泽众性能测试软件
PerformanceRunner

技术白皮书
Version2.0

上海泽众软件科技有限公司

2018 年 1 月

目录

1. 总述.....4

1.1 背景.....4
1.2 解决方案.....4
1.3 概述.....5

2.系统概述.....6

2.1 系统定位.....6
2.2 性能测试软件的概念.....6
2.3 业务提供.....8
2.3.1 PerformanceRunner 适用性说明.....8
2.3.2 性能测试.....9
2.3.3 负载测试.....9
2.3.4 压力测试.....9
2.3.5 配置测试.....9
2.3.6 并发测试.....9
2.3.7 可靠性测试.....10
2.3.8 特性概述.....10
2.4 产品设计目标.....11

3.系统体系结构特性要求.....11

3.1 系统要求.....11
3.2 系统性能.....12
3.3 扩展能力.....12
3.4 可靠性和可用性.....13
3.5 国际支持.....13

4.系统架构.....13

5.系统基本功能.....14

5.1 测试项目创建.....14
5.2 测试脚本创建与录制.....15
5.3 录制结果回放.....17
5.4 测试脚本编辑.....17
5.5 测试案例参数化.....18
5.6 关联动态内容.....20
5.7 增加同步点和验证点.....20
5.8 建立测试场景.....21
5.9 测试场景执行.....23
5.9 测试分析.....24
5.10 统计图表.....24
5.11 IP 欺骗.....29
5.12 系统性能监控.....33

6.PerformanceRunner 的特点..... 33
7.厂商支持能力.....35

1. 总述

1.1 背景

随着软件规模的发展和对软件系统的依赖，人们发现：软件的质量对应用系统的影响日益增加，质量存在问题的软件会导致帐务出错，客户信息丢失，用户的服务出错。

性能对应用系统的重要程度越来越重要，特别是大量的 B/S 系统出现之后。

性能测试、负载测试、压力测试、配置测试和并发测试越来越称为系统上线之前的基本测试。

性能测试需要大量的并发来实现，通过大量的人员集中在一起进行性能测试，是难以达到目标，并且需要大量的人力投入。因此，使用性能测试工具是行业发展的共识。

1.2 解决方案

企业可以建立一整套软件自动测试体系，包括：需求管理、测试分析、测试管理、缺陷跟踪，并且把这个过程纳入整个软件项目开发和软件产品开发过程。

实际上，在 CMM 的规范中，测试本身就是 SQA 的一部分。

自动测试的基础就在于测试工具，只有采用了优秀的自动测试软件，才能够解决自动测试的问题。自动测试工具能够在两个阶段给软件开发企业带来价值：第一，对于软件开发人员来说，高效率的自动测试工具能够提供给程序员自己完成开发过程中的冒烟测试，便于在频繁修改的软件过程中迅速完成测试，保证编码的稳定性；第二，对于项目和产品的测试阶段来说，能够提供稳定的回归测试，保证产品的可靠性。

众所周知，在测试阶段发现问题的投入，相对与在软件投产之后出现错误再去解决问题要小的多。

上海泽众软件科技有限公司开发出了国内第一个拥有自主知识产权的自动测试软件——自动测试引擎(PerformanceRunner)，能够帮助用户实现自动化测试。

1.3 概述

1、本技术白皮书适用于上海泽众软件科技有限公司性能测试软件（PerformanceRunner）。

2、本技术白皮书是上海泽众软件科技有限公司自动测试工具（PerformanceRunner）的技术说明，也是技术谈判的主要内容，是采购方询价、系统选型以及系统测试和验收的主要技术依据。

3、本技术白皮书是根据信息产业部颁布的有关技术体制和技术政策并结合上海泽众软件科技有限公司的实际情况制定的。本技术白皮书没有提出而信息产业部的技术体制以及技术标准已有具体规定的内容，应按信息产业部的技术体制以及技术标准执行，如果存在不一致应以信息产业部颁布的最新技术体制及技术标准内容为准。

4、本技术白皮书在内容或技术指标上如果存在错误（包括印刷错误），经双方确认后可对该错误内容或技术指标进行修正。

5、自动测试工具（PerformanceRunner）版本升级之后，上海泽众软件科技有限公司有权对本技术白皮书进行修改，并不需要主动通知用户。

6、本技术白皮书以下内容为用户重点考察内容：

软件的功能、性能、技术指标和环境要求；

设备容量计算和配置方法；

所提供的数据库的功能和性能指标；

软件安装要求；

提供软件的接口、协议等工程技术要求；

乙方供货范围、交货能力和时间、运输、安装、调测验收和培训等项内容的日程安排；

其他有关技术资料。

7、本软件对涉及专利、知识产权等法律条款承担有限责任。

8、本技术白皮书提供了对上海泽众软件科技有限公司的自动测试工具（PerformanceRunner）的相关技术描述，由于用户使用造成损失，上海泽众软件科技有限公司不承担责任。

9、本技术白皮书以中文编写，未经上海泽众软件科技有限公司同意或授权

的其它语言或形式的技术白皮书无效。
本技术规范书的解释权归上海泽众软件科技有限公司。

2.系统概述

2.1 系统定位

PerformanceRunner 是泽众软件的性能测试软件，主要面向大负载量的各种性能测试，提供并发压力产生、测试监控、测试报表等功能。

PerformanceRunner 主要通过协议来进行性能测试，支持 http、https、socket 等协议。在后续的版本中，还将要对某些特定的协议和应用进行测试，比如数据库测试、web service 测试等。

PerformanceRunner 适用于 B/S、C/S 等架构的系统，模拟客户端的用户操作（action）来实现性能测试。

2.2 性能测试软件的概念

- **测试脚本**

性能测试，主要是通过大并发的执行测试脚本来实现性能模拟。

性能测试是一种自动测试，人工往往难以实现。自动测试脚本主要是用来处理通讯协议、交易处理、并发等。测试脚本通常在测试工具的 IDE 里执行，并且获得 IDE 的支持。

- **自动录制**

在我们操作 B/S 系统、C/S 系统的时候，客户端会向服务器发送报文，这些报文数据是隐藏在系统内部的，就需要性能测试软件能够捕获到这些数据包，并且对数据包进行分析，得到测试脚本。

PerformanceRunner 的测试脚本是采用 java 语法的 beanshell，对于熟悉 java 和 java script 的用户而言是非常简单的。

- **VU（虚拟用户）**

虚拟用户，就是用来模拟真实用户的客户端，每个虚拟用户代表了一个真实的客户来操作。

- **事务**

事务又称为 Transaction，在 PerformanceRunner 中的定义如下：事务是这样一个点，我们为了衡量某个操作的性能，需要在操作的开始和结束位置插入这样一个范围，这样就定义了一个 transaction。

事务的作用：PerformanceRunner 运行到该事务的开始点时，就会开始计时，直到运行到该事务的结束点，计时结束。这个事务的运行时间在 PerformanceRunner 的运行结果中会有反映。通俗的讲 PerformanceRunner 中的事务就是针对某个操作的计时器。一旦发现计时开始标识，就开始计时，一旦发现计时结束，就结束计时，并且记录结果，作为一个事务时间。通常事务时间所反映的是一个操作过程的响应时间。

在 PerformanceRunner 中，使用事务的主要方法有：

1、事务是 PerformanceRunner 度量系统性能指标的唯一手段（没有事务则没有办法衡量系统的响应时间）；

2、事务能够用于度量高风险业务流程的性能指标；

3、事务能够度量在一组操作中每一步的性能指标；

4、通过事务计时实现了不同压力附在下的性能指标对比；

5、通过事务计时可以帮助定位性能瓶颈；

- **集合点**

执行负载测试时，需要模拟系统上有较重的用户负载。要实现此操作，可以同步 Vuser 以便恰好在同一时刻执行任务。通过创建集合点，可以配置多个 VU 同时执行操作。当某个 VU 达到该集合时，将进行等待，直到参与该集合的全部 VU 都到达。指定数量的 VU 均达到后，释放所有这些 VU。

可通过将集合点插入到 VU 脚本来指定回合位置。在 VU 执行脚本并遇到集合点时，脚本将暂停执行，VU 将等待控制器或者控制台的允许以继续执行。VU 从集合释放后，将执行脚本中的下一个任务。

注意：只能向 Action 部分添加集合。

插入集合点是为了衡量在加重负载的情况下的性能情况。在计划中，可能会要求系统能够承受 1000 人同时提交数据，在 PerformanceRunner 中可以通过在提交数据操作前面加入集合点，这样当虚拟用户运行到提交数据的集合点时，PerformanceRunner 就会检查同时有多少用户运行到集合点，如果不到 1000 人，

PerformanceRunner 就会命令已经到集合点的用户在此等待，当集合点等待的用户达到 1000 人时，PerformanceRunner 命令 1000 个 UV 同时去提交数据，从而达到计划中的需求。

- **检查点**

测试的目的是检查数据是否正确。

在测试的过程中，我们需要检查某个组件的某些属性满足某个条件。这个检查的位置和条件，我们称为检查点。

在 PerformanceRunner 中，使用 check（“objectname”， “property”，“期望值”)来作为检查点的脚本语句，它检查对象 objectname 的属性 property 是否和期望值一致。

在使用中，可以使用检查点来检验系统的各个方面，如数据库、GUI 属性等。

- **参数化与数据驱动**

测试脚本是针对一个测试过程的。一个测试过程往往需要众多的数据来测试。通过录制得到的脚本，所有的输入数据都是常数，是固定的。

如果需要使用一个测试脚本测试多组数据，就需要对脚本进行参数化，把固定的常数修改为来自数据源变量。

这个过程我们称为参数化。

采用了参数化的脚本，我们称为数据驱动的模式。

2.3 业务提供

所谓业务提供，就是指使用性能测试软件 PerformanceRunner 能够提供的功能，以及用来解决哪些问题。

2.3.1 PerformanceRunner 适用性说明

PerformanceRunner 是产品家族中的一个产品，它的测试脚本与自动测试软件 AutoRunner 的兼容（从脚本的角度上看，他们是完全相同的，包括很多内置函数和使用方法）。

PerformanceRunner 提供不同的协议支持，以满足不同的测试需求：

2.3.2 性能测试

通过模拟生产运行的业务压力量和使用场景组合，测试系统的性能是否满足生产性能要求。即在特定的运行条件下验证系统的能力状况。

2.3.3 负载测试

在给定的测试环境下，通过在被测试系统上不断增加压力，直到性能指标超过预定指标或者某种资源使用已经达到饱和状态，目的是了解系统性能容量和处理能力极限。负载测试的主要用途是发现系统性能的拐点，寻找系统能够支持的最大用户、业务等处理能力的约束。

2.3.4 压力测试

测试系统在一定饱和状态下系统能够处理的会话能力，以及是否出现错误，一半用于稳定性测试。可以理解为资源的极限测试。测试关注在资源出于饱和或者超负荷的情况下，系统能否正常运行，是一种在极端压力下的稳定性测试。其主要意义是通过测试调优保证系统即使在极端的压力情况下也不会出现错误甚至系统崩溃。

2.3.5 配置测试

通过对被测系统的软硬件环境的调整，了解各种不同环境对性能影响的程度，从而找到系统各项资源的最有分配原则。主要用于性能调优，在经过测试获得了基准测试数据后，进行环境调整（包括硬件配置、网络、操作系统、应用服务器、数据库等），再将测试结果与基准数据进行对比，判断调整是否达到最佳状态。

2.3.6 并发测试

模拟并发访问，测试多用户并发访问同一个应用、模块、数据时是否产生隐藏的并发问题，如内存泄漏、线程锁、资源争用问题。测试目的并非为了获得性

能指标，而是为了发现并发引起的问题。

2.3.7 可靠性测试

通过给系统加载一定的业务压力的情况下，让应用持续运行一段时间，测试系统在这种条件下是否能够稳定运行。需要和压力测试区分开，两者的测试环境和测试目的不一样。压力测试强调在资源极限情况下系统是否出错，可靠性测试强调在一定的业务压力下长时间（如 24×7）运行系统，关注系统的运行情况（如资源使用率是否逐渐增加、响应是否是否越来越慢），是否有不稳定征兆。

2.3.8 特性概述

Java 标准语法脚本。PerformanceRunner 作为性能测试软件，采用最流行的 Java beanshell 脚本语言作为测试脚本，脚本更简单，降低了测试人员的学习成本，也能够在测试人员学习测试工具的过程中学会 java 的基本知识。

强大的录制功能。支持通过一次录制来实现对各种数据、报文的录制，基本上不需要修改就可以直接执行，降低了测试人员修改脚本的工作量。对于测试过程中遭遇不断回放错误的测试人员来说，是非常有价值的。

自动关联技术。PerformanceRunner 支持关联，通过关联能够解决在 B/S 系统中 session 处理，能够自动实现管理 session，不需要脚本修改。

简便的脚本。对于使用 java 语言作为脚本，很多测试人员可能会担心过于复杂。实际上，所有的测试脚本都是继承一个标准的类 TestCase，并且使用它提供的基本方法，因此是非常简单的，没有复杂的 java 成分，便于那些已经学习过其他测试工具的测试人员迁移到这个工具上来。

数据驱动。PerformanceRunner 支持测试脚本的数据驱动功能：录制脚本完成之后，很容易的实现数据驱动，支持 excel 格式的数据源。PerformanceRunner 还提供了一个数据驱动框架，便于测试人员使用。

良好的扩展性。一般的脚本虽然很简便，但是对于特殊的测试，往往需要更复杂的功能，例如：需要对网络上的另一台系统中的数据库的某些数据进行同步。基本的 PerformanceRunner 不提供这个功能。由于 PerformanceRunner 使用了标准的 java（目前为最新的 JDK1.5）那么用户可以自己编写一个同步方法（或者

类）加入到系统中来使用，只要是 java 已经提供的功能，都可以得到完善的支持。

标准化。PerformanceRunner 符合测试工具的基本要求，如：同步点、验证点、错误报告等，都遵守了国际化测试标准，便于用户理解和使用，也便于用户比较各个不同测试工具之间的差异。

2.4 产品设计目标

1. 系统上线前的性能测试。

在系统上线投产之前，特别是在系统测试阶段，进行性能测试具有非常重大的意义：它可以保证在系统上线之后的正常运行，避免发生性能瓶颈，保证安全生产。

PerformanceRunner 能够最大程度的实现系统测试后阶段的性能测试，降低系统上线风险。

2. 系统日常运维的正常保证。

在系统日常工作中，由于数据的变化、环境的变化、网络以及参数的变化，都会对性能造成影响。

因此，保持每个季度或者每个月份的性能测试，对安全生产起到非常重要的作用。比如，在系统小的参数调整或者小版本升级之后的性能测试后，是非常有意义的。PerformanceRunner 将帮助您实现日常性能测试。

3. 具有一致性和可重复性。由于性能测试是自动化实现的，因此每次自动化测试运行的脚本是相同的, 所以每次执行的测试具有一致性, 人是很难做到的. 由于自动化测试的一致性，很容易发现被测软件的任何改变。

3.系统体系结构特性要求

3.1 系统要求

操作系统环境：

Windows XP

Windows2000

Windows 2003

注：理论上对于安装了 jdk1.6 的 windows 系统都提供支持。

系统要求：

JDK1.6

IE5.5 以上（针对 IE 的 plugin）

3.2 系统性能

PerformanceRunner 针对与系统的性能测试自动化，一半可以达到单机 2000 个/秒的并发量。

系统的并发上限与硬件和软件系统的配置有关，一半而言，配置更高的机器可以得到更好的性能。

3.3 扩展能力

- **扩展验证点**

所谓的验证点，就是用来验证被测试系统返回数据或者状态是否和预期一致的点。

PerformanceRunner 提供了完整的验证点功能，用来验证字符串、bitmap 文件是否正确，对字符串可以验证是否符合定义的“正则表达式”。

当然，由于验证往往是非常复杂的，例如：当我们使用一个功能向 database 中增加一条记录后，通过 jdbc 来查看该记录是否已经被增加。这就需要用户根据具体的数据库来编写一个功能来实现特殊的校验点。

系统提供了基本的校验方法，允许用户自己来通过编写一个特殊校验的类，或者一个特殊的方法来定义特殊的校验点（调用的结果如果希望反映的标准的测试报告中，就需要调用系统提供的基本方法），最终实现对验证点功能的扩展。

- **对第三方测试管理工具的支持**

PerformanceRunner 提供了对第三方测试管理工具的支持：通过数据文件或者数据库，就可以传递测试案例信息、测试案例数据信息。

PerformanceRunner 提供了命令行的支持，支持用户在远程启动和调用，这就为第三方的测试管理工具提供了一个执行调用接口。

- **对第三方缺陷跟踪工具的支持**

同样的，PerformanceRunner 可以提供针对缺陷跟踪工具的 API 的调用，和第三方缺陷跟踪工具达到“无缝连接”。

3.4 可靠性和可用性

系统的可用性和可靠性由几个指标来衡量：

1. 系统的出错处理能力。也就是，当系统出现错误之后，是否能够提供完善的错误处理机制，跳过错误，继续执行允许执行的下一个功能点测试。
2. 系统执行过程中工具不会出现异常，导致测试无法正常执行。
3. 测试脚本出现异常，提供强大的调试功能。
4. 当 PerformanceRunner 升级之后，原有测试脚本能够兼容，继续使用。

具体到 PerformanceRunner，如下：

- **系统的出错处理能力**

在进行压力测试过程中，需要大批量的贫乏执行测试脚本。一个测试脚本失败之后，对后续的测试脚本执行没有影响，仍然可以继续执行，最后在执行统计中会分析执行报告。

- **产品升级**

产品提供向下兼容，支持对脚本的扩展，但是不会替换原来的内置方法。另外测试脚本采用 java 的标准语法，java 本身也提供了脚本的向下兼容。

3.5 国际支持

支持多种语言 Unicode 编码形式；

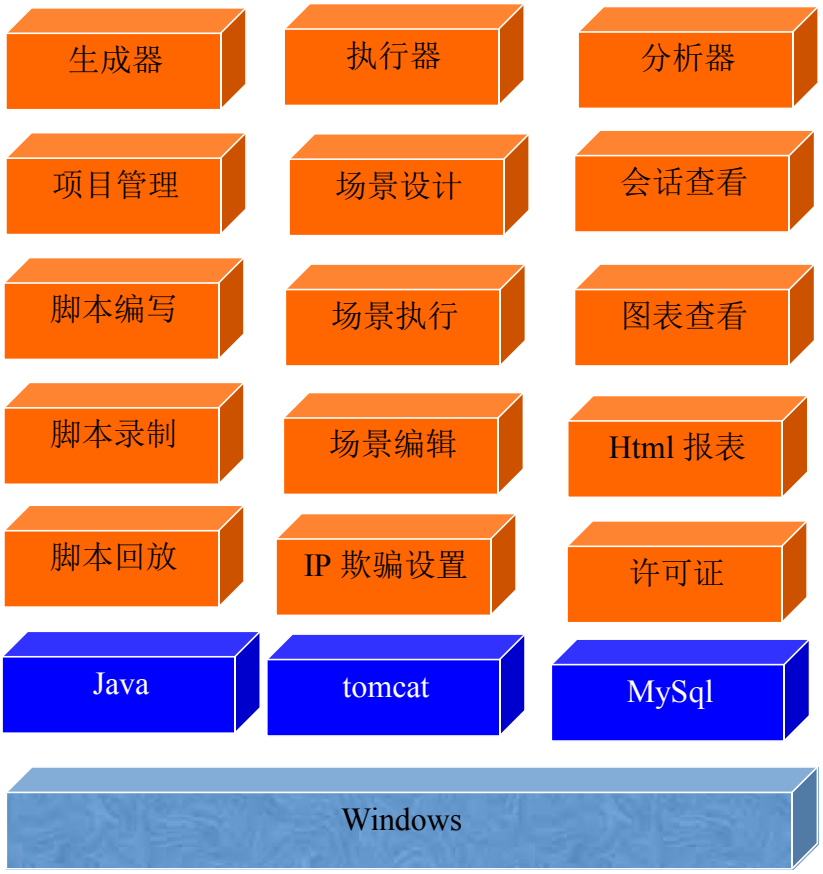
用户可以选择中英文界面的版本。

系统对语言编码的识别是由系统自动完成，用户不必考虑选码的问题。

4.系统架构

PerformanceRunner 主要由生成器、执行器、分析器三大模块构成。生成器主要用来进行项目管理、脚本编写、录制、回放、添加事务、集合点、查看对象库等操作。执行器主要

用来进行场景新建、打开、指定场景计划、运行场景、IP 欺骗设置、场景编辑等操作。分析器模块主要用来对已运行的场景数据进行分析，主要包括虚拟用户图、事务图、点击量、吞吐量、CPU 使用率、物理内存使用、网络流量等图表。如下图所示为 PR 系统架构图。



5 系统基本功能

5.1 测试项目创建

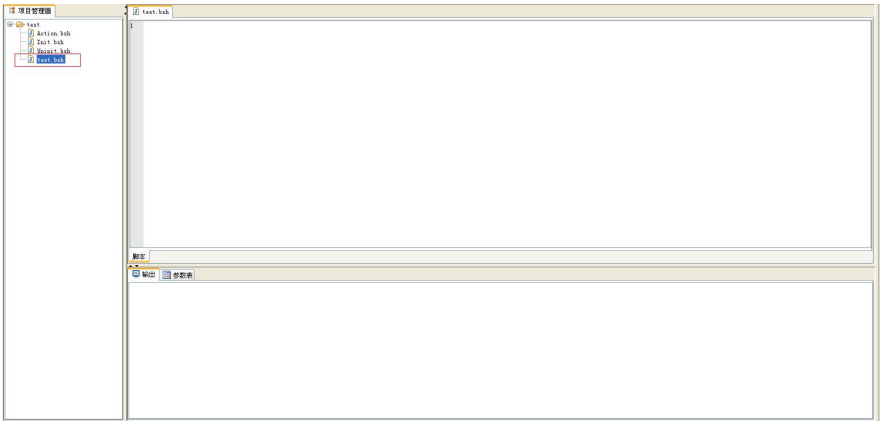
创建测试项目

用户通过生成器“新建”菜单创建测试项目，测试项目新建后包括三个文件 Action.bsh、Init.bsh、Uninit.bsh，用户可通过项目管理器查看这三个文件信息，双击可进入脚本编辑界面，进行测试脚本的编写。如图为测试项目信息。

5.2 测试脚本创建与录制

创建测试脚本

用户能够创建一个测试脚本。创建的测试案例脚本是空的，需要通过录制来获得内容。如图为新创建的 test.bsh 文件。



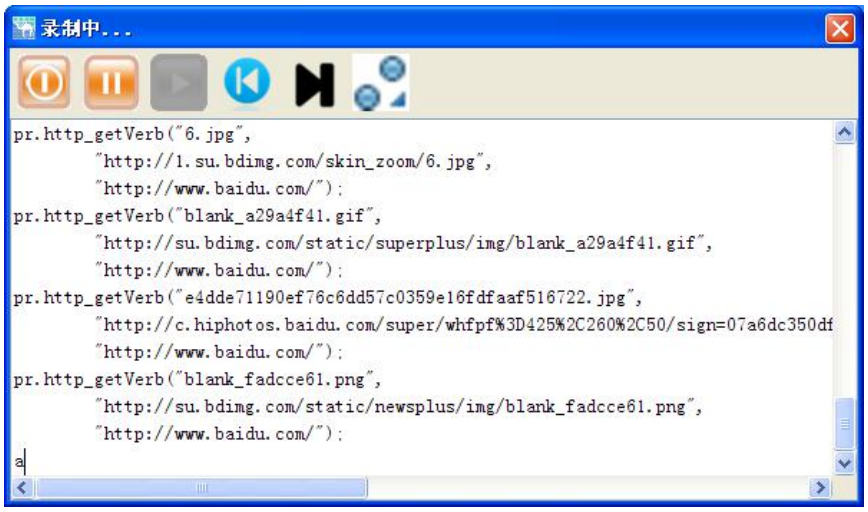
通过录制创建测试脚本

当你从菜单或者工具条启动“录制”命令，系统开始记录你的所有操作，并且在记录过程中把生成的脚本文件显示在编辑器上面。

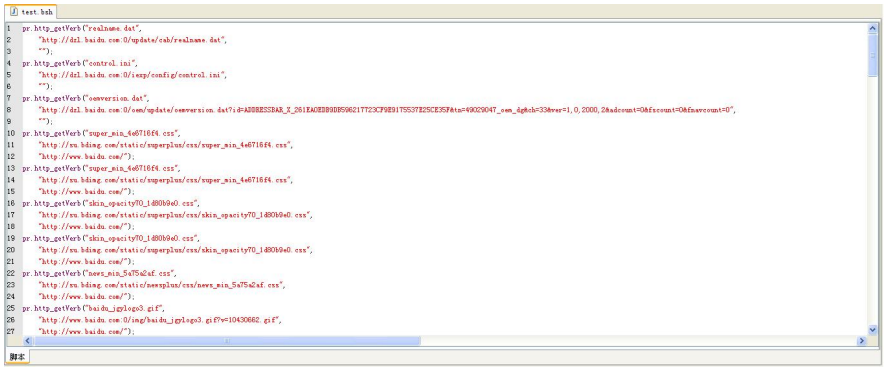
录制的结果是，你得到了一个可以被执行的测试脚本文件，包括通过协议发送数据和接收收据的过程。录制分为两个步骤，第一步，进入开始录制窗口指定相应的参数，如图：

如上图所示，指定录制协议为 Http，需要录制的程序指定为："C:\Program Files\Internet Explorer\iexplore.exe"，程序的输入参数为 www.baidu.com，并指定程序启动就录制、全新录制选项。

第二步，点击确定按钮，开始录制操作，如图：



如上图所示为 PerformanceRunner 录制窗口，通过录制用户操作，生成录制脚本，另外，录制窗口提供了个性化按钮，方便用户停止录制、暂停录制、插入事务、集合点信息。录制结束，点击“停止录制”按钮，PerformanceRunner 将返回脚本界面，如图：

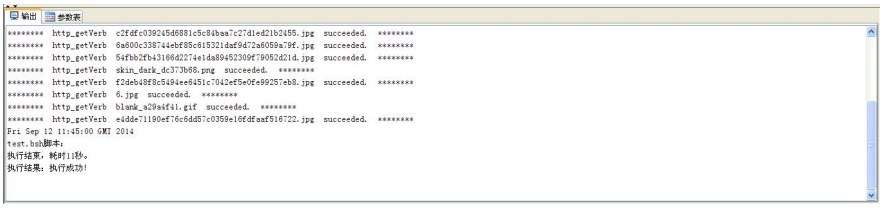


5.3 录制结果回放

录制结果回放

录制结束后，PerformanceRunner 会自动生成录制脚本，为了验证脚本的正确性，我们需要对脚本进行回放（执行）操作。脚本回放通过生成器的回放按钮触发，如图：

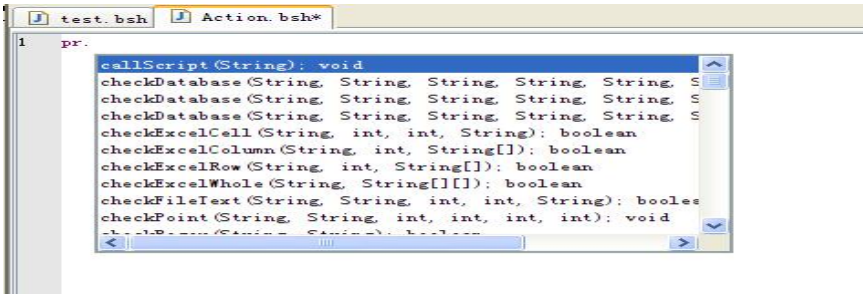
回放结束后，在生成器“输出”窗口会输出脚本执行信息，执行信息会反映出脚本执行成功与否和场景执行所需要的时间，如图：



5.4 测试脚本编辑

测试脚本的结构

根据 Java BeanShell 的规范，测试脚本是被顺序执行的，不需要 function、class 等，使用非常简便。对于没有学习过 java 的用户，是非常简单的。如图为测试脚本编辑界面，performancerunner 对测试脚本编辑提供了类似 Eclipse 那样的代码提示功能，该功能可以极大的方便用户进行脚本编辑，如图：



测试脚本编辑

PerformanceRunner 提供了强大的测试脚本编辑功能：第一，提供了 java 脚本的关键字识别技术，能够识别系统的关键字，避免语法错误；第二，提供了实时语法分析的功能，在编辑过程中动态分析语法，并且对语法错误动态报警，尽量避免编译时刻再出现错误。

5.5 测试案例参数化

什么是数据驱动？

如果一个测试脚本只能够被执行一组数据，并且数据是固定不变的，那么你每一次的测试就只能够执行很简单的功能了。

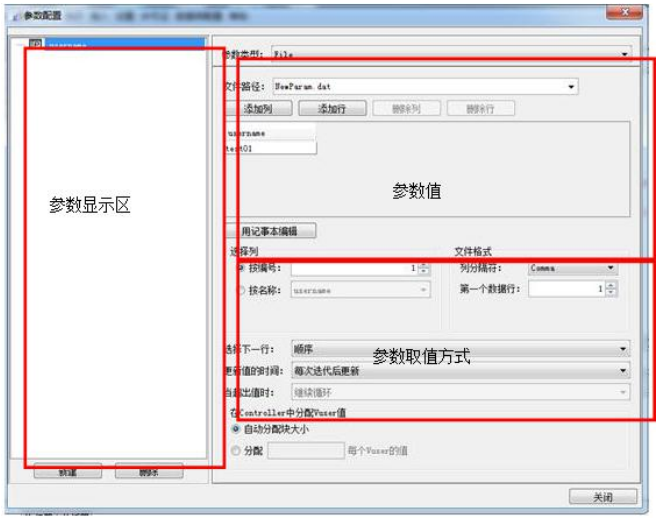
边界条件、路径覆盖，需要使用一个脚本、很多组数据输入才能够完成，固定的数据无法满足要求。

数据驱动就是指能够把需要输入（和验证）的数据参数化，通过脚本执行不同的数据，就实现了数据驱动，也就是数据与脚本分离。

PerformanceRunner 实现了脚本与数据分离：脚本使用 java 的脚本，在脚本执行的时候，从数据源中读取数据。

参数化

一条普通脚本只能执行某个特定的动作，将脚本参数化后则可以执行不同的功能。脚本参数化之前，必需要编辑好参数配置，下面是参数配置的界面：



进入参数配置，可新建或删除参数，设置参数值、设置参数取值方式等

新建：新建一个参数，默认名称为 NewParam，双击名称可修改；

删除：删除已有的参数；

参数类型：可选择参数类型为 file、data/time、number、Vuser ID 等类型

文件路径：显示文件保存的路径

添加列/添加行：可添加列或者行来编辑参数值。添加行即添加一个参数；

用记事本编辑：可用记事本打开参数表，对参数值进行编辑，方便操作；

选择列：设置参数取值的标准：根据编号或者跟踪名称来取值，列分割符：定义每列如果分割；第一个数据行：设置从第几行开始取值；

选择下一行：顺序-根据顺序从参数表内取值；随机-随机取值； 唯一取值：每次执行只取一个值；相同于【参数名】：与其他参数取值方式相同，适用于参数值相对应时；

更新值的时间：设置取值时间，可选择每次迭代后更新，每次访问更新、只更新一次；

参数设置配置完成后，在脚本内设置参数取值，方法如下：

将需要参数化的值替换为“pr.getParamValue("username");”其中 username 为参数名；具体操作如图所示：

```
19 pr.http_postVerb("login.do",
20     "http://103.38.234.24:8080/ProjectCenter/login.do?oper=login&index=0",
21     "http://103.38.234.24:8080/ProjectCenter/newlogin.jsp",
22     "userName="+pr.getParamValue("username")+"&password=111111");
23
24
```

图例是一个登录的脚本，从参数表内取登录名 username 的值。在脚本内执

行默认取第一个值，在场景内择根据参数取值方式取值，从而更真实的模拟多人访问系统的场景。

5.6 关联动态内容

解决什么问题？

在有些情况下，录制好的脚本直接回放时会发生交易不成功的情况，这是由于在发起 HTTP 请求时，其中 Cookie 中和 html body 中存在会变的动态内容，所以需要在回放前对录制好的脚本进行关联处理。

关联处理就是在录制脚本完成之后，回放脚本之前，通过关联的动作找到录制过程和关联过程 Cookie 快照和 Body 快照不同的部分，然后将这部分参数化的过程。关联的过程本身是发起一次网络请求。完成关联后，PerformanceRunner 能够实现将脚本中所有网络请求里的动态内容全部找出来并参数化，然后在回放的时候放入正确的 Cookie 和 Body 以确保请求能够正确得到服务器的响应。

Cookie 快照

发起 HTTP 请求时 Cookie 的内容

Body 快照

发起 HTTP 请求时，html 中 body 的内容。

5.7 增加同步点和验证点

验证点

测试的目的是看执行一个过程，结果是否和预期结果一致。

验证的方法就是查看结果是否一致，这个点我们称作“验证点”。

验证成功则继续执行，验证不成功也需要继续执行，并且把结果写入测试报告。

PerformanceRunner 的验证点需要手工加入——PerformanceRunner 不知道用户需要验证那些内容。

增加验证点

用户可以使用编辑器来增加验证点，PerformanceRunner 提供了方法让用户来增加验证点。

5.8 建立测试场景

在第 2.3 节我们讲了本产品的业务提供，Performance Runner 能够满足的测试类型涵盖了：性能测试、负载测试、压力测试、配置测试、并发测试、可靠性测试等。这些业务提供的方向体现了不同的性能需求和测试目的，为了实现不同的测试目标，需要为性能测试建立测试场景。

Performance Runner 的测试场景包含场景组、场景计划两部分。

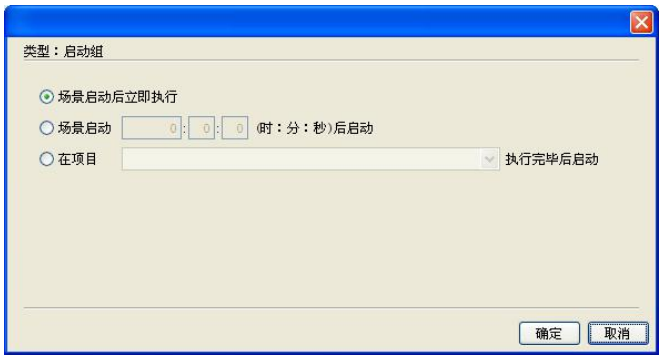
场景组

场景组是一组测试项目组成的集合，每个测试项目对应了 Init、Uninit 和 Action 脚本，其中 Init 脚本和 Uninit 脚本分别在项目开始执行和结束执行时刻被调用一次，Action 脚本将被反复调用，Performance Runner 的执行器通过执行场景组中的性能测试脚本实现测试。在 PerformanceRunner 中，一个场景可以对应一个或多个项目，如图可以为场景指定项目：

场景计划

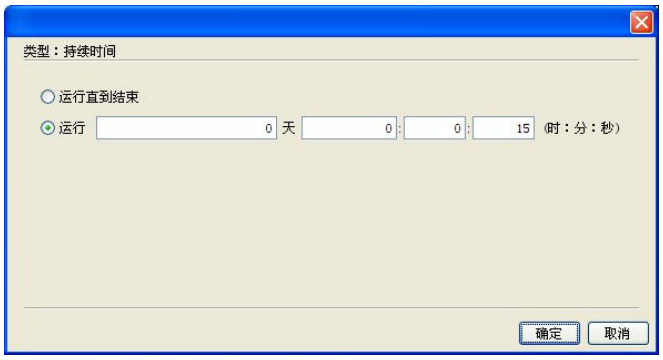
场景计划是为场景组中的测试项目提供的配置项，这些配置项用于定义项目执行的特征，包括：启动组、启动虚拟用户、持续时间、停止虚拟用户等。如图为场景计划配置表格：

启动组：用户双击“启动组”列进入“启动组”配置，可选的启动组选项有“场景启动后立即执行”、“场景启动指定时间后启动”、“在指定项目执行完毕后启动”，如图为启动组配置界面：



启动虚拟用户：通过该界面可以指定启动的虚拟用户数目，启动模式有两种，“同时启动”、“指定时间启动指定数目的虚拟用户”，如图：

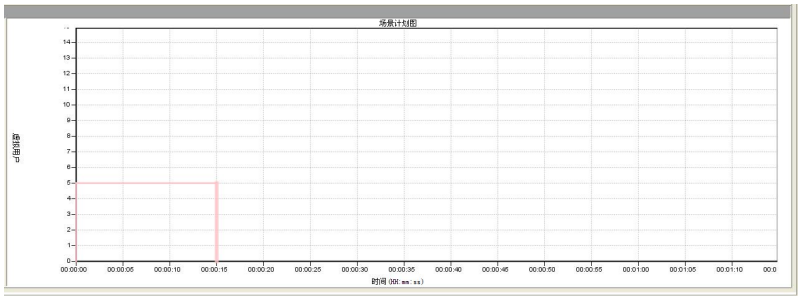
持续时间：持续时间有两个选项，“运行直到结束”、“指定时间”，如图为持续时间配置界面：



停止虚拟用户：用于指定虚拟用户停止策略，可选的停止策略有“同时停止”、“指定时间内停止指定数目的虚拟用户”，如图：

场景计划图

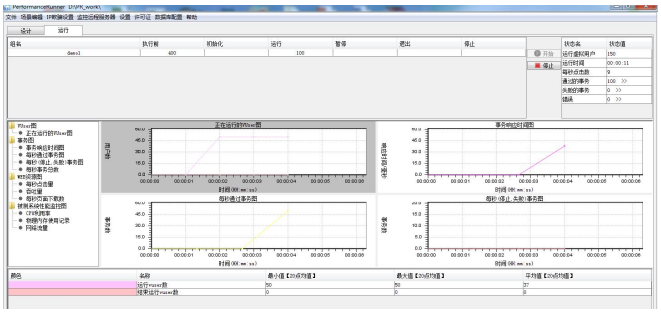
场景计划图便于用户直观查看自己制定的场景计划，如下图所示，设置场景运行 15s，虚拟用户最大值为 0，虚拟用户同时启动、同时停止：



5.9 测试场景执行

脚本执行的目的在于验证脚本的正确性，而性能测试的需求和目标最终是通过执行测试场景实现的，比如说在进行压力测试时可以录制一段脚本，这段脚本包含了需要加压的网络请求代码，并且按照要求建立好事务，在请求代码前增加了集合点，然后需要做的是建立场景，并按照性能模型准备足够多的 VU 虚拟用户，然后根据性能模型设置一段加压时间。最后保存并执行场景，Performance

Runner 会根据场景中制定的执行计划执行性能脚本，并通过动态图表监视执行的结果，可以用来监视执行过程的动态图表包括有：正在运行的 VU 图、事务响应时间图、每秒通过事务图、每秒（停止、失败）事务图、每秒事务总数图、每秒点击量图、吞吐量图、每秒页面下载量图、CPU 使用率、物理内存使用、网络流量等内容。如图为场景运行界面：



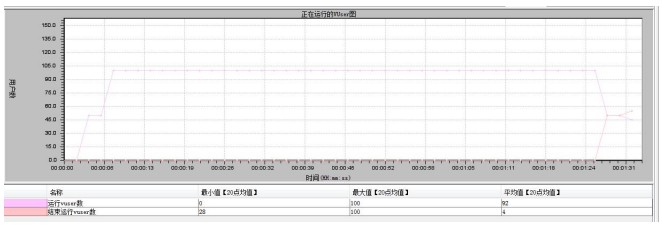
5.9 测试分析

场景执行完成后，Performance Runner 将统计图表记录下来作为分析系统性能的依据，用户可以通过保存的场景名称访问到该场景最后一次执行的结果。统计图表的种类与测试场景执行模块的动态图表一致，这里的统计图表是执行完成后动态图表的最后结果。

5.10 统计图表

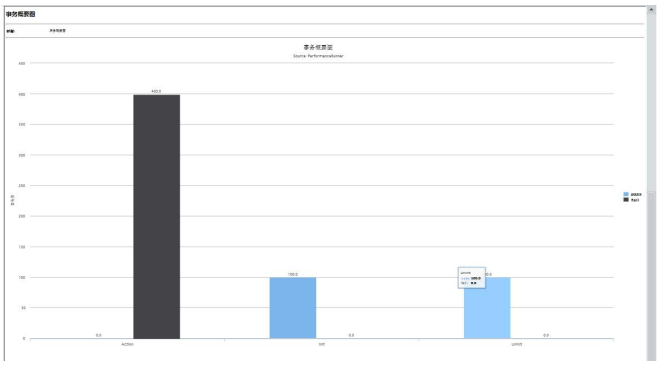
运行的 VUser 图

运行的 VUser 图用于显示每个时刻同时运行的用户个数，是一个反映并发数量和水平的图表。在统计图表中还提供一个表格记录 VUser 图的统计信息，这些统计信息分析运行过程中出现的最小用户数、最大用户数、平均用户数、中间用户数等。运行的 VUser 图如下所示：



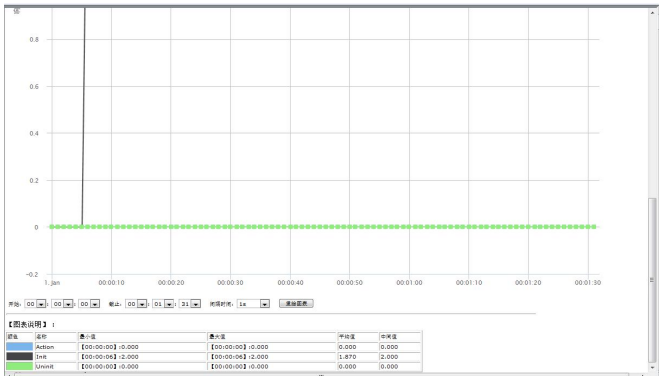
事务概要图

事务概要图用于显示每个事务执行的情况和结果。横轴代表不同的事务，纵轴代表各个事务执行成功和失败的次数（绿色代表成功，红色代表失败）。和 VUser 一样，事务概要图表中也会提供统计信息。事务概要图如下图所示：



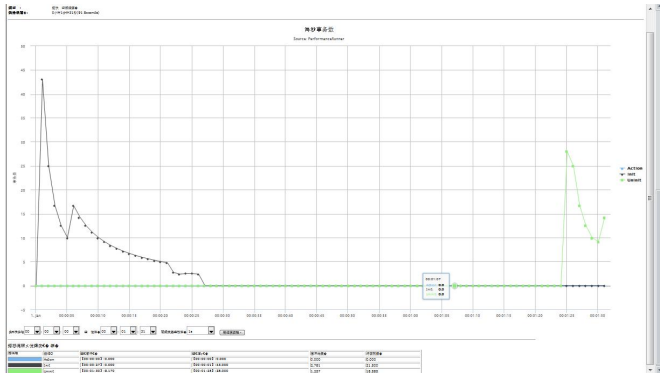
事务响应时间图

事务响应时间图用于显示每一个时刻各个事务的响应时间，是一个反映事务响应快慢以及随时间变化趋势的图表。事务响应时间图表如下图所示：



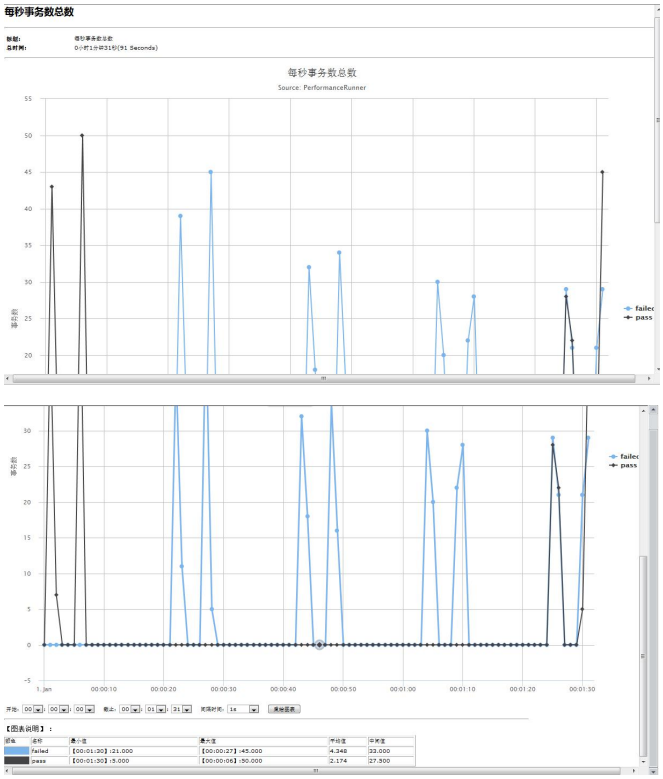
每秒事务数图

每秒事务数图用于统计各个时刻每个事务每秒钟通过的次数，是一个反映事务处理效率以及随时间变化趋势的图表。每秒事务数图表如下图所示：



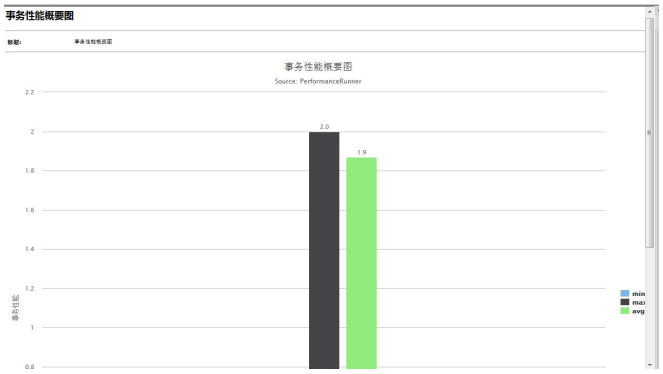
每秒事务数总数图

每秒事务数总数图与每秒事务数图统计的对象是一样的，不过这个图表还包括执行错误的次数统计。每秒事务数总数图表如下图所示：



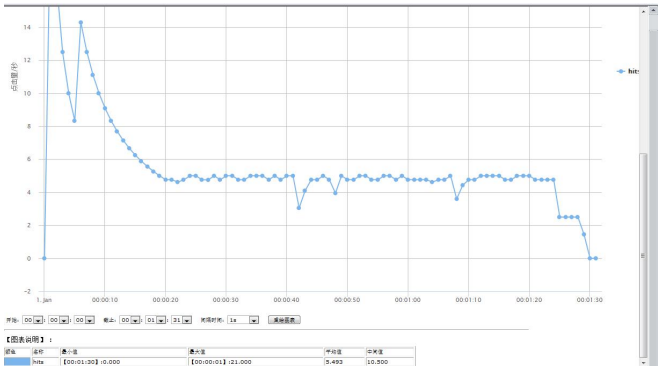
事务性能概要图

事务性能概要图是一个分析图表，显示在整个执行过程区间内执行成功次数的最大值、最小值和平均值。事务性能概要图表如下图所示：



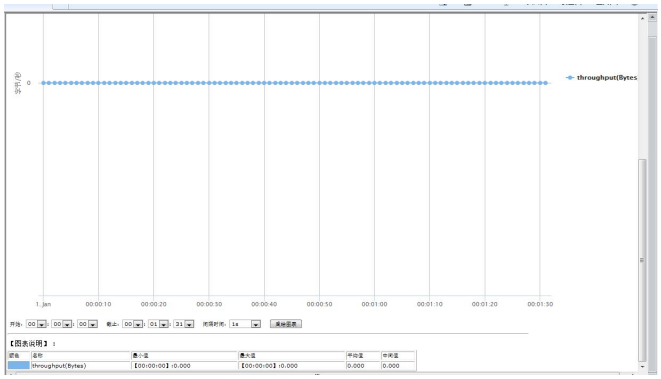
每秒点击量

每秒点击量统计各个时刻发生的每秒请求数。每秒点击量图表如下图所示：



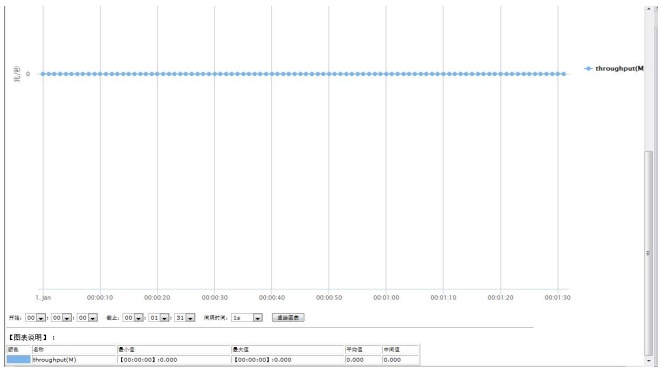
吞吐量（字节）

吞吐量（字节）图统计了各个时刻发生的每秒服务器发送的字节数，包括了请求的字节数和响应的字节数，是一个反映系统处理请求能力的重要指标。吞吐量（字节）图表如下图所示：



吞吐量（兆）

吞吐量（兆）图统计了各个时刻发生的每秒服务器发送的字节数，包括了请求的字节数和响应的字节数，是一个反映系统处理请求能力的重要指标。吞吐量（兆）图表如下图所示：



CPU 使用率

CPU 使用率图统计了被测系统在受到 PR 施压后 CPU 的使用情况，CPU 使用率

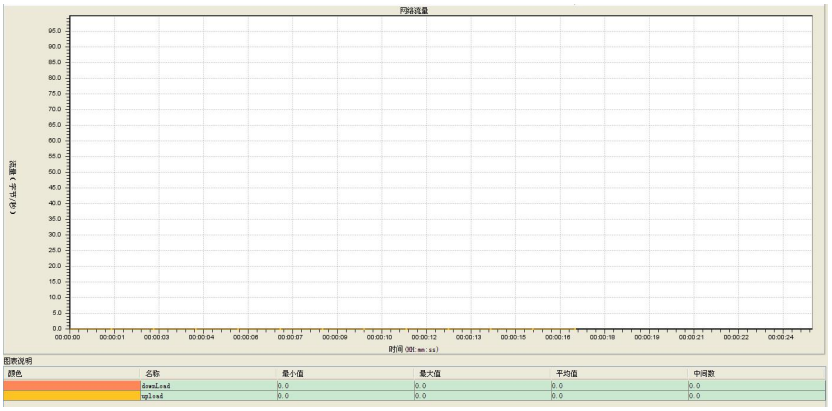
图表如下图所示：

物理内存使用

物理内存使用率图统计了被测系统在受到 PR 施压后物理内存的使用情况，物理内存使用率图表如下图所示：

网络流量

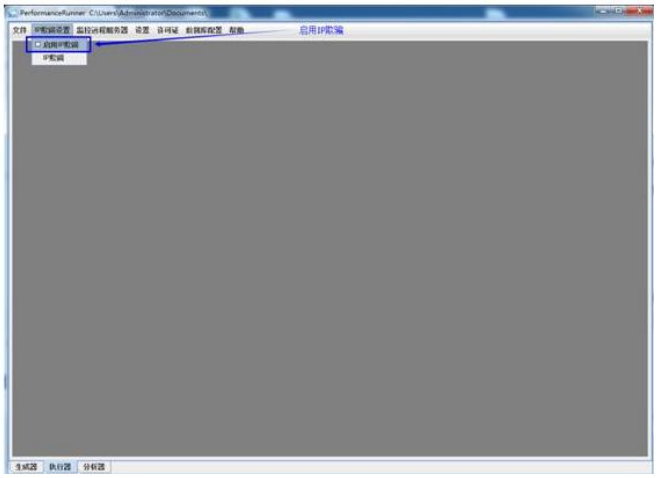
网络流量图表统计了被测系统在受到 PR 施压后网络上传、下载情况，网络流量图表如下图所示：



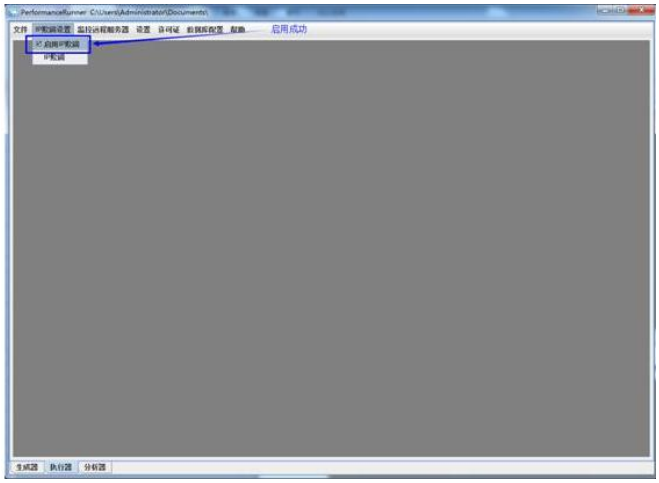
5.11 IP 欺骗

IP 地址欺骗是指产生的 IP 数据包为伪造的源 IP 地址，以便冒充其他系统或发件人的身份。按照 Internet Protocal (IP)网络互联协议，数据包头包含来源地和目的地信息。 而 IP 地址欺骗，就是通过伪造数据包包头，使显示的信息源不是实际的来源，就像这个数据包是从另一台计算机上发送的。PerformanceRunner 通过为本机绑定多个 IP 地址，在访问被测系统时实现了 IP 欺骗。在 PerformanceRunner 上实现 IP 欺骗主要分为以下几个步骤：

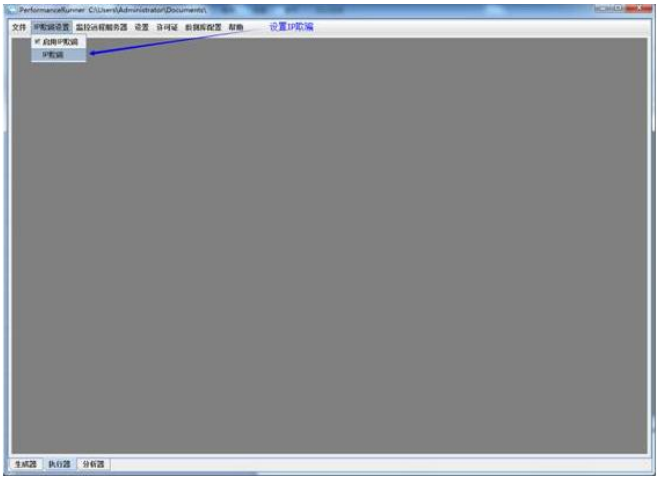
1、点击【IP 欺骗设置】→【启用 IP 欺骗】



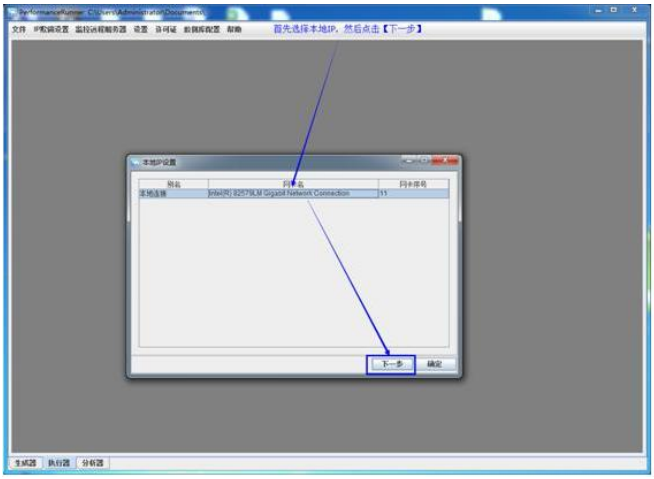
待“启用 IP 欺骗”前面被勾选，启用成功

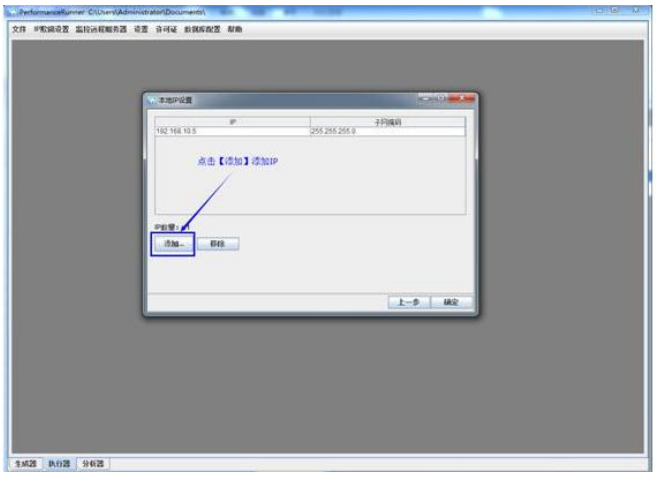


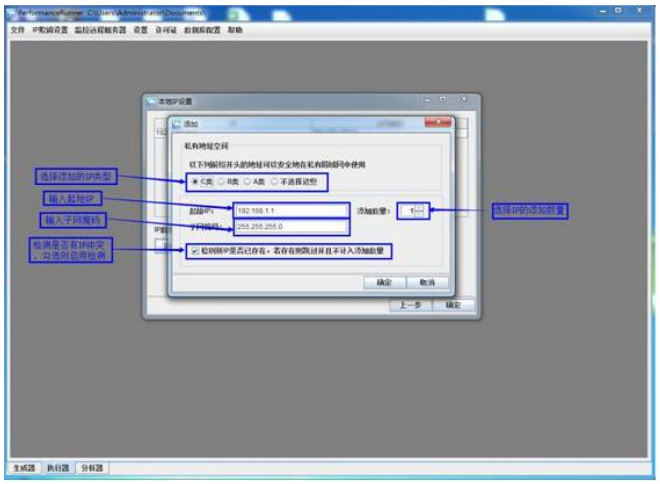
2、点击【IP 欺骗】



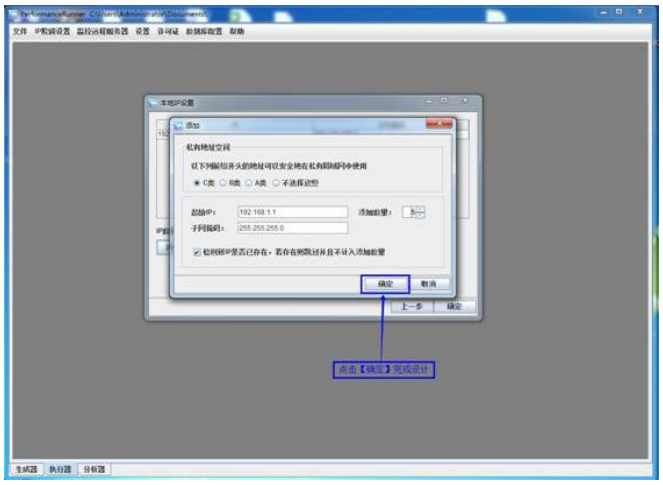
弹出【本地 IP 设置】窗口。选择本地 IP，然后点击下一步

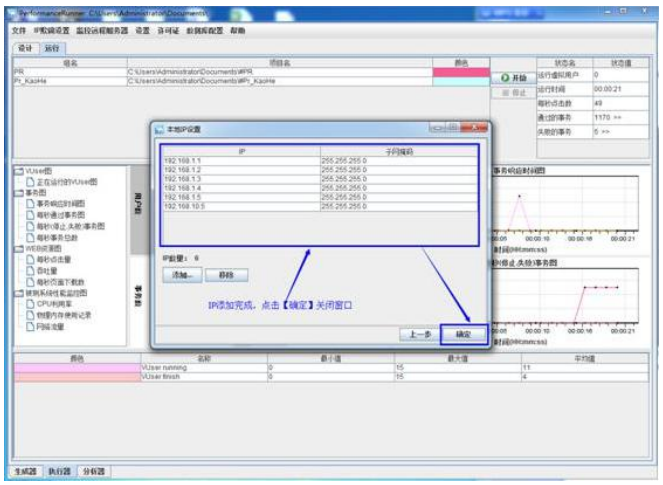






选择 IP 类型，输入起始 IP，子网掩码，选择是否启用 IP 冲突检查，选择 IP 添加数量，点击【确定】结束设计，【添加】窗口关闭





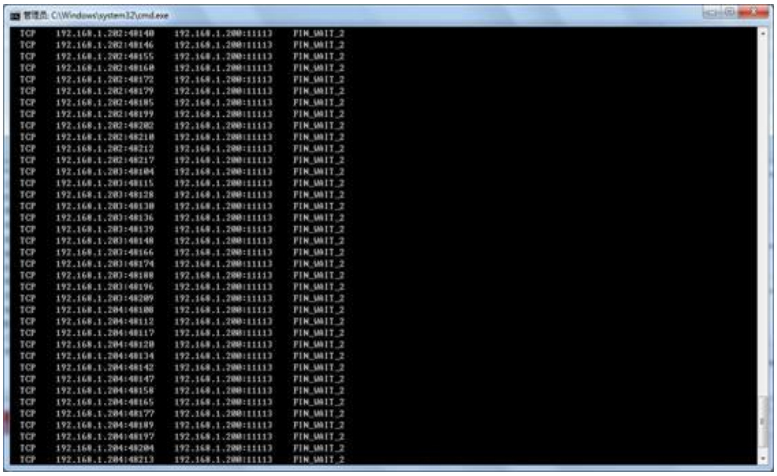
打开 windows cmd.exe，输入“ipconfig”命令，可以查看到新添加的 ip 地址信息

至此，ip 欺骗配置结束

IP 地址欺骗使用状况：

完成 IP 欺骗设置后，在 pr 运行界面，加入项目，点击“开始”按钮，执行场景。

在场景执行时，打开 windows cmd.exe，输入“netstat -n”，可以查看到虚拟 ip 的访问情况



5.12 系统性能监控

通过被测系统所在服务器的 IP 地址、用户名、密码，在 PerformanceRunner 访问被测系统过程中，可实时获取到被测系统 CPU、物理内存、网络流量信息，用户可通过这些信息了解到被测系统更为确切的性能指标。如图为监控远程服务器界面：

6.PerformanceRunner 的特点

评估自动测试工具的关键在于：第一，很高的建立测试案例的生产率；第二，降低用户的二次开发成本；第三，便于维护使用；第四，便于测试案例的数据驱动扩展；第五，测试案例资源的延续性；第六，扩展性。

下面，我们就 PerformanceRunner 在这几个方面的特点简要介绍：

PerformanceRunner 具有很高的生产率。自动测试工具建立一个测试案例脚本的时间成本为手工测试一次的 3－10 倍，可见建立自动测试的起始是需要一定的成本的。

如何降低建立测试案例的成本，是自动测试工具的关键。PerformanceRunner 的优势在于：首先，优秀的自动识别组件功能。脚本能够在录制完成之后直接使用，能够自动适应出现的各种情况，如：窗口位置、title、大小等的变化，组件位置、名称的变化。通过自动识别能够识别处组件，从而降低对编写脚本的要求，提高了自动录制的可用性。第二，提供了数据驱动框架。很多测试工具虽然支持参数化的功能，但是需要手工完成数据驱动框架，才能够实现数据驱动：从指定的文件中获取数据。PerformanceRunner 自动定义标准的数据驱动模式，定义了标准的数据驱动格式，降低了增加测试案例的成本。虽然建立一个测试脚本需要一定的时间，但是在测试脚本建立之后增加一组数据的时间却非常短。

模糊识别。PerformanceRunner 对每种组件定义了标准的模糊识别指标。在录制测试案例之后，系统的资源文件就会根据系统的配置文件生成确定识别权重的指标。在测试脚本被执行的时候，通过权重算法来进行模糊识别和匹配。

关键字驱动。PerformanceRunner 提供了领先的关键字驱动技术，支持脚本编写使用专家视图，不熟悉脚本的用户使用关键字视图，并且实现在脚本视图与关键字视图之间的相互转换，既提升了效率，也提升了易用性，既能够给熟悉脚本的测试工程师提供高效的工作平台，也能够给不熟悉测试脚本的测试工程师使用方便。

用户可以对系统配置文件的识别参数进行调整，达到修改整个录制脚本识别参数权重的目标，便于提高整个项目中脚本开发的效率。

在用户录制完成脚本之后，可以对对应的资源文件的权重属性进行修改，使系统能够定制具体的模糊识别对象，对脚本组件识别算法作特殊处理。

通过模糊识别算法，能够极大地提高脚本执行的可靠性，对于由于类似组件位置、大小等变化之下的脚本执行，能够起到非常良好的效果：用户不需要因为界面小的修改而导致来修改测试脚本。

便于维护使用。案例完成之后，随着应用系统的修改、应用系统版本的提升，同样需要维护这个测试用例库，因此维护使用是非常重要的功能。

维护方便性主要体现在几个方面：简洁的框架、容易理解的脚本、方便的调试功能。

PerformanceRunner 提供了针对测试案例的框架，这个框架包括：案例层次划分(PerformanceRunner 的案例由 Action 组成，每个 Action 包含对一个 Window 的所有操作，PerformanceRunner 允许在案例之间共享 Action 来提高系统的可维护性）、数据驱动框架、自动同步、数据校验模式等。使用这些框架能够非常容易的维护测试案例库。

PerformanceRunner 采用了 java 的语法，测试人员使用的语法非常简单，便于理解和使用。并且，由于系统提供了关键字驱动的框架，所以对一般的维护而言，根本不需要了解 java，只需要知道最基本的操作就可以。

PerformanceRunner 遵守 JDA 的标准，提供了最强大的系统调试功能：从设置断点、单步执行、变量查看、表达式查看等方面提供支持，便于测试人员容易

排除错误。

另外，PerformanceRunner 提供了强大的编辑器，在一般编辑器能够动态识别语法关键字的基础上，还能够同时提供语法检查——在编辑的时候从事语法检查，对错误的语法实时提示。这个编辑器对于比较缺乏编程经验的程序员来说，非常重要。

测试案例资源的延续性和扩展性。测试案例库本身也是一种资源它和应用版本是对映的关系，随着应用系统版本的升级，案例库也会升级，那么回归测试的效果才能够最大化。

对于测试工具来说，要保证这个资源，就需要保证：测试脚本的兼容性。另外由于随着应用的发展，测试工具的功能需要大幅度的提升，因此工具的可扩展性也需要保证才能够保证测试案例资源的延续性。

PerformanceRunner 使用了 java 语言作为基础，并且实现了 java 调试功能，可以随着 java 的发展不断的发展，扩展自己的功能。

采用 java 语言是一个巨大的优势，比测试工具自己使用一种语言要方便的多。从根本上说，PerformanceRunner 不是采用了哪种语言的语法，而是从根本上就是 java 语言。这和采用 vbscript 或者 c 语言语法的工具是截然不同的。

在扩展外部功能方面，由于 PerformanceRunner 使用了 java 语言，允许使用外部的包，也就是说可以任意增加脚本本身的功能而不受语法的限制和工具本身是否支持外部包的限制——在最大程度上提高了扩展性。

7. 厂商支持能力

泽众性能测试软件 PerformanceRunner，我们通过在线 QQ、微信、电话、电子邮件为您提供支持与服务，您也可访问我们的网站 http://www.spasvo.com/ 寻求帮助；为保证服务质量，确保有效地解决用户的问题，保障用户的项目实施进度，技术支持仅向授权用户和授权试用用户提供。请您在联系泽众技术支持时，告知您的单位名称和服务代码。

技术支持

电话：021-60725088-8007

传真：021-60725088-8017

电子邮件：support@spasvo.com

QQ：1404189128

泽众微信公众号

产品服务

有关培训、产品购买及试用授权方法的问题，请与销售代表联系，或联系泽众咨询热线.

电话：021-60725088-8006

传真：021-60725088-8017

电子邮件: sales@spasvo.com

提供完备的用户手册，管理员使用手册，系统技术手册并再系统升级后及时修改更新服务。

厂商能够根据在实际应用中的问题，迅速给予解答（2 小时内），并给出解决方案(48 小时内)。