泽众性能测试软件
PerformanceRunner

技术白皮书

Version4.0

上海泽众软件科技有限公司

2026 年 8 月

文档名称： 泽众性能测试软件 PerformanceRunner 技术白皮书

版本号： V4.0

文档状态： 正式发布

发布日期： 2026 年 8 月

编制单位： 上海泽众软件科技有限公司

版权声明

目 录

1. 总述

1.1 背景与行业趋势

在当今数字化转型加速推进的时代背景下，软件系统已成为企业业务运营的核心基础设施。随着软件应用规模的持续扩张和系统架构复杂度的指数级增长，软件质量对组织业务连续性和竞争力的影响日益凸显。质量缺陷可能导致严重的经济损失，包括账务差错、客户数据泄露、服务中断等一系列连锁反应，直接影响企业品牌声誉和市场地位。

特别是在 Web 应用主导的企业信息化架构中，性能表现已成为衡量系统可用性的关键质量属性。系统响应延迟、吞吐量不足、并发处理能力受限等性能问题，往往在业务高峰期集中暴露，造成灾难性后果。据统计，性能问题导致的系统故障占生产环境事故的高比重，而生产环境中发现和修复性能缺陷的成本，通常是开发测试阶段的数十倍乃至上百倍。

性能测试作为保障系统非功能质量的核心手段，涵盖性能基准测试、负载测试、压力测试、配置测试及并发测试等多种类型，已成为系统正式上线投产前的强制性质量门禁。然而，真实场景下的性能测试需要模拟大规模并发用户访问，依赖人工方式组织数百乃至数千名真实用户同时操作在实操层面几乎不可行，且人力成本极为高昂。因此，采用专业的性能测试自动化工具已成为行业普遍共识和实践标准。

1.2 企业级自动化测试解决方案

构建成熟、可持续的软件质量保证体系，需要建立覆盖全生命周期的自动化测试框架，包括需求可追溯性管理、测试需求分析、测试过程管理、缺陷跟踪闭环等核心能力，并将测试活动深度嵌入软件开发和产品交付的完整流程中。

在 CMMI（能力成熟度模型集成）规范框架下，测试验证是 SQA（软件质量保证）的关键组成部分，是确保软件产品满足既定质量标准的系统性方法。自动化测试体系建设的根基在于测试工具平台的选型与部署——只有采用先进的自动化测试

工具，才能从根本上解决测试效率、覆盖率和可重复性等核心痛点。自动化测试工具的价值主要体现在两个维度：

开发阶段价值： 高效的自动化测试工具为开发人员提供即时的冒烟测试（Smoke Testing）能力，使其在频繁迭代的编码过程中快速验证代码变更的稳定性，实现”开发即测试”的左移（Shift-Left）实践，显著降低缺陷逃逸率。

测试阶段价值： 在系统测试和产品验收阶段，自动化测试工具提供稳定可靠的回归测试（Regression Testing）能力，确保每次代码变更后核心业务流程的完整性和正确性，为产品发布决策提供量化依据。

业界经验表明，测试阶段发现并修复缺陷的投入成本远低于生产环境故障的应急处理成本。遵循”早发现、早修复”的质量经济学原则，上海泽众软件科技有限公司自主研发了拥有完全自主知识产权的自动化性能测试引擎——**PerformanceRunner**，旨在为企业客户提供企业级性能测试自动化解决方案，助力客户构建高可靠、高性能的软件系统。

1.3 文档说明

1. **适用范围：** 本文档适用于上海泽众软件科技有限公司 PerformanceRunner 性能测试平台 V4.0 版本。

2. **文档定位：** 本文档为 PerformanceRunner 产品的权威技术说明文件，是技术商务谈判的核心依据，同时作为采购方进行产品选型、系统测试及项目验收的主要技术参考标准。

3. **技术合规性：** 本文档的编制参考了国家信息产业部颁布的相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技术实践和产品特性进行制定。对于信息产业部已有明确技术标准规定的内容，应优先遵循国家标准，如本文档与国家标准存在差异，以国家标准的新版本为准。

4. **文档修正机制：** 如本文档中的内容描述或技术指标存在疏漏或错误（含印刷错误），经供需双方共同确认后，可对相关内容进行勘误修正。

5. **版本演进声明：** PerformanceRunner 产品版本升级迭代后，上海泽众软件科技有限公司保留对本技术白皮书的修订权利，修订内容不另行通知。

6. **重点考察内容：** 本文档涵盖以下用户重点考察维度：

    - 软件功能规格、性能指标及运行环境要求
    - 系统容量规划与资源配置方法
    - 数据库功能与性能指标
    - 软件安装部署要求
    - 软件接口协议及工程技术规范
    - 供货范围、交付计划、运输、安装调试及培训安排
    - 相关技术文档与资料交付物

7. **知识产权声明：** 本产品涉及的相关专利技术及知识产权，上海泽众软件科技有限公司依法承担相应法律责任。

8. **免责条款：** 本文档提供 PerformanceRunner 产品的技术描述信息，因用户不当使用造成的直接或间接损失，本公司不承担责任。

9. **语言有效性：** 本文档以中文版本为正式有效版本，未经上海泽众软件科技有限公司书面授权，任何其他语言或形式的译本均不具备法律效力。本文档的最终解释权归上海泽众软件科技有限公司所有。

---

2. 系统概述

2.1 产品定位

PerformanceRunner（以下简称 PR）是上海泽众软件科技有限公司自主研发的企业级性能测试平台，专注于大规模并发负载场景下的性能验证与评估。平台提供从测试脚本生成、并发压力模拟、实时监控到测试报告分析的全链路性能测试能

力，帮助企业全面评估目标系统在高并发业务压力下的性能表现、稳定性边界及容量规划依据。

PR 采用协议级仿真技术实现性能测试，原生支持 HTTP、HTTPS、Socket、SIP 等主流网络通信协议。在产品演进路线图中，将进一步扩展对数据库协议、Web Service 等特定应用协议的深度测试支持，以满足更为复杂的性能验证需求。

PR 适用于 B/S（浏览器/服务器）和 C/S（客户端/服务器）架构的多元化应用系统，通过模拟真实终端用户的业务操作行为（Action）构建负载模型，实现对目标系统的高保真性能压力测试。

2.2 核心概念体系

2.2.1 测试脚本（Test Script）

性能测试的核心执行单元为测试脚本，通过大规模并发执行测试脚本来模拟真实业务负载。性能测试本质上属于自动化测试范畴，人工方式难以在时间和精度上满足要求。测试脚本负责处理底层通信协议交互、业务流程编排及并发控制逻辑。PR 提供集成开发环境（IDE），测试脚本在 IDE 中获得完整的开发、调试和执行支持。

2.2.2 自动录制（Auto Recording）

在 B/S 或 C/S 系统的正常操作过程中，客户端与服务器之间持续进行报文交互，这些底层通信数据对用户通常不可见。性能测试工具必须具备网络数据包捕获与解析能力，将用户操作自动转换为可执行的测试脚本。

PR 采用业界广泛应用的 **Java/Groovy 标准语法**作为脚本语言，对于具备 Java 或 JavaScript 基础的测试人员具有极低的学习曲线，可快速掌握脚本开发与维护技能。

2.2.3 虚拟用户（Virtual User, VU）

虚拟用户是 PR 中模拟真实终端用户行为的核心抽象单元。每个虚拟用户代表一个独立的逻辑客户端会话，按照预设脚本执行业务操作流程。通过部署大规模虚

拟用户并发运行，PR 能够在单台或多台测试机上构建数千乃至数万级的并发负载，真实模拟生产环境的高峰业务流量。

2.2.4 事务（Transaction）

事务（Transaction）是 PR 中定义性能度量边界的核心机制。事务的精确定义为：为量化特定业务操作的性能表现，在操作的起始点和终止点之间划定的一个逻辑执行区间。

事务的核心作用： - 当 PR 执行至事务起始标记时自动启动计时器，到达事务结束标记时停止计时，将测得的执行时长记录为事务响应时间。 - 事务是 PR 度量系统性能指标的唯一标准化手段，无事务定义则无法进行响应时间量化分析。 - 事务支持对高风险关键业务流程的专项性能度量。 - 事务可嵌套使用，支持对复杂业务操作中各步骤的细粒度性能拆解。 - 基于事务计时数据，可在不同压力负载等级下进行横向性能对比分析。 - 事务响应时间的异常波动可有效辅助性能瓶颈的定位诊断。

在实际应用中，事务响应时间直接反映业务操作的端到端响应性能，是评估系统用户体验质量的关键指标。

2.2.5 集合点（Rendezvous Point）

在负载测试场景中，为了模拟极端高峰流量对系统的冲击，需要实现多个虚拟用户在精确同一时刻执行特定操作，形成瞬间的并发尖峰。集合点机制正是为实现这种同步并发而设计。

通过在 VU 脚本中插入集合点，可配置指定数量的虚拟用户在集合点处等待，直到所有参与该集合的 VU 均到达后，由控制器统一释放，实现真正意义上的同步执行。例如，在验证系统是否支持 1000 用户同时提交数据的场景中，可在提交操作前设置集合点，PR 将确保当且仅当 1000 个 VU 全部到达集合点后，才同时触发提交操作，从而精确验证系统的并发处理能力。

技术说明： 集合点仅可在 Action 脚本段落中添加，Init 和 Uninit 段落不支持集合点机制。

2.2.6 检查点（Checkpoint）

性能测试不仅要验证系统在高负载下的性能表现，还需确保业务执行结果的正确性。检查点机制用于验证被测系统在特定操作后返回的数据或状态是否与预期一致。

PR 采用 `check("objectName", "property", "expectedValue")`的标准语法定义检查点，系统将在运行时自动比对对象属性的实际值与期望值。检查点可应用于数据库状态验证、GUI 属性校验、响应内容匹配等多个维度，支持正则表达式等高级匹配模式。

2.2.7 参数化与数据驱动（Parameterization & Data-Driven Testing）

录制生成的原始脚本中，所有输入数据均为固定常量。为实现同一脚本的多组数据复用，需要对脚本进行参数化处理，将硬编码的常量替换为来自外部数据源的动态变量，实现测试数据与测试逻辑的解耦。

采用参数化机制的脚本即进入数据驱动模式。PR 内置数据驱动框架，支持 Excel、CSV 等标准格式数据源，测试人员可通过简单的配置实现复杂的数据驱动场景，显著提升测试覆盖率和数据组合验证能力。

2.3 业务支撑能力

2.3.1 产品适用性说明

PerformanceRunner 是泽众软件测试产品矩阵的核心组件之一，其测试脚本与功能自动化测试工具 AutoRunner 保持完全兼容。从脚本语法、内置函数到 API 调用方式，两者遵循统一的技术规范，便于测试资产在性能测试与功能测试之间的无缝复用和迁移。

PR 提供多协议适配能力，可根据被测系统的技术栈特征灵活选择对应的协议模块，满足异构系统环境下的多样化测试需求。

2.3.2 性能测试（Performance Testing）

通过模拟真实生产环境的业务压力负载和使用场景组合，验证系统的性能表现是否满足业务投产要求。性能测试在可控的测试条件下对系统的响应时间、吞吐量、资源利用率等关键性能指标进行量化评估，为系统上线决策提供数据支撑。

2.3.3 负载测试（Load Testing）

在预设的测试环境基线条件下，通过向被测系统逐步递增施加负载压力，持续观测系统性能指标的变化趋势，直至性能指标超出预设阈值或某项系统资源（CPU、内存、连接数等）达到饱和状态。

负载测试的核心目标是发现系统性能的拐点（Breaking Point），确定系统能够支持的最大并发用户数、最大业务处理量等容量上限，为系统扩容规划和容量管理提供量化依据。

2.3.4 压力测试（Stress Testing）

将系统置于资源饱和或过载的极端运行状态下，验证系统在极限压力条件下的会话处理能力、错误恢复机制及整体稳定性。压力测试关注在 CPU、内存、网络带宽等资源处于饱和或超负荷状态时，系统是否仍能保持核心功能可用，是否出现数据不一致、服务崩溃等严重故障。

压力测试的本质是极端条件下的稳定性验证，其核心意义在于通过压力暴露系统的脆弱点，经调优后确保系统即使在异常高峰流量下也不会发生灾难性失效。

2.3.5 配置测试（Configuration Testing）

通过系统性地调整被测系统的软硬件配置参数（硬件规格、网络拓扑、操作系统版本、中间件参数、数据库配置等），对比不同配置组合下的性能表现差异，从而识别影响系统性能的关键配置因子，确定较优的资源分配策略。

配置测试是性能调优的核心方法论。在完成基线性能测试获取参考数据后，逐一调整各项配置参数，将调优后的测试结果与基线数据进行对比分析，判断配置变更是否带来性能增益，持续迭代直至达到系统性能的较优状态。

2.3.6 并发测试（Concurrency Testing）

模拟多用户并发访问同一应用模块、共享数据资源的场景，重点检测并发条件下可能出现的隐藏缺陷，包括内存泄漏、线程死锁、资源竞争、数据不一致等问题。

并发测试的核心目的并非获取性能指标数值，而是发现并发交互引发的逻辑缺陷和稳定性问题，属于稳定性验证的重要组成。

2.3.7 可靠性测试（Reliability Testing）

在系统承载一定的持续业务压力负载的条件下，让应用系统长时间稳定运行（如 7×24 小时持续运行），监测系统在长期运行过程中的性能衰减趋势和资源消耗变化，验证系统是否出现响应时间逐渐延长、资源使用率持续攀升、服务间歇性不可用等稳定性劣化征兆。

可靠性测试与压力测试存在本质区别：压力测试聚焦极端资源极限场景下的错误行为，可靠性测试关注在常规业务压力下长时间运行的稳定性表现。

2.3.8 核心特性概述

特性维度	详细说明
Java 标准语法脚本	PR 采用业界主流的 Java/Groovy 标准语法作为脚本语言，语法简洁直观，大幅降低测试人员的学习成本。测试人员在使用 PR 的过程中，可同步掌握 Java 编程基础，实现技能增值。
智能录制引擎	支持单次录制完成协议数据、业务报文的自动捕获和脚本生成，录制产物通常可直接执行，无需大量手动修改，显著降低脚本开发工作量，特别适用于频繁迭代的敏捷测试场景。
自动关联技术	内置智能关联引擎，可自动识别并处理

特性维度	详细说明
	B/S 系统中 Session、Token 等动态会话标识的管理，无需测试人员手动编写关联脚本，实现 Session 生命周期的自动维护。
简洁脚本模型	所有测试脚本均继承自标准基类 `TestCase`，使用其提供的标准化方法即可满足绝大多数测试场景，无需深入掌握复杂的 Java 高级特性，便于其他测试工具用户平滑迁移。
数据驱动框架	内置标准数据驱动框架，支持 Excel 格式数据源的直接绑定。录制完成后，通过简单的参数配置即可实现数据驱动，无需测试人员自行开发数据读取逻辑。
良好扩展性	基于标准 Java 语言（JDK 1.5+）构建，用户可自由引入外部 Jar 包扩展脚本功能。凡是 Java 生态提供的功能能力，均可无缝集成到 PR 测试脚本中，满足特殊场景的定制化需求。
标准化兼容	严格遵循国际测试工具标准规范，完整支持同步点、验证点、错误报告等核心机制，便于用户理解使用，并支持与其他商业测试工具进行功能对标。
国产系统支持	全面适配国产操作系统生态，支持在中标麒麟、银河麒麟等国产操作系统上部署运行，满足信创环境下的性能测试需

特性维度	详细说明
	求。

2.4 设计目标与价值主张

目标一：系统投产前的性能质量门禁

在系统正式上线投产前，特别是在系统测试（ST）阶段执行全面的性能测试具有重大的质量保障意义。通过前置的性能验证，可以有效识别和消除潜在的性能瓶颈，规避系统上线后可能出现的性能故障，保障生产环境的安全稳定运行。

PerformanceRunner 致力于帮助客户较大程度地覆盖系统测试后期的性能验证场景，通过系统化的性能评估降低系统上线风险，为业务投产决策提供可靠的技术依据。

目标二：日常运维性能基线保障

系统的性能表现并非静态不变，而是随着数据量增长、环境变更、网络条件波动及配置参数调整而动态变化。建立定期的性能回归测试机制（如月度或季度执行），对于保障生产系统的长期稳定运行至关重要。

特别是在系统参数微调、补丁升级、小版本迭代后，执行快速的性能回归测试可有效验证变更对系统性能的影响。PR 为企业建立常态化的性能监控体系提供技术支撑，助力客户实现主动式性能管理。

目标三：一致性与可重复性保障

性能测试由自动化引擎执行，每次测试运行的脚本逻辑、数据输入、执行环境完全一致，确保了测试结果的高度可重复性和可比性。人工测试难以避免的操作差异和主观因素在自动化测试中被彻底消除。

基于自动化测试的一致性保障，任何被测系统的性能衰减或异常波动均可被精确捕捉和量化，为性能回归分析和趋势预测提供可靠的数据基础。

3. 系统体系结构与技术要求

3.1 运行环境要求

3.1.1 操作系统支持

PerformanceRunner 支持在以下操作系统平台上部署运行：

操作系统类别	具体版本
Windows 桌面系列	Windows XP、Windows 7/8/10/11 系列
Windows Server 系列	Windows Server 2000 及更高版本
国产操作系统	中标麒麟、银河麒麟等国产化操作系统

技术说明： 理论上，凡安装有 JDK 1.6 及以上版本 Java 运行环境的操作系统，PR 均可提供运行支持。

3.1.2 软件依赖要求

依赖组件	版本要求	说明
JDK（Java Development Kit）	1.6 及以上	PR 运行所需的 Java 运行时环境
Internet Explorer	5.5 及以上	IE 浏览器插件功能所需

3.2 性能指标

PerformanceRunner 针对系统性能测试自动化场景进行了深度优化，单机测试引擎的理论并发处理能力可达 **2000 次请求/秒**（具体数值受测试脚本复杂度、协议类型及网络环境影响）。

系统的实际并发上限与硬件配置（CPU 核心数、内存容量、网络带宽）及被测系统响应速度密切相关。在常规测试场景中，更高配置的测试主机可支撑更大规模的并发负载，用户可根据测试目标进行硬件资源的弹性调配。

3.3 扩展能力设计

3.3.1 验证点扩展机制

验证点是用于确认被测系统返回数据或状态与预期是否一致的关键检查机制。PR 内置完整的验证点功能体系，支持字符串精确匹配、Bitmap 图像比对、正则表达式规则验证等标准验证方式。

考虑到实际业务场景的验证需求往往极为复杂（例如：执行数据插入操作后，需通过 JDBC 连接数据库验证记录是否正确写入），PR 提供了开放的验证点扩展接口。用户可根据具体的被测系统特征，自定义编写验证类或验证方法，通过继承和调用 PR 提供的基础验证 API，将自定义验证逻辑的结果无缝集成到标准测试报告中，实现验证点功能的无限扩展。

3.3.2 第三方测试管理工具集成

PR 提供标准化的外部集成接口，支持通过数据文件或数据库表与第三方测试管理工具进行测试用例信息、测试数据的交换。同时，PR 提供完整的命令行调用接口（CLI），支持远程启动和调度执行，为第三方测试管理平台提供标准化的执行调用入口，实现测试执行与测试管理的流程贯通。

3.3.3 第三方缺陷跟踪工具集成

PR 内置缺陷跟踪工具的 API 调用能力，可与主流缺陷管理系统实现”无缝连接”，支持在测试执行过程中自动提交缺陷报告，将测试失败信息直接推送至缺陷跟踪系统，实现测试与缺陷管理的闭环集成。

3.4 可靠性与可用性保障

系统的可用性和可靠性通过以下多维指标进行综合衡量：

1. **出错处理与恢复能力：** 系统在运行过程中遭遇错误时，是否具备完善的异常处理机制，能够跳过当前失败点，继续执行后续可运行的测试步骤，避免因单点故障导致整个测试批次中断。

2. **运行稳定性：** 测试工具本身在长时间高负载运行过程中不出现进程崩溃、内存泄漏、无响应等异常状况，确保测试任务的完整执行。

3. **脚本调试支持：** 当测试脚本出现异常时，提供强大的诊断调试功能，支持断点设置、单步执行、变量追踪、表达式求值等标准调试能力，帮助测试人员快速定位问题根因。

4. **版本向下兼容：** 产品升级迭代后，历史版本的测试脚本保持兼容，无需重写即可在新版本中继续使用，保障测试资产的长效价值。

PR 可靠性保障具体措施

系统的出错处理机制： 在压力测试执行过程中，大量 VU 以高频次循环执行测试脚本。当某个 VU 的脚本执行失败时，该失败被隔离处理，不影响其他 VU 的继续执行。失败的详细信息被记录到执行日志中，最终在测试报告中统一汇总分析，确保测试任务的整体完成度。

产品升级兼容策略： PR 承诺版本间的向下兼容保障，新增功能通过扩展方式提供，不替换或废弃原有的内置方法。同时，由于测试脚本采用标准 Java 语法，天然继承了 Java 语言的向后兼容特性，历史脚本在新 JDK 环境下仍可正常运行。

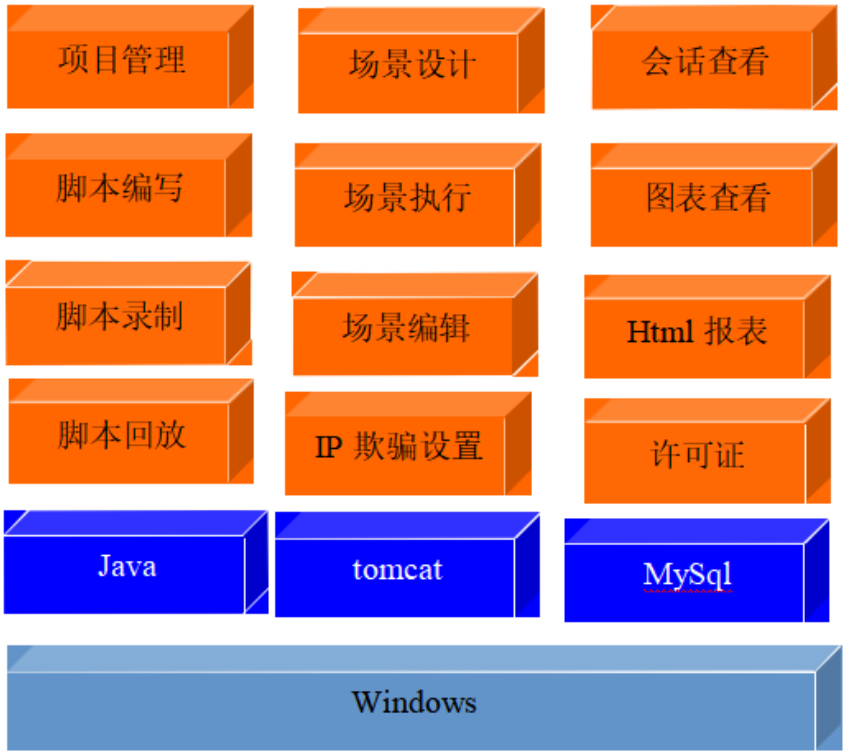
3.5 国际化与本地化支持

多语言编码支持： 全面支持 Unicode 国际编码标准，可正确处理中文、英文、日文、韩文等多国语言文字，满足全球化应用的测试需求。

界面语言切换： 提供中英文双语界面版本，用户可根据使用习惯自由切换。

自动编码识别： 系统内置智能编码识别引擎，自动检测并适配不同字符编码格式，用户无需手动配置编码参数。

4. 系统架构



PerformanceRunner 采用模块化分层架构设计，由三大核心子系统构成：**生成器（Generator）**、**执行器（Executor）**和**分析器（Analyzer）**。

模块名称	**核心职责**	**功能范围**
生成器	测试资产开发	项目管理、脚本编写与编辑、操作录制与回放、事务与集合点插入、对象库管理等
执行器	测试场景执行	场景创建与编辑、场景执行计划配置、VU 并发调度、IP 欺骗设置、场景运行控制等

模块名称	核心职责	功能范围
分析器	测试结果分析	虚拟用户状态图、事务响应时间图、点击量统计、吞吐量分析、CPU/内存/网络资源监控图表等

三大模块协同工作，形成从测试脚本开发到场景执行再到结果分析的完整性能测试闭环。测试人员在生成器中完成脚本开发和调试后，在执行器中配置并发场景并启动负载测试，最终通过分析器对采集的性能数据进行深度分析，生成全面的性能评估报告。

4.1 性能测试引擎关键技术优化

PerformanceRunner V4.0 在底层引擎层面进行了三项关键技术升级，从日志基础设施、脚本执行引擎到 HTTP 通信层实现全面重构，确保性能测试工具本身不会成为被测系统性能评估的干扰变量。

4.1.1 日志框架升级：Log4j 转 Log4j2

维度	优化前（**Log4j 1.x**）	优化后（**Log4j2**）
架构模式	单线程同步处理	基于 LMAX Disruptor 的高性能异步架构
吞吐量	受限于同步 I/O 瓶颈	异步模式下吞吐量提升 **10 倍以上**
延迟表现	高并发下日志写入阻塞业务线程	无锁化设计，日志写入延迟降低至微秒级
安全与维护	已停止维护，存在已知安全漏洞	活跃维护，内置 JNDI 防护等安全机制

设计目标：消除日志系统在压测高并发场景下的性能瓶颈，解决历史版本的安全隐患，构建低延迟、高吞吐、可审计的日志基础设施。

技术优势：Log4j2 的异步日志器通过无锁环形缓冲区实现业务线程与 I/O 线程的彻底解耦，在 PR 性能测试工具的并发执行场景中，可显著降低因日志刷盘导致的线程阻塞，提升整体压测吞吐量的同时保障日志数据的完整性与实时性。

4.1.2 脚本执行引擎重构：BeanShell 解释执行 转 Groovy 编译执行 + 缓存机制

维度	优化前（BeanShell）	优化后（GroovyClassLoader）
执行模式	逐行解析、逐条解释执行	预编译为 JVM Bytecode，反射调用
编译开销	每次执行均触发完整解析流程	首次编译后缓存 Class 对象，后续直接复用
内存特性	高频创建临时解析对象，GC 压力大	编译产物常驻元空间，减少堆内存分配
执行性能	解释执行效率低，脚本成为瓶颈	接近原生 Java 的执行效率

设计目标：将脚本执行引擎从解释型架构升级为编译型架构，通过编译产物缓存策略消除重复解析的开销，从根本上降低压测过程中的 GC 频率与停顿时间，确保脚本逻辑不会成为性能测试的干扰因子。

技术优势：

1. **性能飞跃：**GroovyClassLoader 将脚本源码编译为标准 JVM Class 文件，执行路径与原生 Java 一致，较 BeanShell 的解释执行模式实现数量级性能提升。

2. **资源集约**：编译结果通过 Class 级缓存持久化，避免每次执行重复创建解析树、词法分析器等临时对象，显著降低 Young GC 触发频率与 Full GC 风险。

3. **稳定性增强**：消除 BeanShell 动态解释过程中的内存泄漏隐患，提升长时间压测任务的执行稳定性。

4.1.3 HTTP 通信层优化：URLConnection 转 OkHttp 连接池化架构

维度	**优化前（HttpURLConnection）**	**优化后（OkHttp + 单例 Client）**
连接管理	每次请求新建连接对象，TCP 握手开销大	全局单例 Client，连接池复用 TCP 连接
对象创建	高频创建/销毁 URLConnection 实例	复用 OkHttpClient 实例，减少对象分配
GC 影响	短生命周期对象堆积，触发频繁 Young GC	长生命周期对象池化，GC 压力显著下降
协议支持	HTTP/1.1 仅	原生支持 HTTP/2、GZIP、响应缓存等现代特性

设计目标：重构 HTTP 通信层的资源生命周期管理，通过连接池化与客户端实例复用策略，消除高频 HTTP 请求场景下的对象创建开销，降低 GC 对压测结果准确性的干扰。

技术优势：

1. **资源复用**：全局单例 OkHttpClient 配合底层连接池机制，实现 TCP 连接的长期复用，避免每次 http_get 指令执行时的连接建立与拆除开销，降低网络延迟。

2. **GC 优化**：将短生命周期的 HttpURLConnection 对象替换为长生命周期、线程安全的单例 Client，大幅减少堆内存中的临时对象数量，降低 GC 触发频率与停顿时间，提升压测数据的准确性与一致性。

3. **扩展性提升**：OkHttp 框架原生支持拦截器链、超时精细化控制、HTTP/2 多路复用等特性，为后续扩展 HTTPS、请求重试、链路追踪等高级能力提供统一的技术底座。

4.1.4 优化收益汇总

优化模块	核心收益
日志框架	消除日志 I/O 瓶颈，提升并发吞吐量，修复安全漏洞
脚本引擎	编译执行替代解释执行，消除重复编译开销，降低 GC 频率
HTTP 通信	连接池化复用，减少对象创建，提升网络请求性能与压测数据准确性

以上三项优化从技术架构层面实现了 PR 性能测试引擎在**执行效率、资源消耗、稳定性**三个维度的全面提升，确保性能测试工具本身不会成为被测系统性能评估的干扰变量。

---

5. 系统基本功能

5.1 测试项目管理

PR 以项目（Project）作为测试资产的管理单元。用户通过生成器的”新建”菜单创建测试项目，项目创建后自动生成三个核心脚本文件：

脚本文件	执行时机	用途说明
`Init.bsh`	场景开始时执行一次	初始化操作，如建立数据

脚本文件	执行时机	用途说明
		库连接、登录系统、准备测试数据等
`Action.bsh`	场景运行期间循环执行	核心业务操作脚本，承载主要的负载压力
`Uninit.bsh`	场景结束时执行一次	清理操作，如关闭连接、登出系统、释放资源等

用户可通过项目管理器查看项目结构，双击脚本文件即可进入集成编辑环境进行脚本开发。

5.2 测试脚本创建与录制

5.2.1 手动创建测试脚本

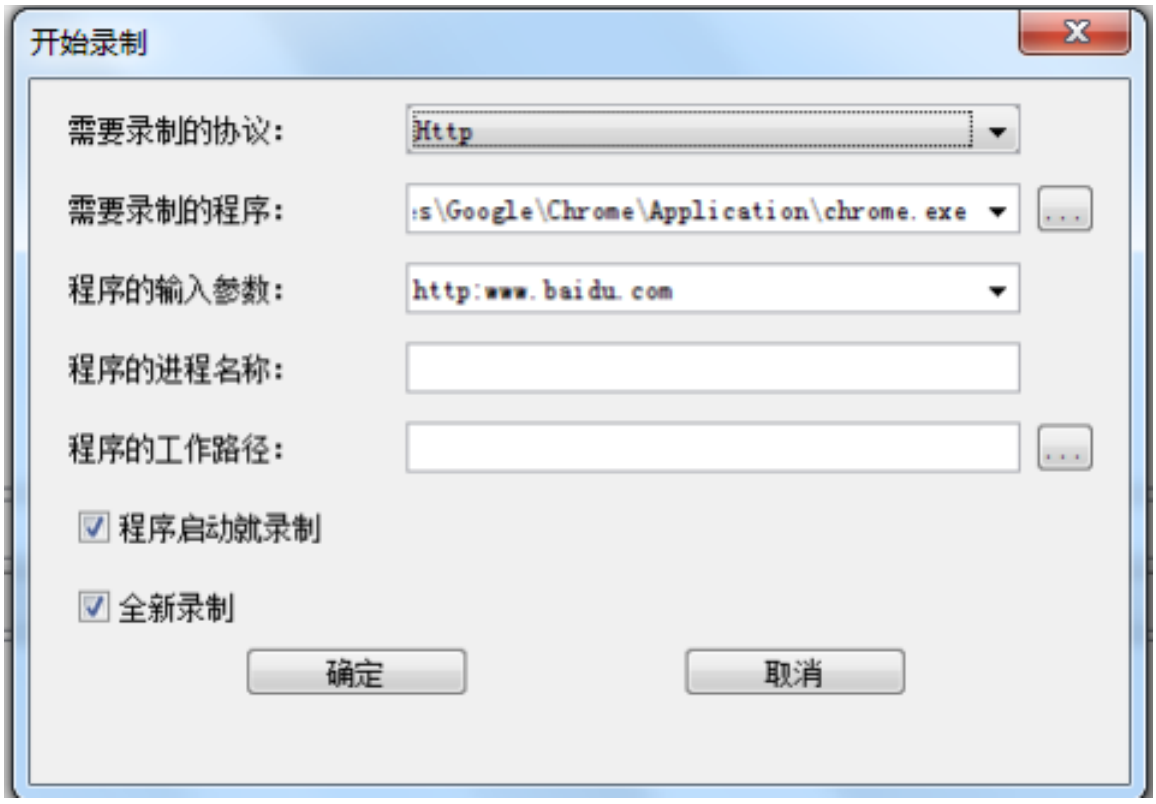
用户可创建空的测试脚本文件（如 `test.bsh`），创建后的脚本需通过录制或手工编写填充业务逻辑。空脚本为后续录制操作提供容器。

5.2.2 自动录制测试脚本

PR 提供业界领先的协议录制能力。用户从菜单或工具栏启动”录制”命令后，系统自动捕获用户在被测应用上的所有操作，并实时将录制的操作转换为可执行的测试脚本代码，同步显示在脚本编辑器中。

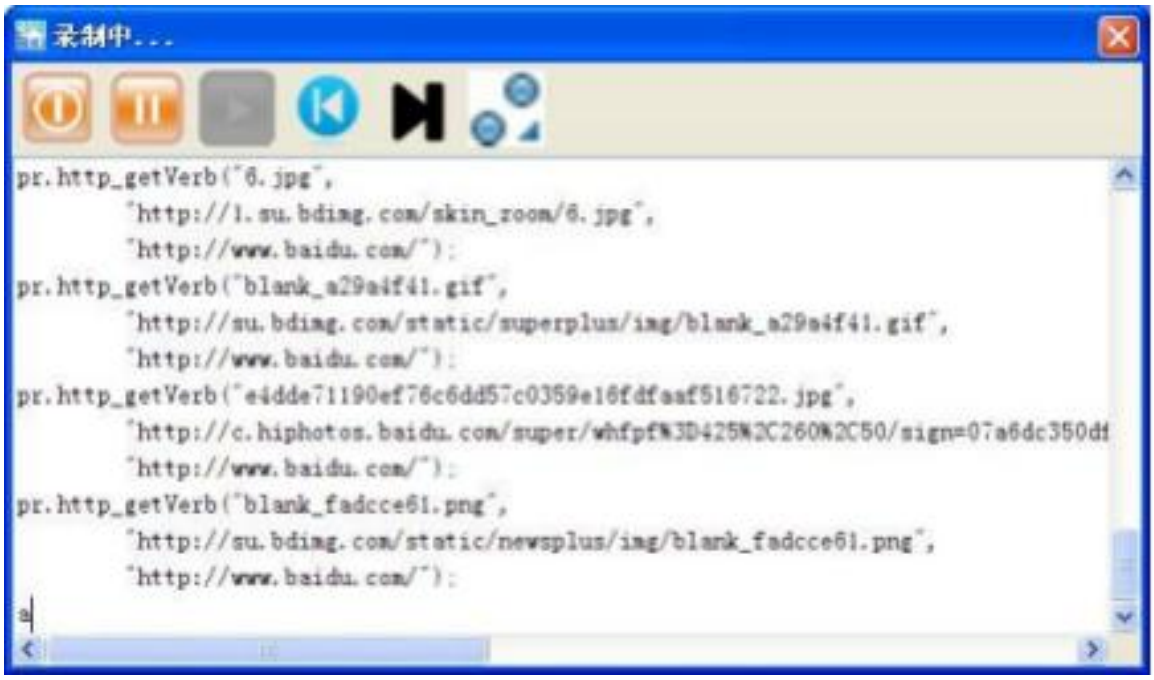
录制过程涵盖完整的协议数据交互——包括客户端发送的请求数据和接收的服务器响应数据。PR 支持两种录制启动方式：

方式一：程序级录制 - 指定录制协议类型（如 HTTP） - 指定被录制程序路径（如 IE 浏览器可执行文件） - 配置程序启动参数（如目标 URL） - 选择录制模式（全新录制/追加录制）



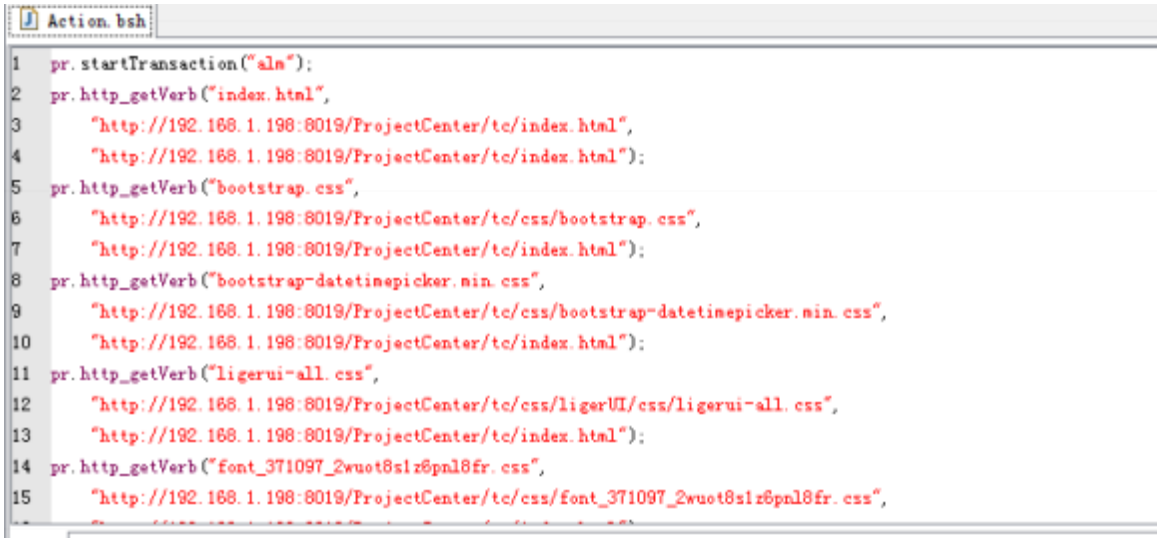
方式二：URL 级录制 - 直接输入目标网址，PR 自动启动内置浏览器或代理捕获方式进行录制，简化配置步骤。

录制过程中，PR 提供悬浮控制面板，支持实时暂停录制、停止录制、插入事务边界、插入集合点等快捷操作。录制结束后，自动返回脚本编辑界面，生成的脚本可直接回放验证。



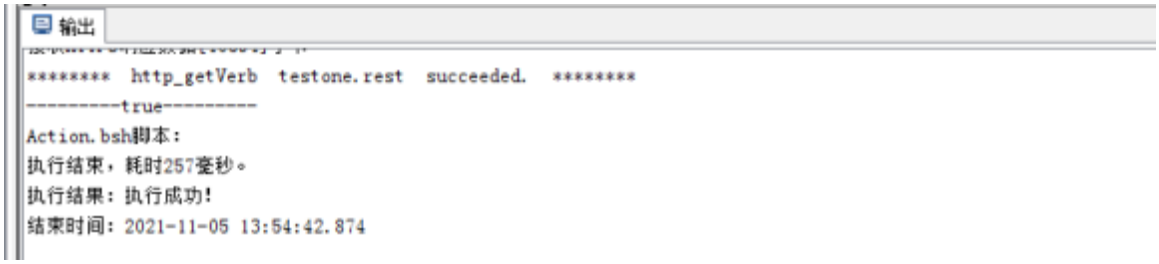
5.3 录制结果回放与验证

录制完成后，PR 自动生成对应的录制脚本。



为验证脚本的正确性和可执行性，需对脚本进行回放（Playback）操作。回放通过生成器工具栏的回放按钮触发执行。

回放结束后，生成器的”输出”窗口（Output Console）将显示完整的脚本执行日志，包括：- 每条脚本语句的执行状态（成功/失败） - 事务执行耗时统计 - 错误信息和堆栈跟踪（如有） - 场景执行总时长汇总

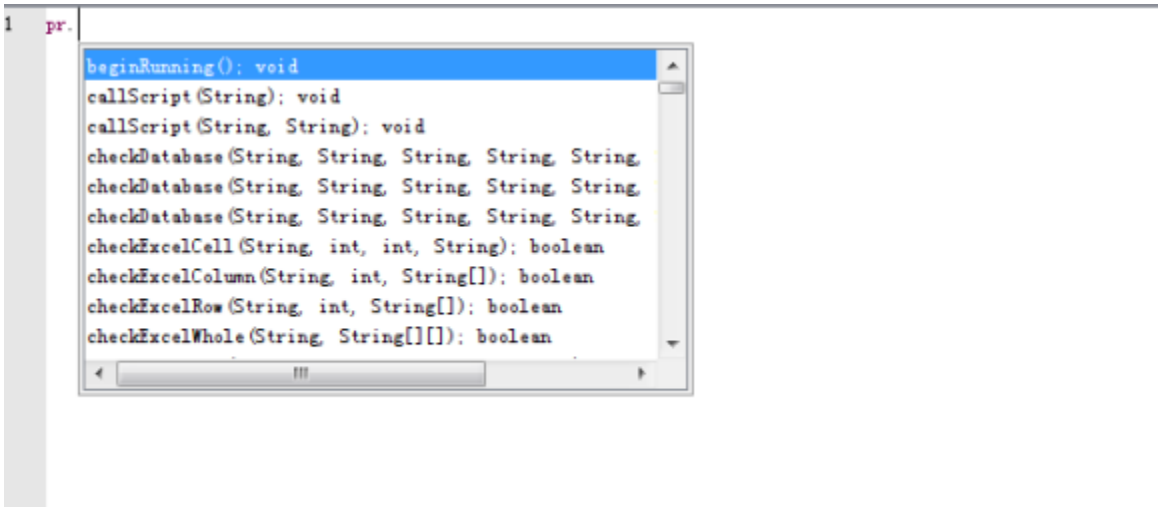


通过分析输出日志，测试人员可快速判断脚本录制质量，识别需要关联处理或参数化调整的动态内容。

5.4 测试脚本编辑环境

5.4.1 脚本结构规范

PR 测试脚本遵循 **Groovy 语法规范**（与 Java 语法高度兼容），采用顺序执行模型，无需显式声明函数或类定义，语法简洁直观。对于无 Java 编程经验的测试人员，通过短期学习即可掌握脚本开发技能。



5.4.2 智能代码编辑器

PR 集成类 Eclipse 风格的智能代码编辑器，提供以下核心能力：

1. **关键字识别与高亮：** 自动识别 Java 语法关键字、PR 内置 API 方法、标准类库引用，以不同颜色和字体样式区分显示，避免基础语法错误。
2. **实时语法分析：** 在脚本编辑过程中进行动态语法检查，对语法错误实时标注并给出错误提示，将编译期错误前置到编码阶段解决，显著提升脚本开发效率。
3. **代码自动补全：** 提供智能代码提示（Code Completion）功能，根据当前输入上下文自动建议可用的方法、变量和类成员，减少记忆负担和输入错误。

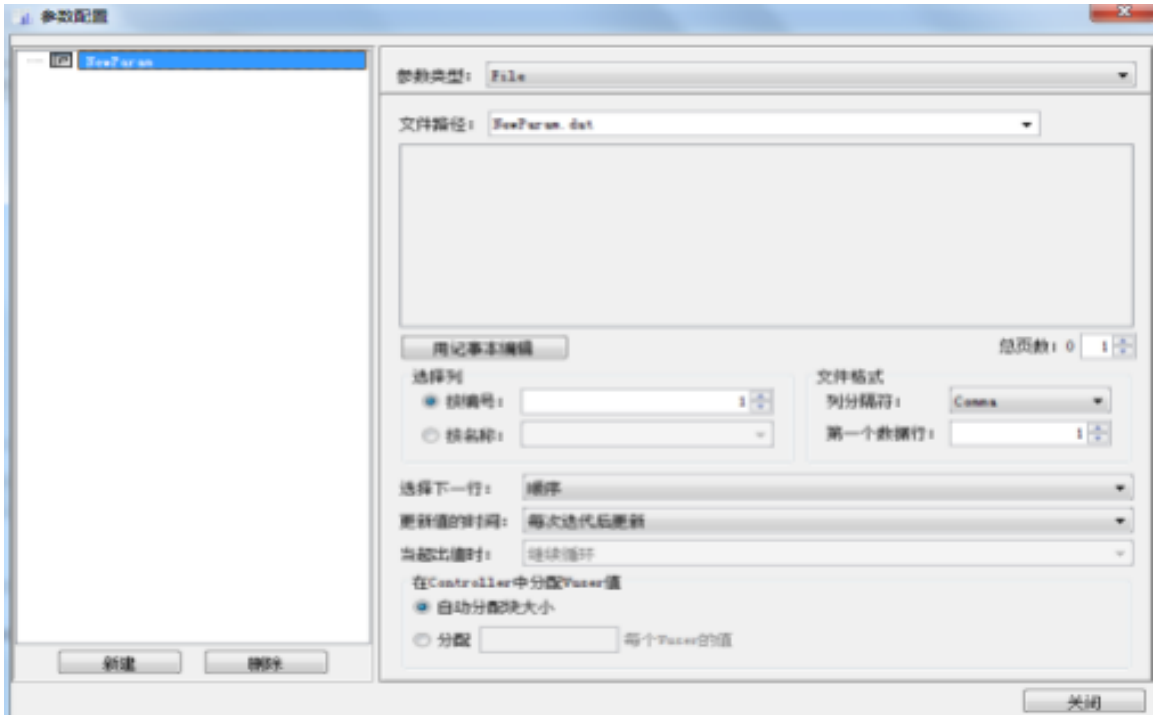
5.5 测试数据参数化

5.5.1 数据驱动概念

固定数据的测试脚本只能验证单一执行路径，无法满足边界值分析、等价类划分、路径覆盖等系统化测试设计方法的要求。数据驱动测试（Data-Driven Testing）通过将输入数据和验证数据从脚本逻辑中解耦，实现”一个脚本、多组数据”的高效测试模式。

PR 完整实现脚本与数据的物理分离：脚本保留业务操作逻辑，执行时从外部数据源动态读取测试数据，实现测试逻辑与测试数据的独立维护和灵活组合。

5.5.2 参数配置体系

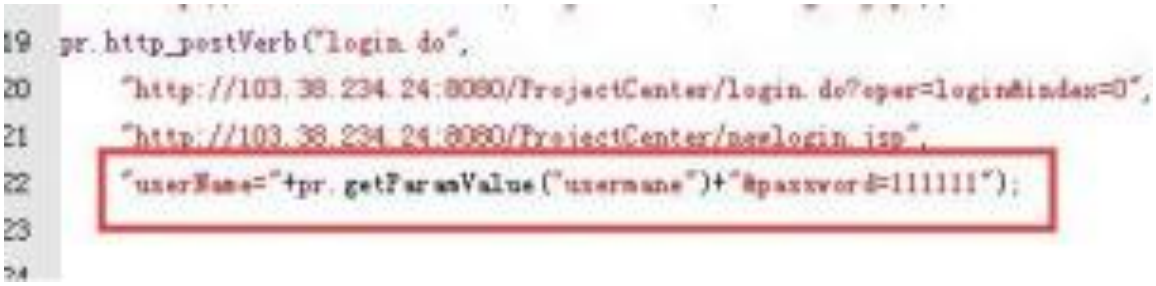


PR 提供可视化的参数配置界面，支持以下配置维度：

配置项	配置说明
参数名称	自定义参数标识符，在脚本中通过该名称引用
参数类型	支持 File（文件）、Date/Time（日期时间）、Number（数字）、Vuser ID（虚拟用户编号）等类型
数据源路径	指定参数数据文件（Excel/CSV）的存储路径
数据编辑	支持可视化表格编辑或记事本批量编辑
列选择策略	按列编号或列名称指定取值列；配置列分隔符；设置数据起始行
取值策略	顺序取值（Sequential）、随机取值（Random）、唯一取值（Unique）、关联取

配置项	配置说明
	值（与指定参数同步）
更新时机	每次迭代更新、每次访问更新、仅初始化时更新一次

参数配置完成后，在脚本中将硬编码值替换为 pr.getParamValue("参数名") 方法调用，即可实现运行时动态数据注入。在场景执行时，VU 将按照配置的取值策略从参数表中读取数据，真实模拟多用户多数据的并发访问场景。



5.6 动态内容关联

5.6.1 关联解决的问题

在 B/S 架构系统的性能测试中，录制完成的脚本直接回放时经常出现执行失败的情况。根本原因在于 HTTP 请求中的 Cookie、Session ID、Token 等动态会话标识以及 HTML 响应体中的动态生成内容（如隐藏表单域、时间戳、随机数等）在每次请求时均会发生变化。录制阶段捕获的静态值在回放时已被服务器判定为失效，导致请求被拒绝。

关联（Correlation）机制正是为解决动态内容问题而设计。PR 的关联引擎通过对比录制会话和回放会话中 Cookie 及响应体的差异，自动识别变化部分并将其参数化。在后续回放时，PR 自动从服务器响应中提取最新的动态值并注入到后续请求中，确保每次请求的会话有效性。

5.6.2 关联执行过程

1. **Cookie 快照分析：** 捕获 HTTP 请求中的 Cookie 内容，识别需要关联的动态会话标识。

2. **响应体快照分析：** 解析 HTML 响应体中的动态生成内容，定位变化数据的位置和格式。
3. **自动参数化：** 将识别出的动态内容替换为变量引用，建立请求间的数据传递链路。

PR 的自动关联功能可一次性扫描脚本中所有网络请求，全面识别动态内容并完成参数化配置，大幅降低手动关联的工作量和技术门槛。

5.7 同步点与验证点

5.7.1 验证点设计原理

测试的根本目的是验证被测系统在特定操作后的实际结果与预期结果的一致性。验证点（Checkpoint）是 PR 中实现结果断言的核心机制。验证点的执行遵循”失败继续”原则——即使验证失败，脚本仍继续执行后续步骤，并将验证结果（通过/失败）详细记录到测试报告中，供事后分析。

5.7.2 验证点添加方法

PR 无法自动推断用户需要验证的业务内容，因此验证点需要测试人员根据测试需求手工插入。PR 提供标准化的验证点 API：`check("objectName", "property", "expectedValue")`，用户可在脚本编辑器中调用该方法，指定待验证的对象、属性及期望值。

验证点可灵活应用于多层次验证：HTTP 响应状态码验证、响应体内容匹配、数据库记录校验、GUI 控件属性断言等。

5.8 测试场景构建

PerformanceRunner 的测试场景（Scenario）由**场景组（Scenario Group）**和**场景计划（Scenario Plan）**两部分构成，通过灵活的配置组合可模拟多种复杂的负载模型。

5.8.1 场景组（Scenario Group）

场景组是由一个或多个测试项目构成的逻辑集合。每个测试项目对应一组 Init、Uninit 和 Action 脚本： - **Init 脚本：** 在场景组开始执行时调用一次，完成测试环境初始化。 - **Uninit 脚本：** 在场景组执行结束时调用一次，完成资源清理释放。 - **Action 脚本：** 在场景运行期间被 VU 循环反复调用，构成实际的负载压力。

一个场景可关联单个或多个测试项目，实现混合业务负载的模拟（如同时模拟用户登录、商品查询、订单提交等多种业务操作的并发组合）。

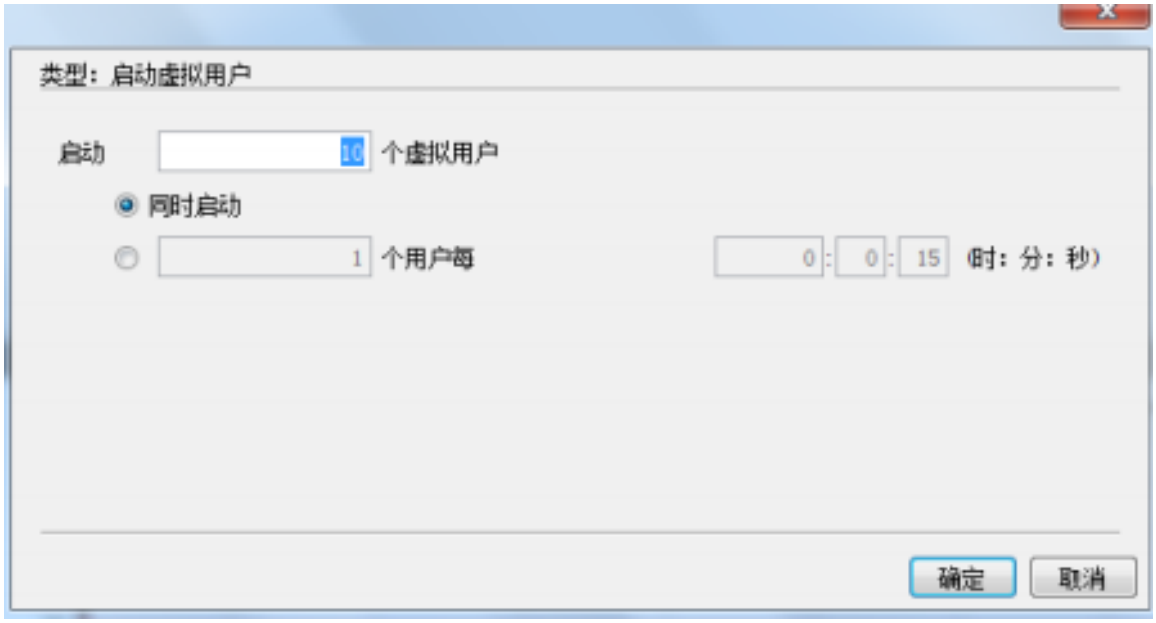
5.8.2 场景计划（Scenario Plan）

场景计划为场景组中的各测试项目提供细粒度的执行策略配置，

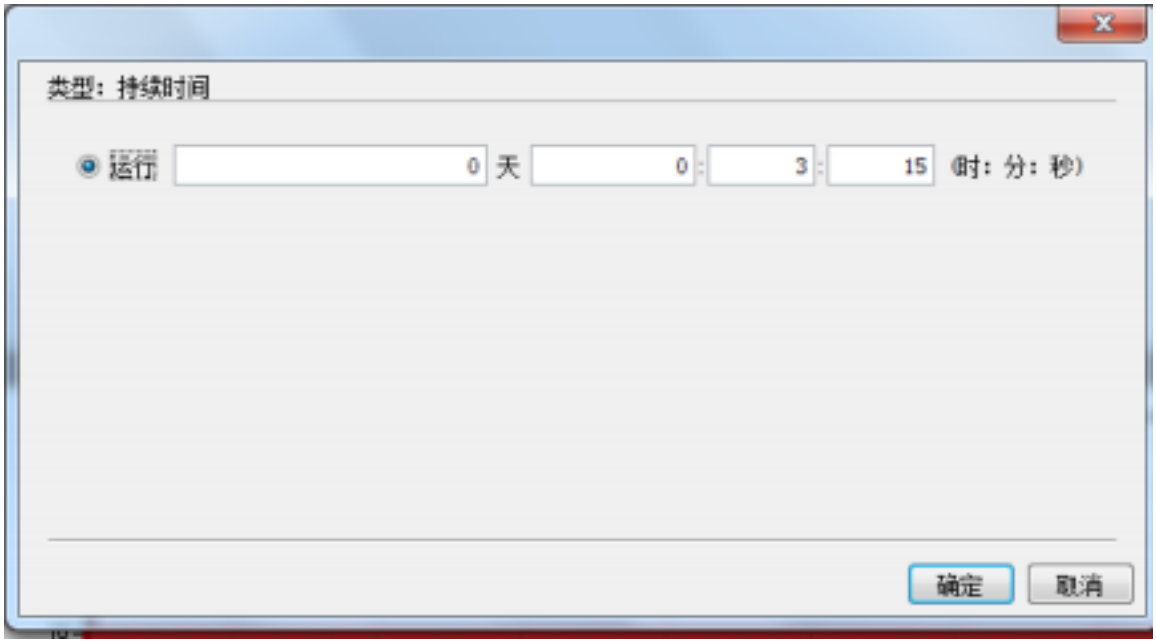
主要包括：

启动组策略： - 场景启动后立即执行 - 场景启动后延迟指定时间再执行 - 在指定前置项目执行完毕后触发执行（支持项目间的依赖编排）

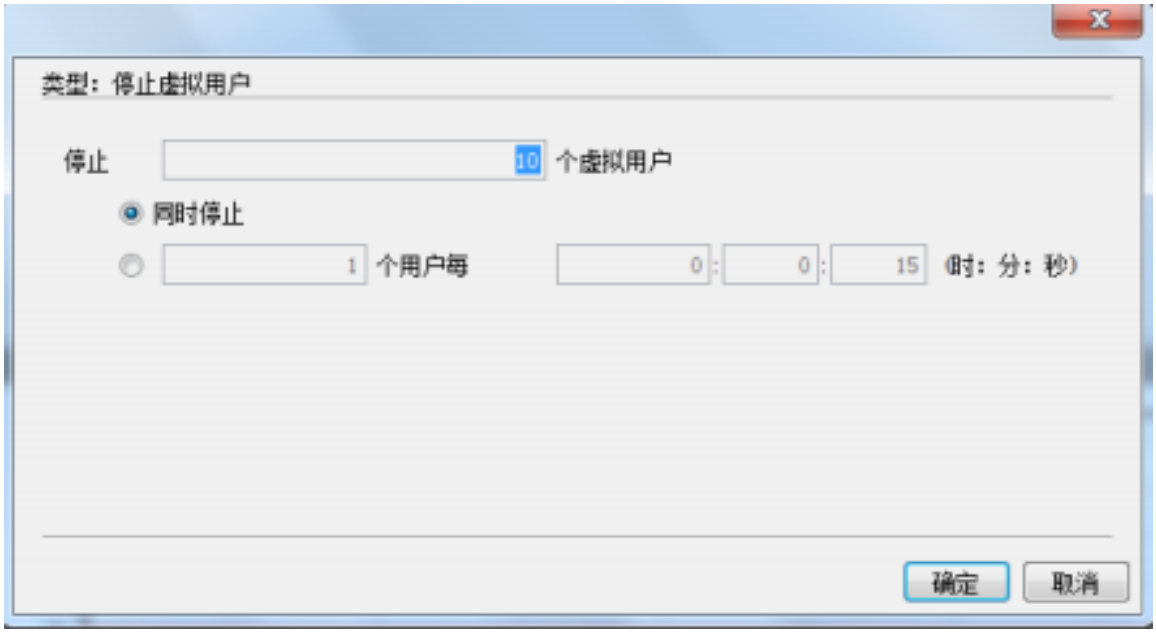
虚拟用户启动策略： - 同时启动：所有 VU 在同一时刻瞬间启动（模拟突发流量） - 梯度启动：在指定时间内分批递增启动指定数量的 VU（模拟流量逐渐增长）



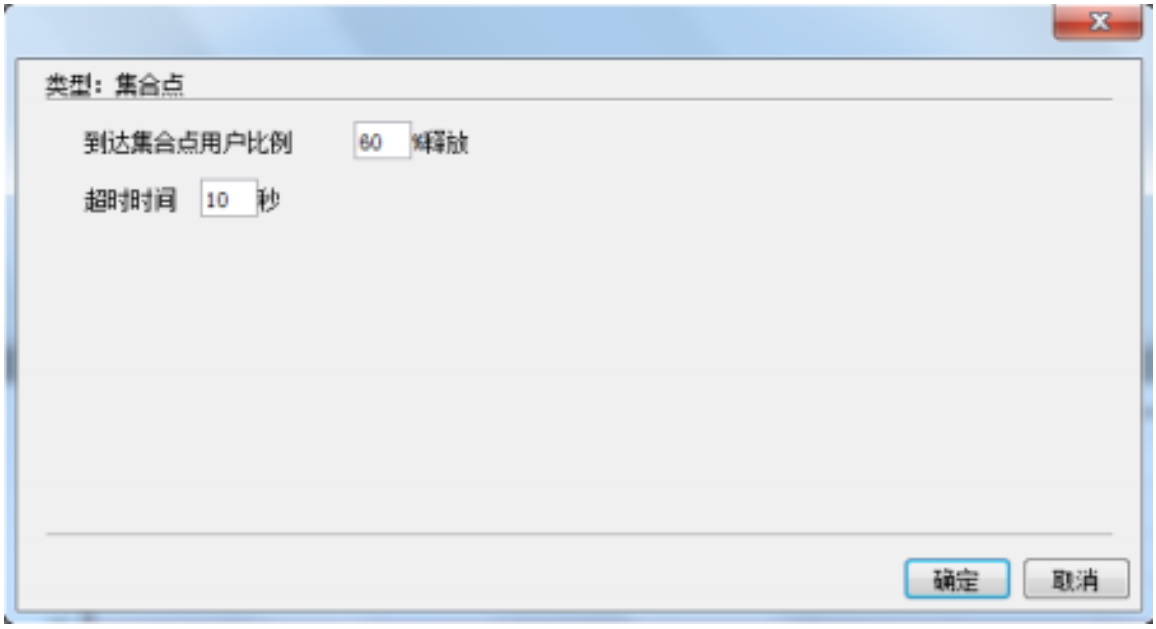
持续时间策略： - 运行直到结束：VU 执行完 Action 脚本后自动退出 - 指定持续时间：VU 在设定的时间窗口内持续循环执行 Action 脚本



虚拟用户停止策略： - 同时停止：所有 VU 在持续时间到达后同时停止 - 梯度停止：在指定时间内分批递减停止 VU 数量（模拟流量平滑下降）

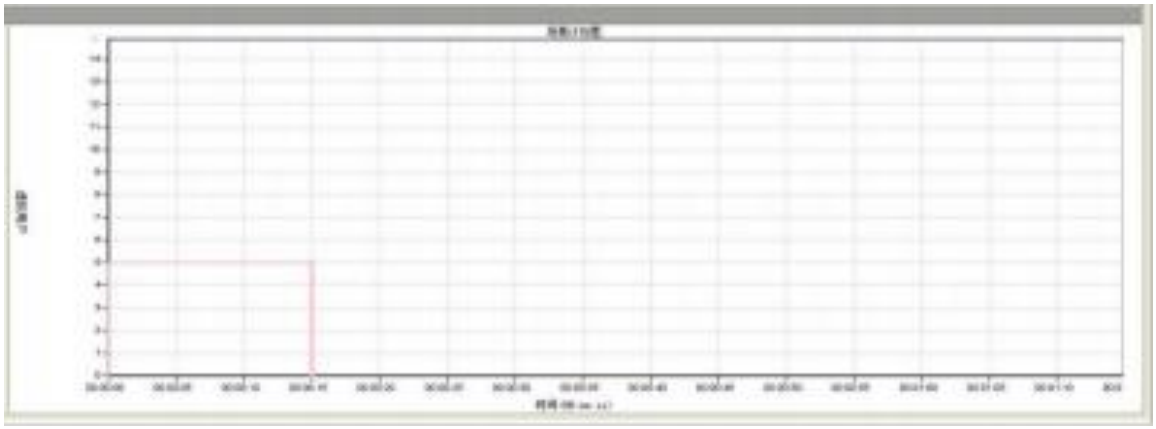


集合点配置： 支持为场景中的关键操作设置集合点，配置参与集合的 VU 比例和超时等待策略。



5.8.3 场景计划可视化

PR 提供直观的场景计划图形化展示界面（Scenario Plan Graph），以时间轴方式呈现 VU 数量随执行时间的变化曲线，帮助测试人员直观审视和调优负载模型的合理性。

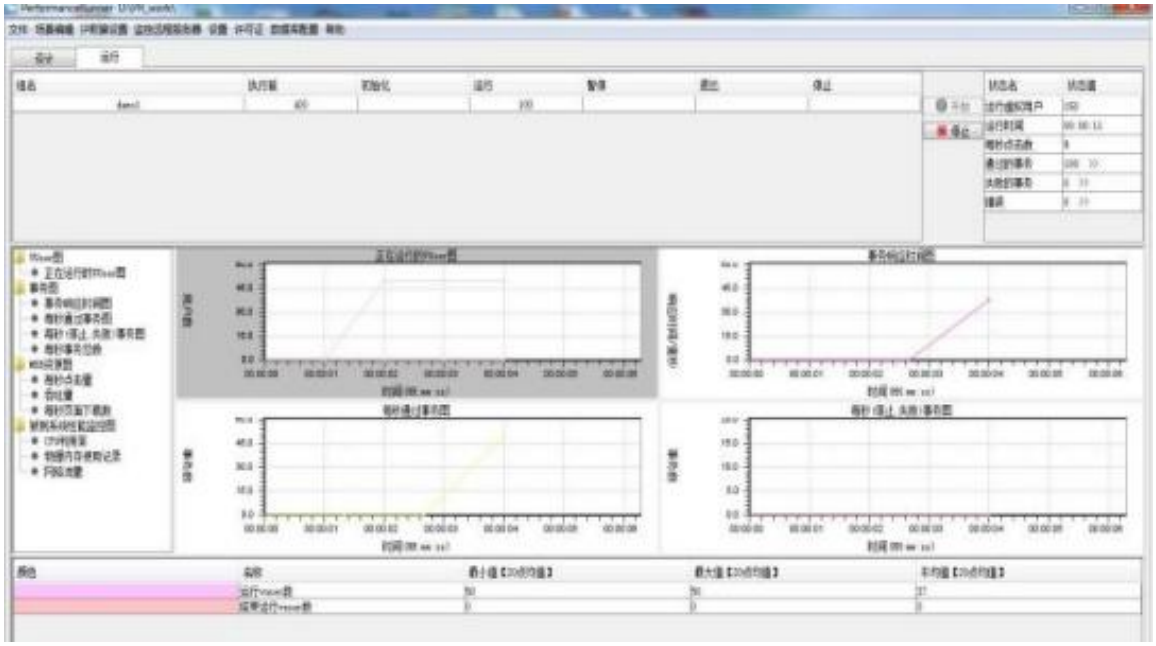


5.9 测试场景执行与监控

脚本回放用于验证单个脚本的正确性，而性能测试的核心目标——验证系统在特定负载下的性能表现——则通过场景执行来实现。

以压力测试为例，测试人员完成脚本录制后，按需求在关键请求前插入事务边界和集合点，然后在执行器中构建测试场景，根据性能模型配置 VU 数量、加压策略和持续时间，保存并启动场景执行。

PR 在执行过程中提供丰富的实时监控图表，帮助测试人员动态观察系统性能表现：



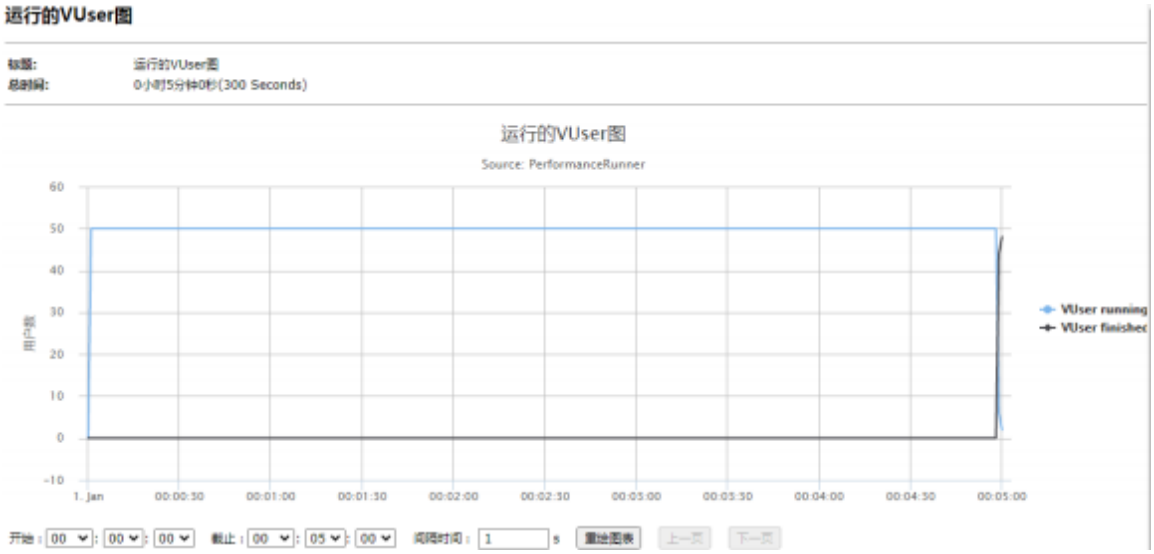
监控图表	监控内容
运行中 VU 图	当前时刻并发执行的 VU 数量
事务响应时间图	各事务的实时响应时间变化趋势
每秒通过事务数图	各事务每秒钟的成功处理次数
每秒失败事务数图	各事务每秒钟的失败次数
每秒事务总数图	综合统计每秒事务处理总量（含成功与失败）
每秒点击量图	系统每秒接收的请求总数
吞吐量图	系统每秒处理的数据流量（字节/兆字节）
CPU 使用率图	被测系统 CPU 资源利用率
物理内存使用图	被测系统内存资源消耗
网络流量图	网络上传/下载流量监控

5.10 性能分析图表

场景执行完成后，PR 自动保存完整的性能统计数据，用户可通过场景名称访问该场景的历史执行结果。分析图表类型与实时监控图表保持一致，呈现的是整个执行周期的汇总统计视图。

5.10.1 运行 VU 统计图

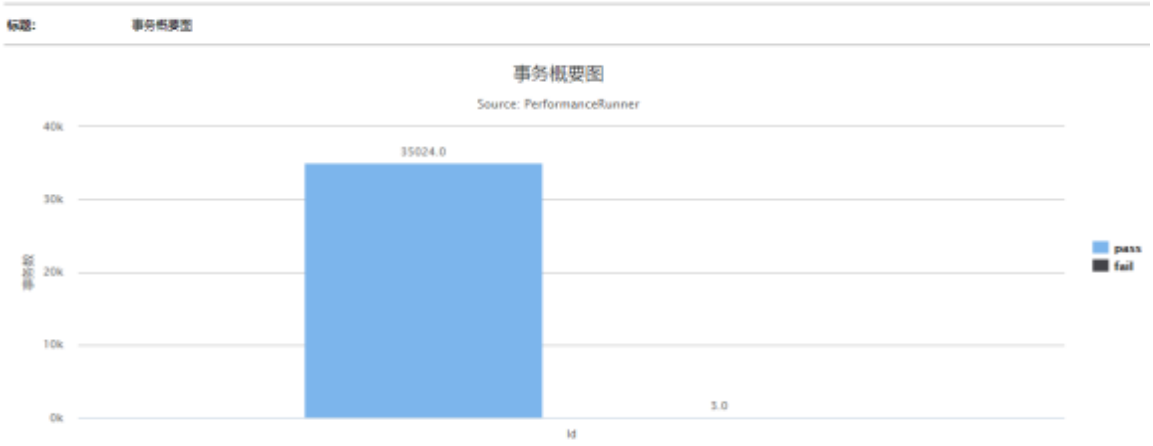
以时间序列方式展示每个采样时刻同时运行的 VU 数量，直观反映测试过程中的并发规模变化。统计表格提供最小并发数、最大并发数、平均并发数、中位数等统计指标。



5.10.2 事务概要图

以柱状图形式展示各事务的执行结果汇总，横轴为事务名称，纵轴为执行次数。绿色柱体表示成功次数，红色柱体表示失败次数，一目了然地呈现事务成功率分布。

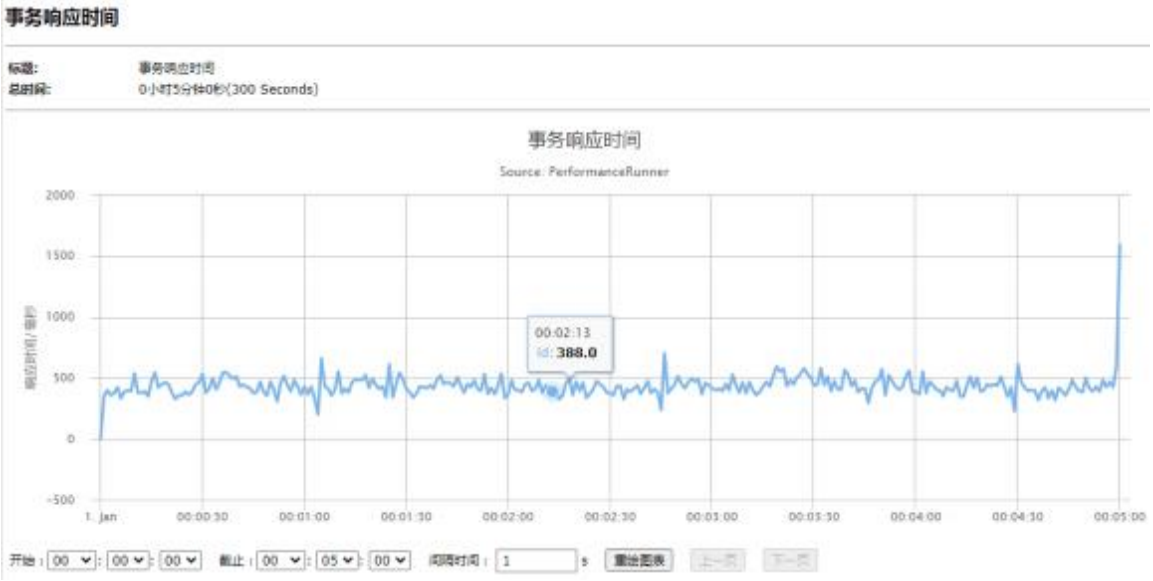
事务概要图



5.10.3 事务响应时间图

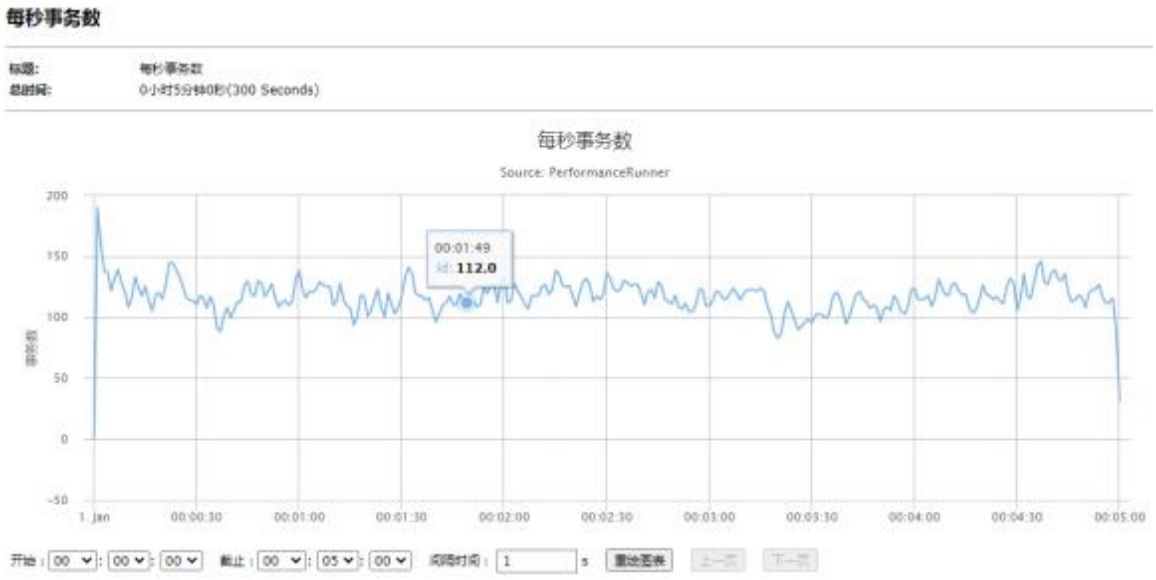
时间序列折线图，展示各事务在每个采样时刻的响应时间。通过观察响应时间曲线的走势，可判断系统性能是否随时间或负载变化而衰减。

事务响应时间



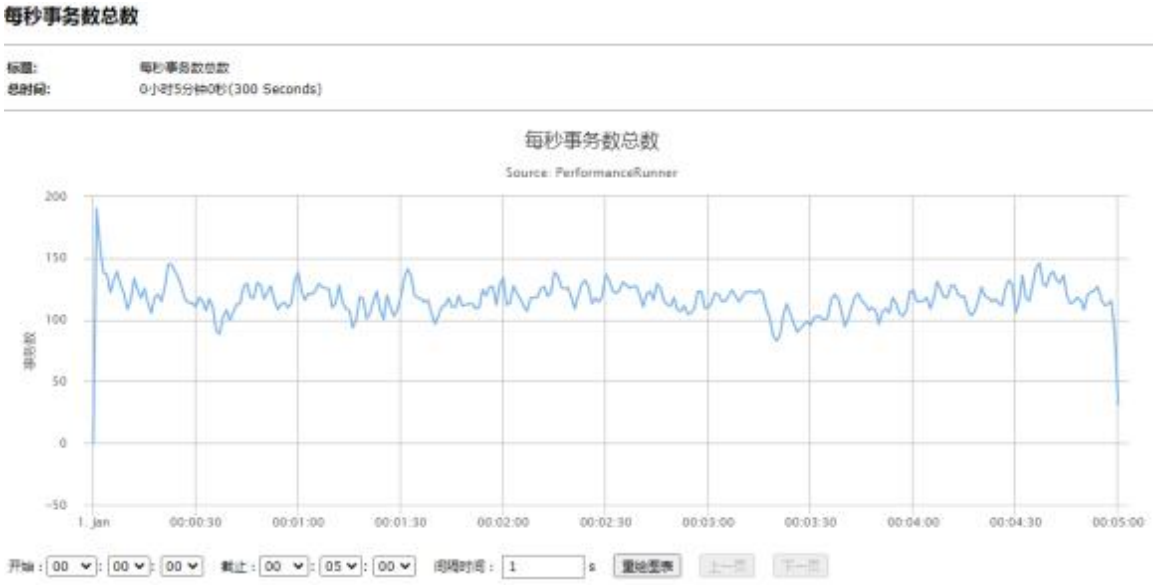
5.10.4 每秒事务数图（TPS）

统计各事务在每个采样时刻的每秒处理量（Throughput），是衡量系统事务处理能力的关键指标。TPS 趋势图可有效反映系统处理效率的变化规律。



5.10.5 每秒事务总数图

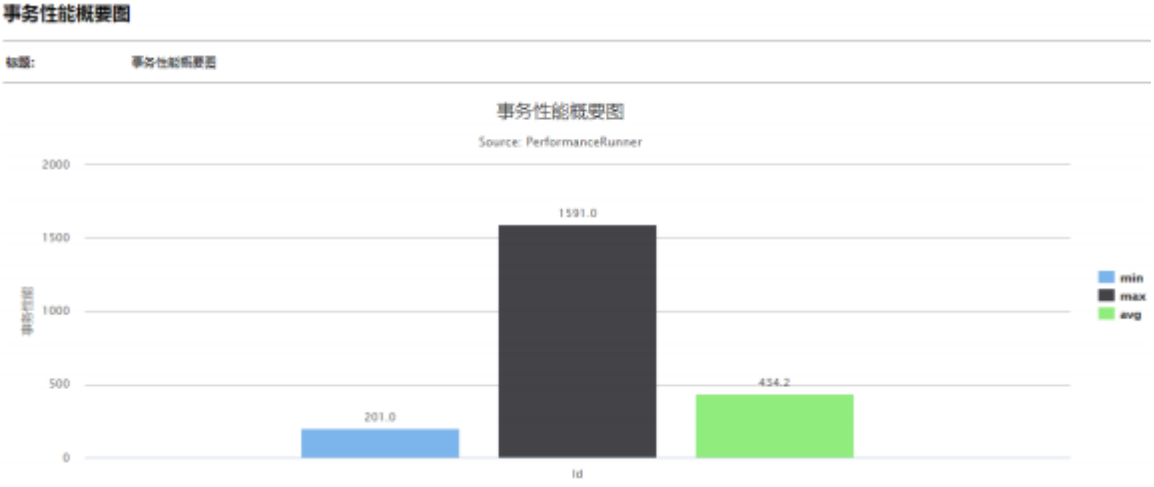
在 TPS 基础上增加错误事务的统计，提供事务处理总量的完整视图，便于快速发现事务失败率的异常增长。



5.10.6 事务性能概要图

分析型图表，展示各事务在整个执行周期内的响应时间统计特征：最小响应时间、最大响应时间、平均响应时间。通过横向对比不同事务的性能指标，识别性能热点。

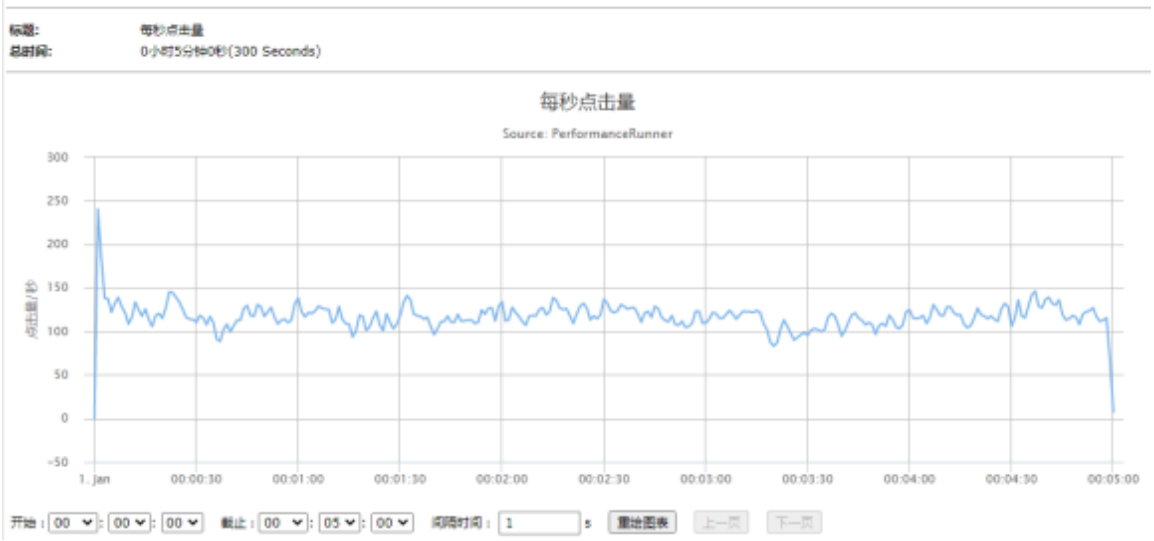
事务性能概要图



5.10.7 每秒点击量（Hits Per Second）

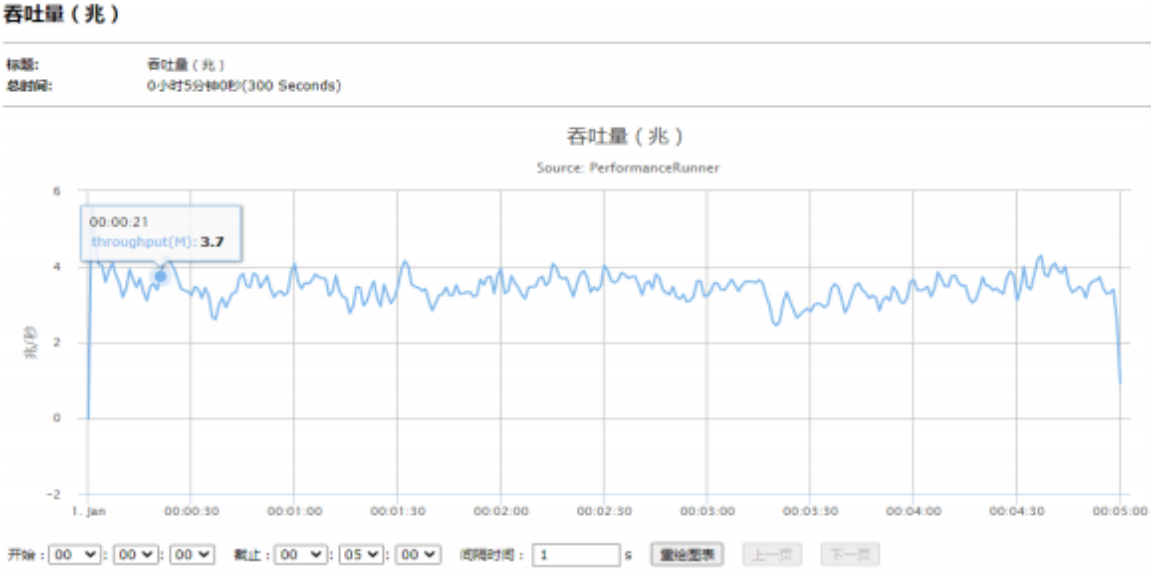
统计各采样时刻系统每秒处理的请求数量，反映系统接收请求负载的强度。

每秒点击量



5.10.8 吞吐量（Throughput）

以字节/秒或兆字节/秒为单位，统计服务器与客户端之间的数据传输速率，综合反映请求和响应的数据流量，是评估系统网络处理能力的重要指标。

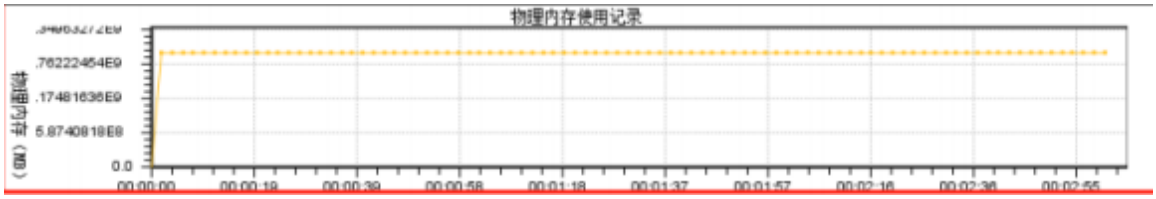


5.10.9 CPU 使用率

监控被测服务器在 PR 施压过程中的 CPU 资源消耗情况，识别 CPU 资源是否成为性能瓶颈。

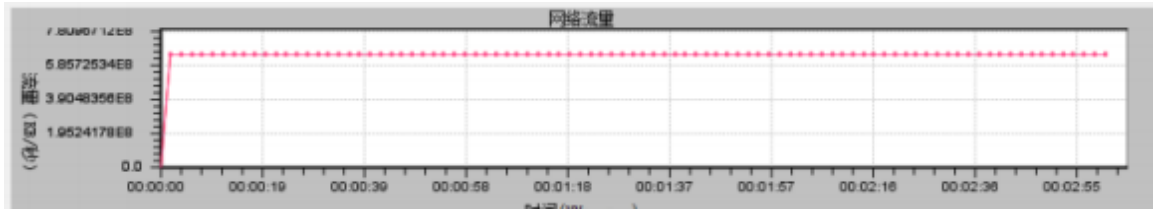
5.10.10 物理内存使用

监控被测服务器的物理内存占用率变化，帮助发现内存泄漏或内存不足导致的性能问题。



5.10.11 网络流量

监控被测服务器网络接口的上行和下行流量，评估网络带宽是否满足业务传输需求。



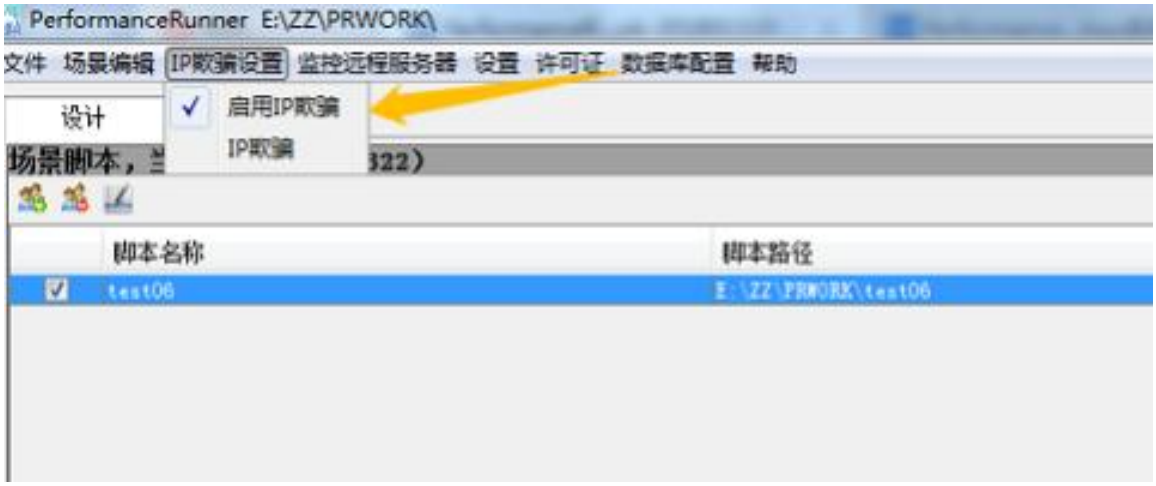
5.11 IP 欺骗技术

5.11.1 技术原理

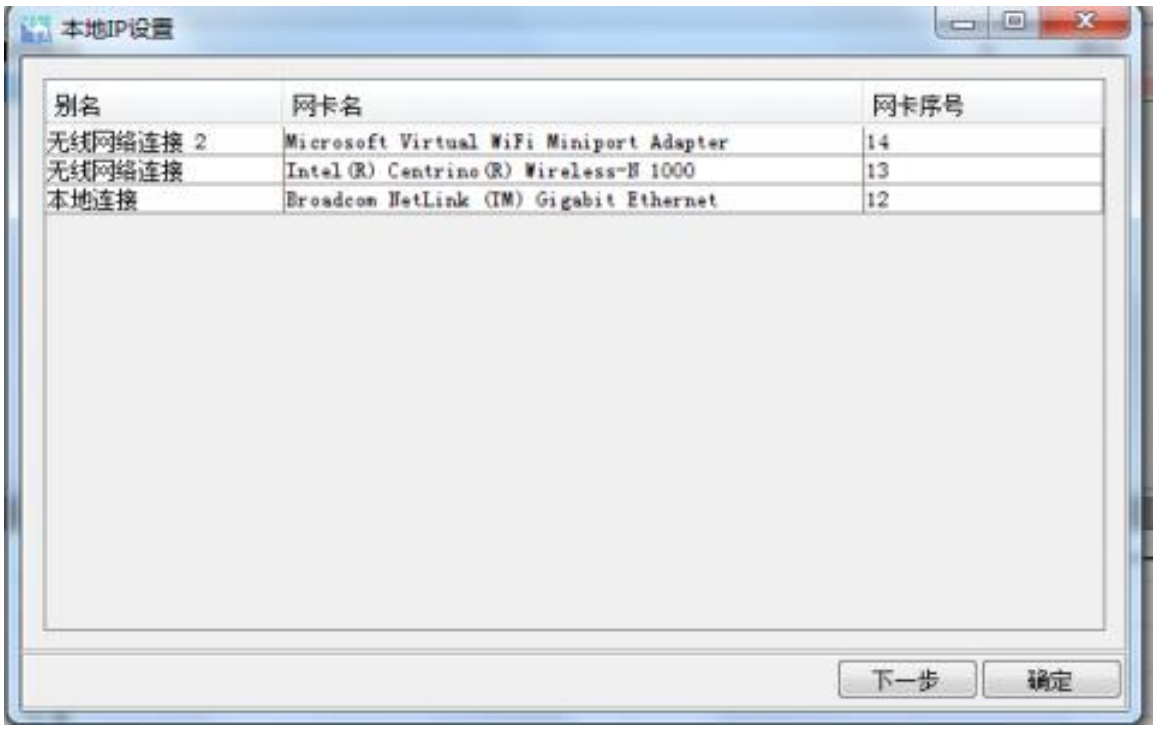
IP 地址欺骗（IP Spoofing）是一种网络测试技术，通过为测试主机绑定多个虚拟 IP 地址，使 PR 产生的请求数据包携带不同的源 IP 地址，从而模拟来自不同网络终端的用户访问行为。在标准的 IP 网络互联协议中，数据包头包含源地址和目的地址信息，PR 通过伪造源 IP 地址字段，实现多 IP 并发访问的仿真效果。

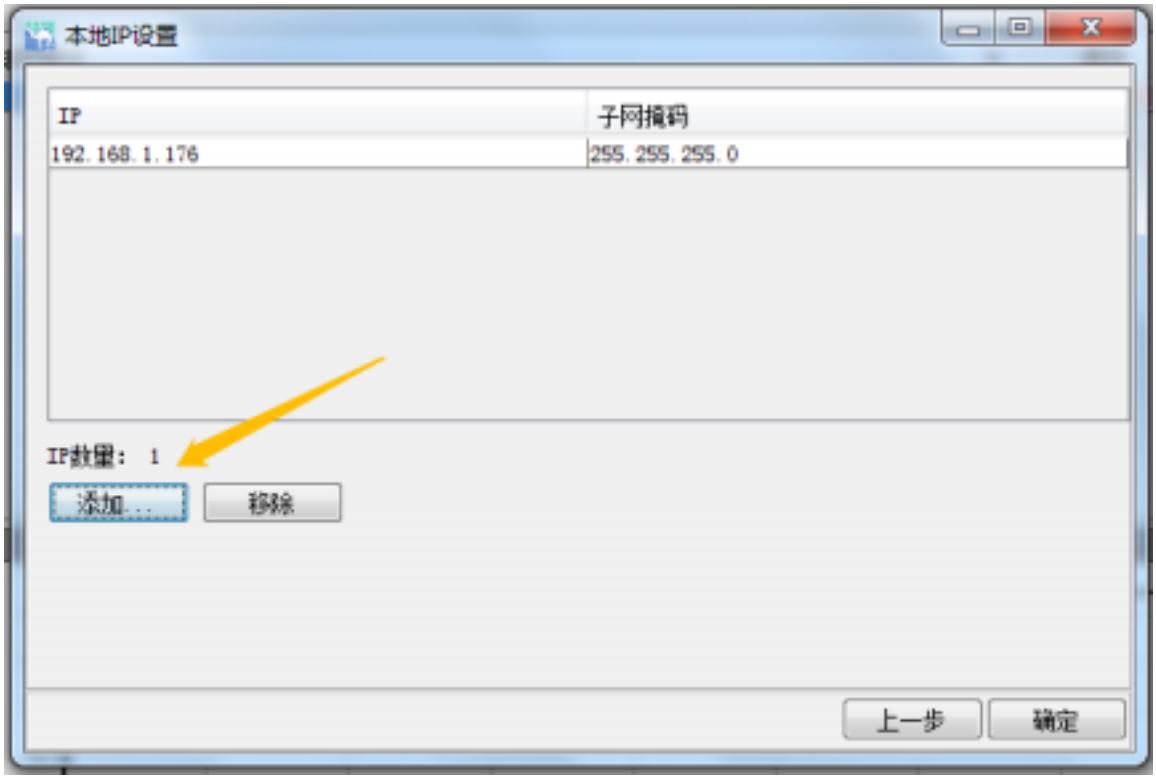
5.11.2 配置流程

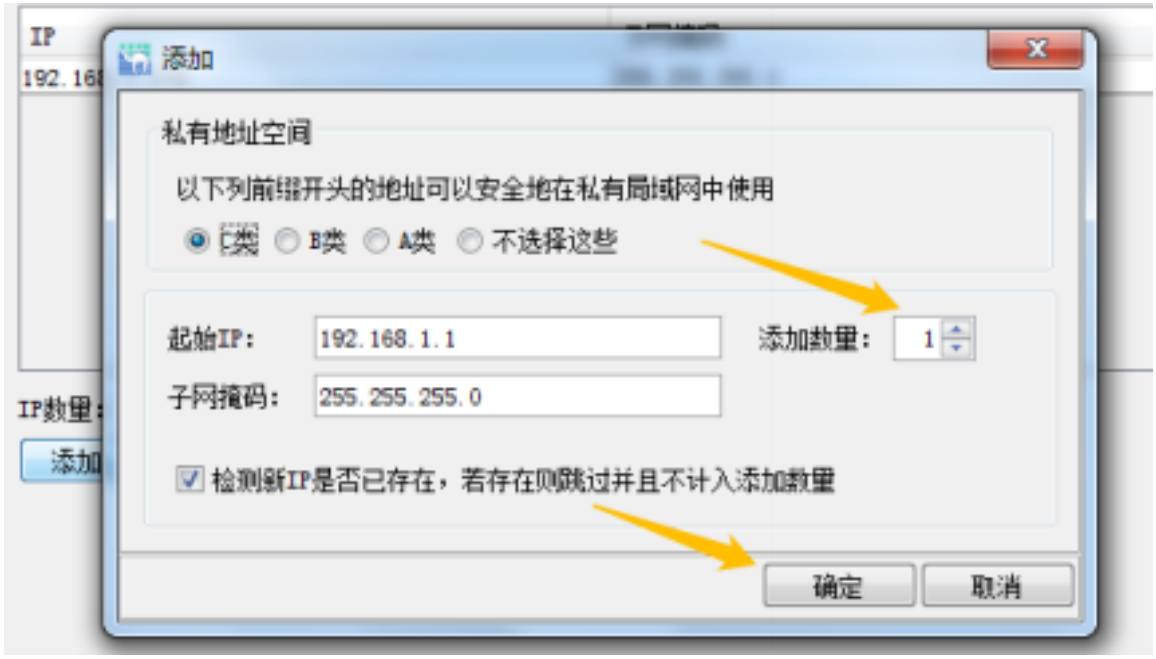
步骤 1：启用 IP 欺骗功能 - 进入菜单【IP 欺骗设置】→【启用 IP 欺骗】，勾选启用选项。

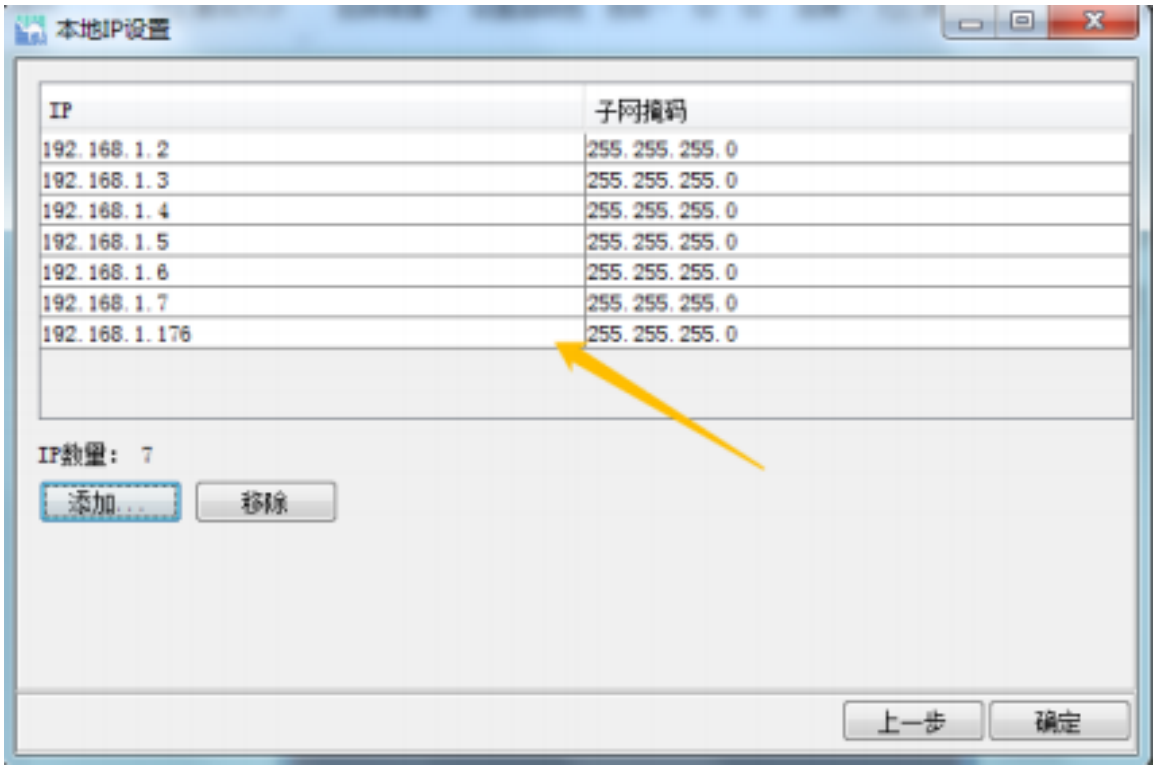


步骤 2：配置虚拟 IP 地址 - 点击【IP 欺骗】菜单，打开【本地 IP 设置】对话框。 - 选择本地物理网卡，进入 IP 地址配置向导。 - 指定 IP 地址类型、起始 IP 地址、子网掩码。 - 选择是否启用 IP 冲突检测（推荐启用，避免与网络中现有 IP 冲突）。 - 设置需要添加的虚拟 IP 数量，确认后完成配置。

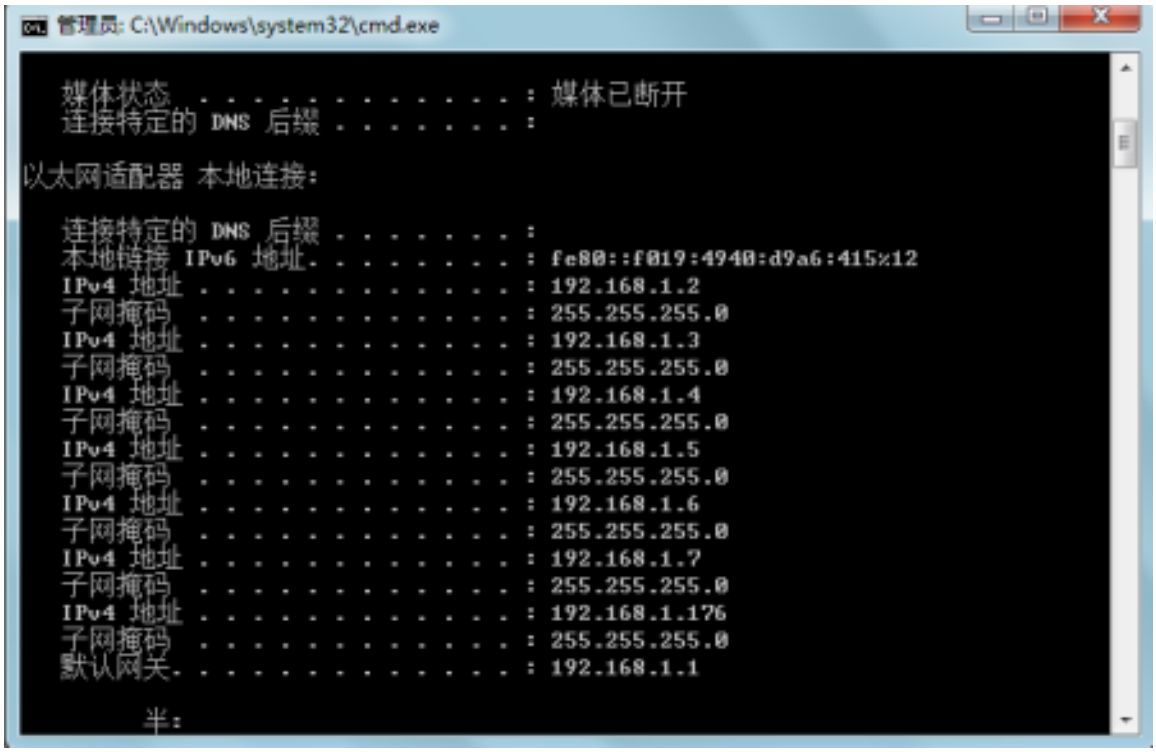



本地IP设置
IP
子网掩码
192.168.1.176
255.255.255.0
IP数量： 1
添加...
移除
上一步
确定


IP
添加
私有地址空间
以下列前缀开头的地址可以安全地在私有局域网中使用
C类
B类
A类
不选择这些
起始IP:
192.168.1.1
添加数量:
1
子网掩码:
255.255.255.0
检测新IP是否已存在，若存在则跳过并且不计入添加数量
确定
取消

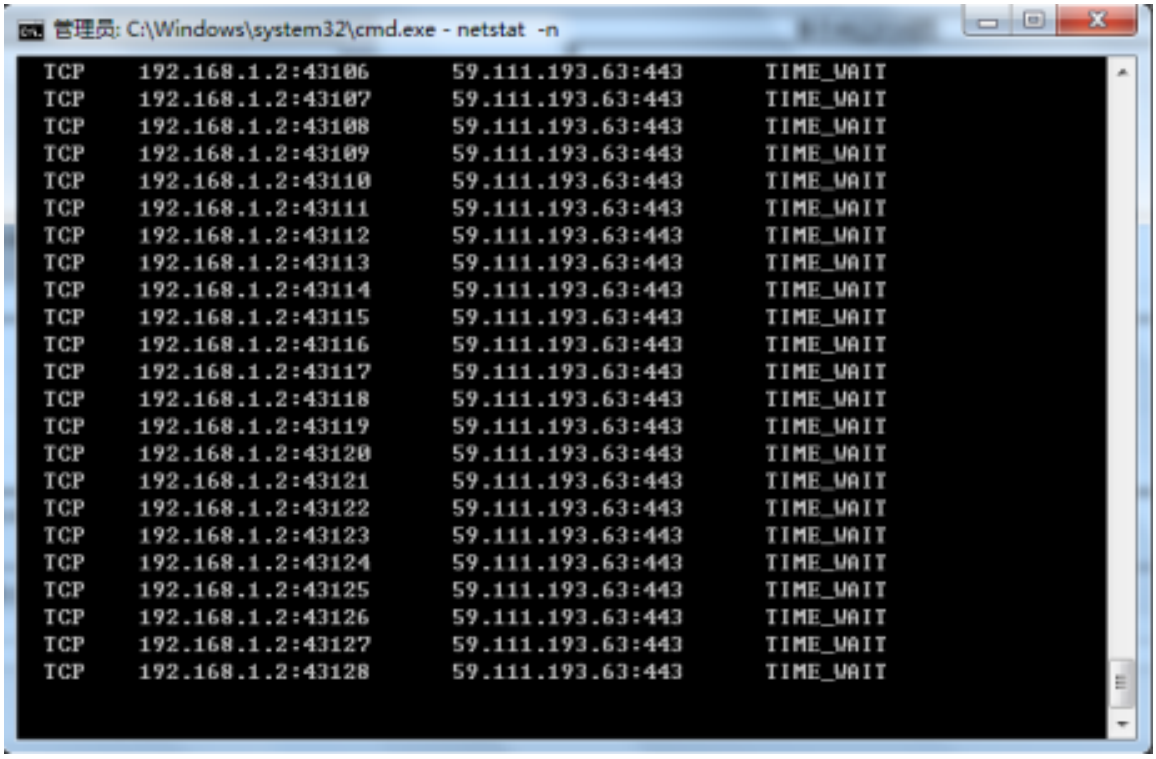


步骤 3：验证配置结果 - 打开 Windows 命令提示符，执行 `ipconfig` 命令，验证虚拟 IP 地址是否已成功绑定到网卡。



5.11.3 使用与验证

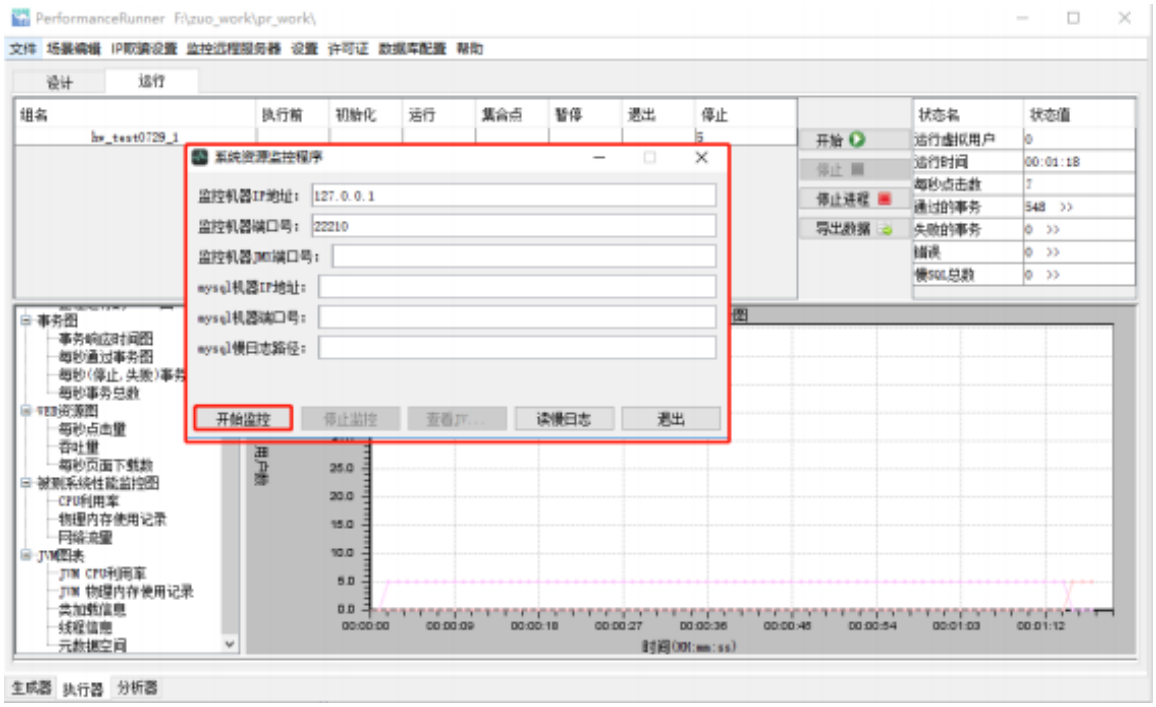
完成 IP 欺骗配置后，在执行器中加载测试项目，启动场景执行。在场景运行期间，通过在被测服务器上执行 `netstat -n` 命令，可观察到来自不同虚拟 IP 地址的连接请求，确认 IP 欺骗功能生效。



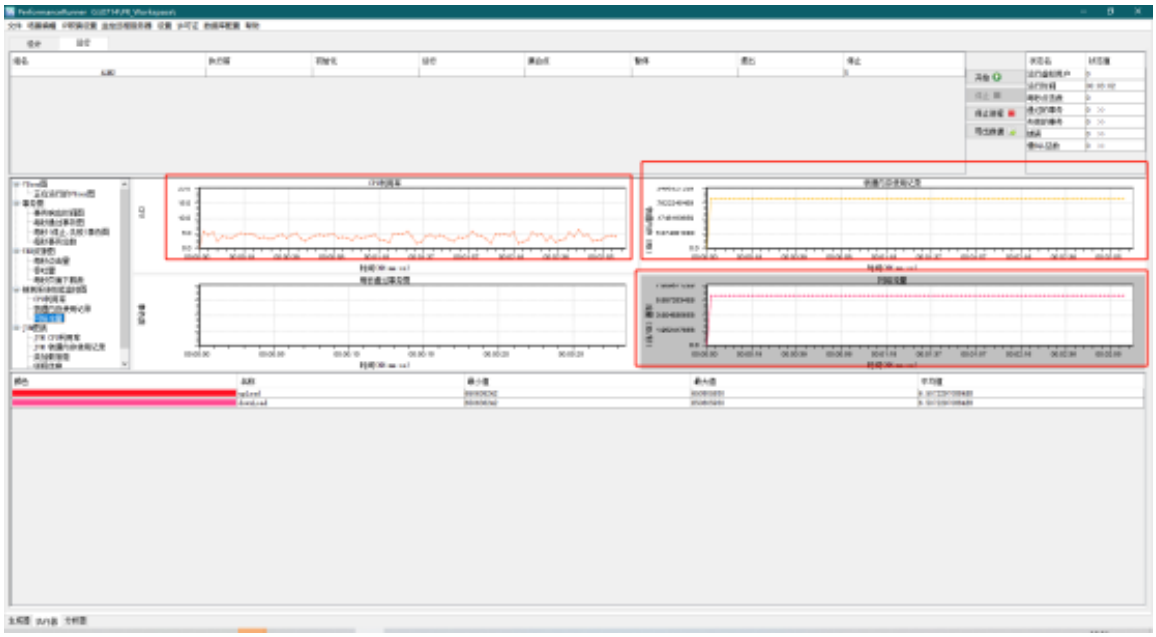
5.12 系统性能监控

PR 提供远程服务器性能监控能力，可在施压的同时实时采集被测目标服务器的核心性能指标。用户只需配置被测服务器的 IP 地址和监控端口，PR 即可建立监控连接，在测试执行过程中实时获取并展示以下监控数据：

- **CPU 利用率：** 处理器时间占用百分比
- **物理内存使用：** 内存占用量及使用率
- **网络流量：** 网络接口的收发数据量



PR 支持对 Windows 和 Linux 操作系统服务器的性能监控，同时支持对 Java 虚拟机（JVM）的深入监控，包括堆内存使用、GC 活动、线程状态等 JVM 专属指标，满足 Java 应用的深度性能分析需求。



6. 产品核心优势

评估一款自动化测试工具的核心维度包括：测试资产构建效率、二次开发成本控制、维护便捷性、数据驱动扩展能力、测试资产延续性以及平台扩展性。PerformanceRunner 在上述维度均表现出显著优势：

6.1 卓越的测试构建效率

业界经验表明，建立一个自动化测试脚本的初始时间成本约为手工执行一次测试的 3-10 倍。因此，降低测试资产构建成本是自动化测试工具竞争力的关键。

PR 通过以下机制实现效率突破：

智能组件识别： PR 具备业界领先的自动识别能力，录制的脚本通常可直接回放执行，无需大量手动修正。脚本能够自适应窗口位置、标题、尺寸变化以及组件属性变更，显著降低脚本开发的技术门槛和手工调整工作量。

内置数据驱动框架： 不同于其他工具仅提供参数化基础功能而要求用户自行搭建数据驱动框架，PR 内置标准化的数据驱动模式和数据格式规范。测试脚本建成后，增加新的测试数据组仅需数分钟，实现测试覆盖的快速扩展。

6.2 先进的模糊识别算法

PR 为每类 UI 组件定义了标准化的模糊识别指标体系。录制完成后，系统根据配置文件自动生成包含多维度识别权重的资源描述文件。脚本执行时，通过加权匹配算法进行模糊识别，即使面对界面布局微调、组件位置偏移、尺寸变化等情况，仍能准确定位目标元素。

用户可通过调整系统配置文件的识别参数权重，全局优化项目的识别策略；也可针对特定脚本手动编辑资源文件的权重属性，实现个性化的识别调优。模糊识别算法极大提升了脚本在迭代变更环境下的执行鲁棒性，避免了因界面微小调整导致的脚本失效问题。

6.3 关键字驱动技术

PR 提供领先的关键字驱动（Keyword-Driven）技术支持，实现专家视图与关键字视图的双模式协作：

- **专家视图：** 面向熟悉脚本编程的测试工程师，提供完整的 Groovy 脚本编辑能力，最大化开发效率。
- **关键字视图：** 面向非编程背景的测试人员，以可视化关键字方式呈现测试步骤，无需编写代码即可完成测试设计。

两种视图支持实时双向转换，既满足专业用户的高效率需求，也保障了业务用户的易用性体验，实现不同技能水平团队的高效协作。

6.4 简洁的脚本语法与强大的调试能力

PR 基于 Java 标准语法构建，测试人员仅需掌握基础语法即可开展测试开发。结合关键字驱动框架，日常维护工作甚至无需深入 Java 知识，大幅降低团队技术门槛。

PR 遵循 JDA（Java Debugger Architecture）标准，提供企业级的调试功能支持： - 断点设置与条件断点 - 单步执行（Step Over / Step Into / Step Out） - 变量值实时查看与修改变量 - 表达式求值与监控

这些调试能力帮助测试人员快速诊断脚本错误，显著缩短问题定位时间。

智能语法检查编辑器

PR 编辑器在标准关键字高亮基础上，集成实时语法检查引擎，在编码过程中动态分析语法正确性，对错误语法即时标注提示。这一特性对于编程经验有限的测试人员尤为重要，可有效预防常见语法错误，提升编码效率。

6.5 测试资产的延续性与扩展性

测试案例库是与应用版本紧密关联的重要组织资产。随着被测系统的版本演进，测试资产需同步升级，才能持续发挥回归测试的价值。

PR 通过以下机制保障测试资产的长期价值：

版本向下兼容： 产品升级不破坏历史脚本的可用性，确保既有测试资产持续有效。

Java 生态扩展： PR 从根本上基于 Java 语言构建，而非仅借用 Java 语法。这意味着 PR 可完整利用 Java 生态的发展成果，随着 JDK 版本的迭代持续增强自身能力。用户可自由引入第三方 Jar 包，在不受工具自身功能边界限制的前提下任意扩展脚本能力，实现真正意义上的无限扩展。

---

7. 技术支持与服务体系

上海泽众软件科技有限公司为 PerformanceRunner 用户提供全生命周期的专业技术支持与服务保障。

7.1 技术支持渠道

服务类型	联系方式
技术支持热线	400-035-7887
技术支持邮箱	support@spasvo.com
官方网站	https://www.spasvo.com/
微信公众号	泽众软件官方公众号

服务范围说明： 技术支持服务仅向正式授权用户和授权试用用户开放。联系技术支持时，请提供单位名称和服务代码以便快速验证身份。

7.2 产品服务与商务咨询

服务类型	联系方式
销售咨询热线	400-035-7887
销售邮箱	sales@spasvo.com

7.3 文档与培训服务

- 提供完整的用户操作手册、管理员配置手册及系统技术参考手册
- 产品版本升级后同步更新配套文档
- 提供专业的产品培训服务及认证体系

7.4 服务响应承诺

服务等级	响应时效
问题初步响应	2 小时内
解决方案提供	48 小时内

上海泽众软件科技有限公司致力于通过高效、专业的技术支持服务，保障用户的项目实施进度和系统运行质量，助力客户构建高性能、高可靠的软件系统。

文档结束
