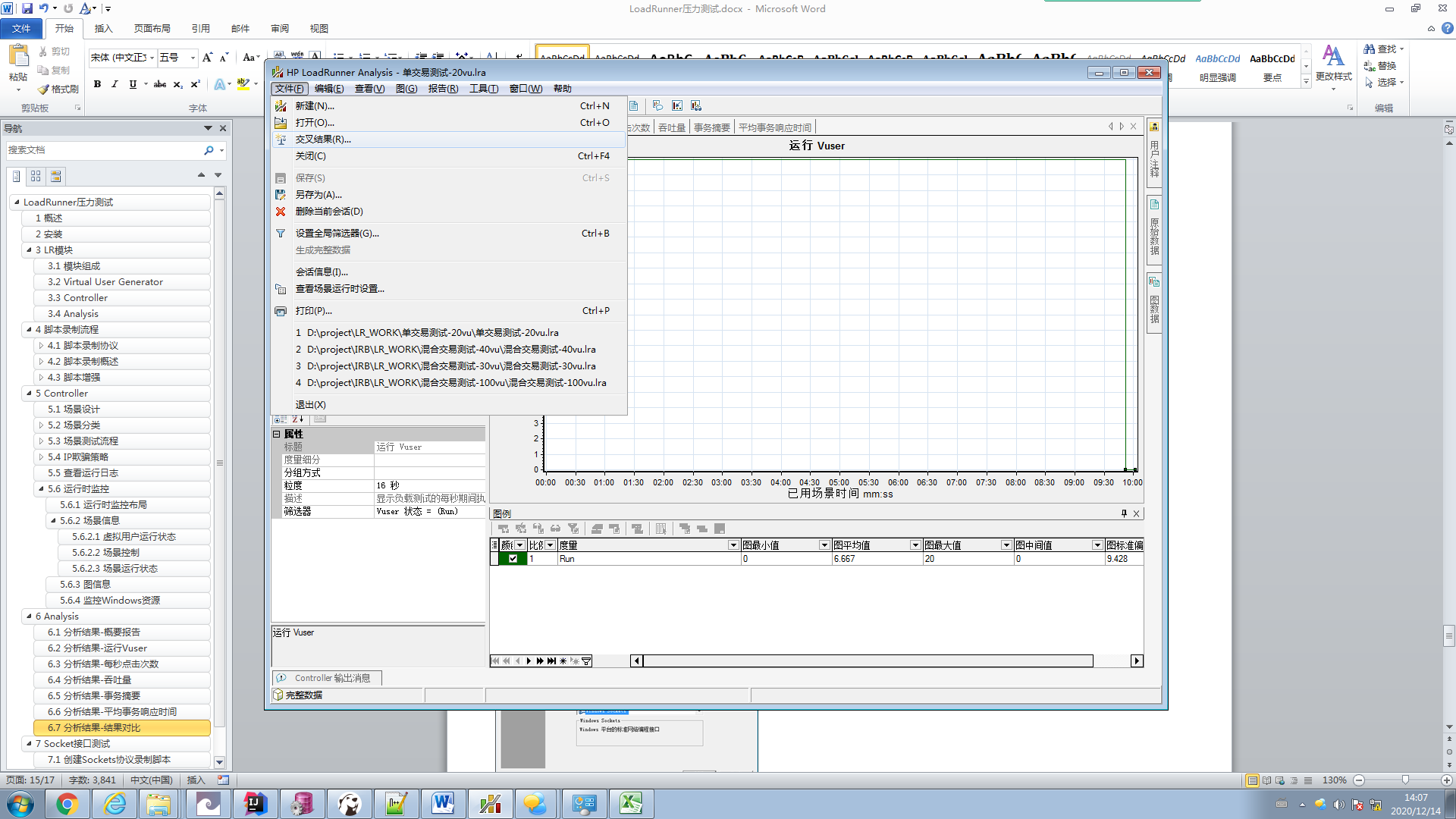


### 分析结果-平均事务响应时间



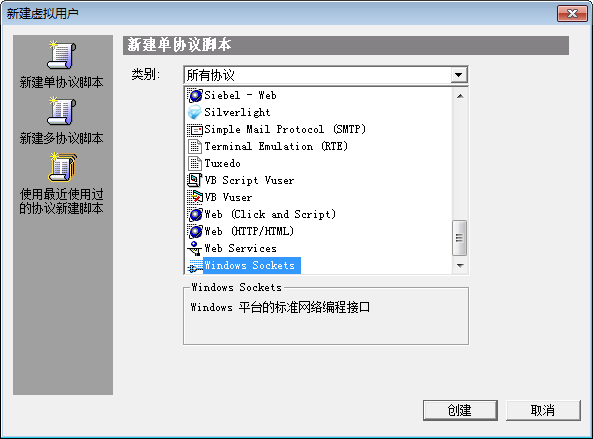
### 分析结果-结果对比

文件->点击交叉结果，对两次（多次）测试结果的对比与判断，使得测试人员可以将同一个场景（服务器优化前后）的结果进行对比



## Socket接口测试

### 创建Sockets协议录制脚本



### Action编码

|  |
| --- |
| #include "lrs.h"  Action()  {  char \*recvbuf;  int recvlen=0;  int num;  //lrs\_create\_socket("socket0","TCP","RemoteHost=172.17.49.160:8899",LrsLastArg);  //lrs\_create\_socket("socket0","TCP","RemoteHost=172.17.49.196:8899",LrsLastArg);  // 开启事务  lr\_start\_transaction("申请评分接口");  // 创建Socket连接  lrs\_create\_socket("socket0", "TCP", "RemoteHost=172.17.48.131:8899", LrsLastArg);  lr\_output\_message("=========发送报文开始===========");  // 发送报文  lrs\_send("socket0", "buf0", LrsLastArg);  // 发送报文结束  lrs\_disable\_socket("socket0", DISABLE\_SEND);  lr\_output\_message("=========发送报文结束===========");  // 判断结束标签  lrs\_set\_receive\_option(EndMarker,BinaryStringTerminator,"</root>");  // 设置接收报文超时时间  lrs\_set\_recv\_timeout(120,0);  // 获取返回报文  lrs\_receive("socket0","buf1",LrsLastArg);  //num=lrs\_get\_last\_received\_buffer\_size("socket0");  //lrs\_save\_param\_ex("socket0","received",NULL,0,num,LR\_ENC\_SYSTEM\_LOCALE,"param1");  // 获取收到的最后一个缓冲区及其大小  lrs\_get\_last\_received\_buffer("socket0",&recvbuf,&recvlen);  lr\_output\_message("=========返回报文开始===========");  lr\_output\_message("返回内容长度：%d",recvlen);  // 输出返回报文  lr\_output\_message("：%s",recvbuf);  lr\_output\_message("=========返回报文结束===========");  // 结束事务  lr\_end\_transaction("申请评分接口",LR\_PASS);  // 关闭Socket连接  lrs\_close\_socket("socket0");  return 0;  } |

### 处理发送报文

|  |
| --- |
| ;WSRData 2 1  send buf0 3309  "<?xml version=\"1.0\" encoding=\"GBK\"?>  <root>  <head>  <branch\_id>8011001021002</branch\_id>  </head>  <body>  <customername>王小二</customername>  <queryreason>01</queryreason>  <querytime>2020/09/03 08:24:32:631</querytime>  <ret\_message></ret\_message>  <remark1>备注1</remark1>  <remark2>备注1</remark2>  <remark3>备注1</remark3>  </body>  </root>"  recv buf1 1024  -1 |

## 常见问题

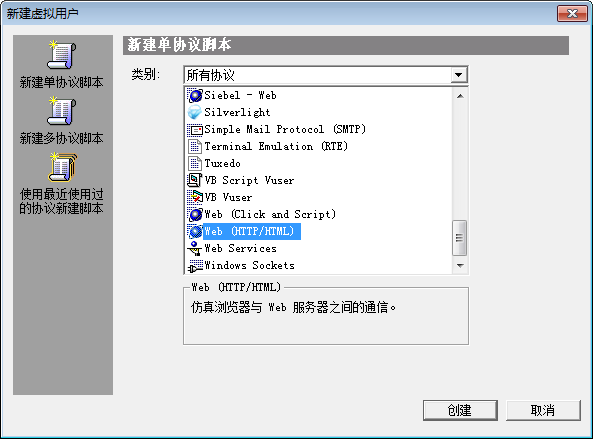
### 脚本录制不自动打开浏览器且录制无结果

#### 问题原因

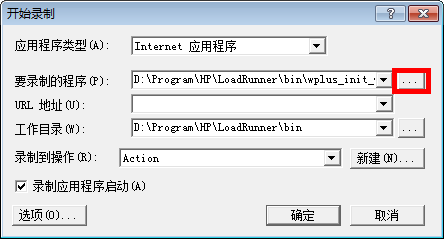
LoadRunner在遇到高版本的IE/谷歌/火狐等浏览器时，无法实现自动录制，此时可以使用LoadRunner自动的wplus\_init\_wsock插件实现自动化录制

#### 解决方案

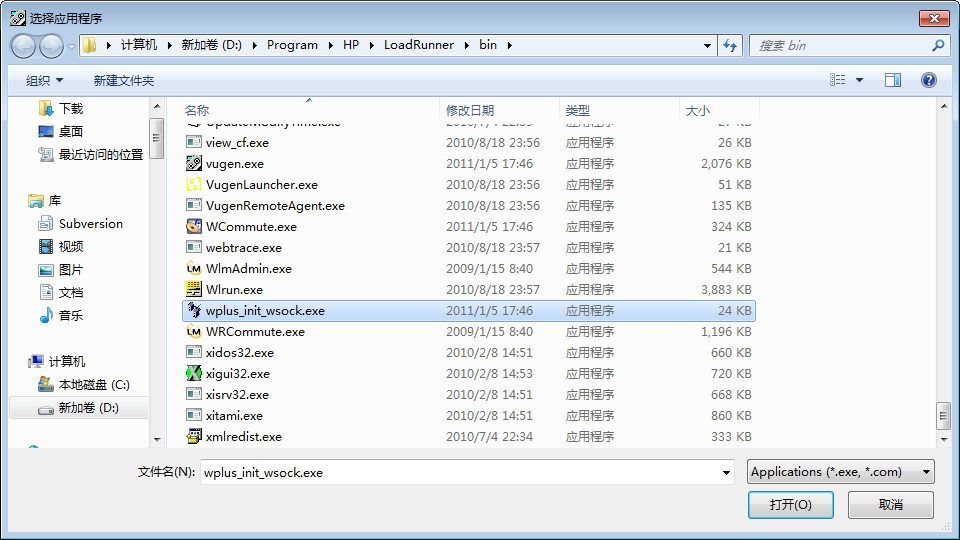
##### 创建脚本



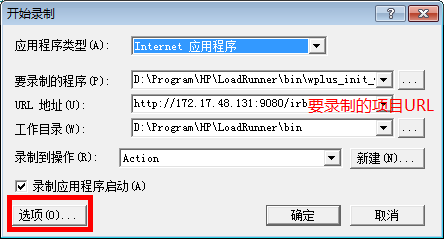
##### 选择wplus\_init\_wsock



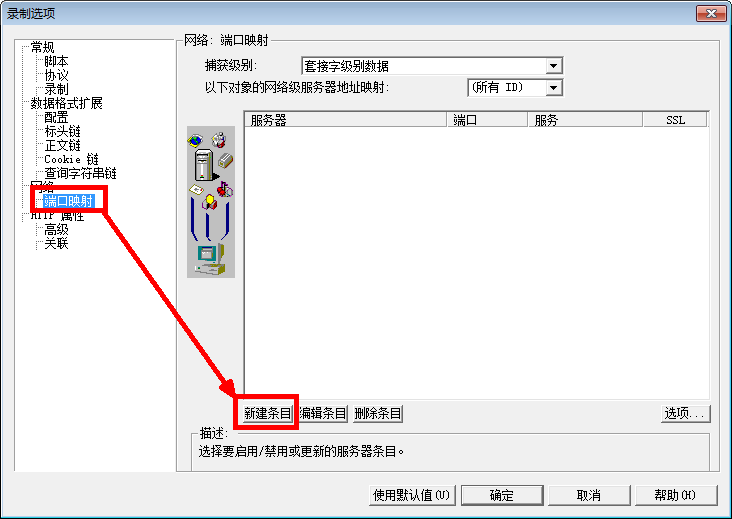
在LoadRunner安装目录下，找到bin目录，找到wplus\_init\_wsock.exe并选择

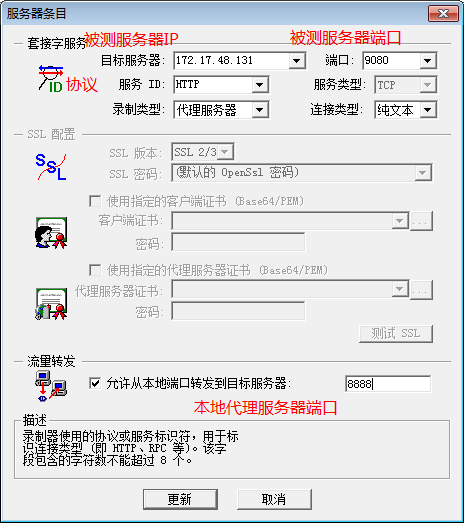


##### 创建端口映射



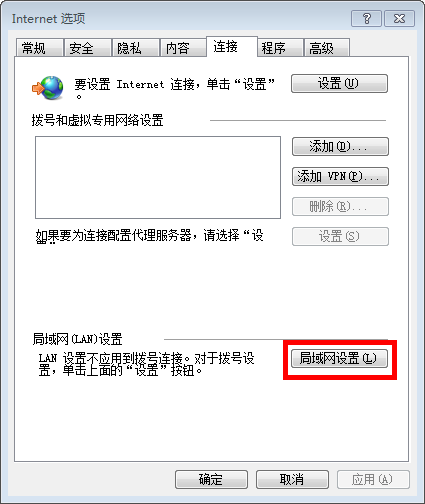
选择选项按钮，新建端口映射

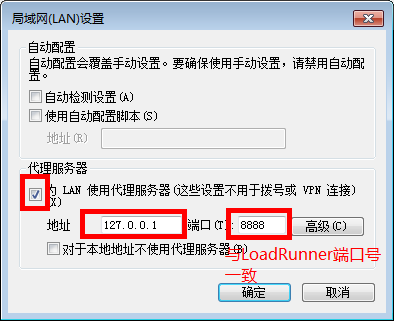




##### 设置目标服务器代理

打开IE浏览器，设置->Internet选项->连接->局域网设置





##### 启动录制开启代理服务器

