文章编号：1005-8451（2015）06-0001-04

goTicket性能测试平台的开发与应用

于　澎，刘文韬，朱建生，任贝贝
（中国铁道科学研究院　电子计算技术研究所，北京　100081）

摘　要：为满足对互联网售票系统进行大并发模拟测试的需求，基于开源Grinder框架设计开发goTicket性能测试平台，并应用该平台对系统进行压力测试和性能调优。

关键词：互联网售票；压力测试；Grinder；性能调优
中图分类号：U293.22：TP39　**文献标识码：**A

goTicket performance testing platform

YU Peng, LIU Wentao, ZHU Jiansheng, REN Beibei
(Institute of Computer Technologies, China Academy of Railway Sciences, Beijing　100081, China)

Abstract: In order to meet the test demand on Internet Ticketing and Reservation System with high concurrency simulation, goTicket performance testing platform based on open source Grinder framework was designed, and applied for pressure testing and performance tuning.

Key words: Internet Ticketing and Reservation System; pressure testing; Grinder; performance tuning

12306互联网售票系统的研发与应用，拓展了购票渠道，使旅客出行购票变得更加方便快捷。随着订票用户的逐渐增加，旅客购票习惯的不断改变和订票功能的持续优化，系统最大日售票量已超过500万张，超过全路日售票量的50%。系统实时处理大量用户订票并发请求，并确保交易准确、完整，尤其在预售国庆节、春节车票期间，系统承担着巨大压力。系统处理用户请求速度快慢，响应时间长短，直接影响着用户对系统的体验度。因此，在系统上线前，有必要对系统进行性能测试，并对其进行性能调优，确保系统功能在大并发请求下正常稳定运行。

1　对比选型

目前业界有多种性能测试工具可用，如LoadRunner，它功能强大，支持众多常用协议，但是由于是商业产品，其需要依照并发用户数和协议购买指定时间内的授权，并且软件较庞大，对系统配置要求较高；Compuware Gomez是一款拥有云服务器的性能测试工具，需要协调各地云服务器来产生压力，所以每次测试准备和运行的周期较长，使用费用昂贵。众多开源免费的性能测试工具，如Jmeter、Tsung、Grinder等，对模拟互联网用户请求有较好的支持，但较少支持其它协议，缺少对测试结果的分析展示，工具安装部署比较繁琐，操作使用不友好，因此用户使用工具的学习成本较高，体验度差。各种性能测试工具的对比如表1所示。

表1　各种性能测试工具对比

性能测试工具	使用授权	消耗资源	HTTP协议	其它协议	友好性	操作性	环境配置
LoadRunner	收费	大	支持	支持	好	简单	简单
Gomez	收费	云服务器	支持	不支持	好	复杂	云服务器
Tsung	开源	少	支持	不支持	差	复杂	复杂
Jmeter	开源	中	支持	不支持	差	简单	复杂
Grinder	开源	中	支持	插件支持	差	简单	简单

针对12306互联网售票高并发的特点，适宜采用模拟HTTP协议的测试架构进行压力测试，其中Jmeter、Tsung和Grinder都可以模拟HTTP协议请求，但是Jmeter、Tsung架构安装配置比较复杂，Tsung运行环境中要求安装并配置Erlang并发语言环境，并且两种工具的测试脚本编写不够灵活，无法真实模拟用户请求流量。Grinder自身结合了HTTPClient插件，对协议可以灵活处理，可以模拟访问服务器的数据流量，测试脚本采用Jython语言编写，可以采用Java第三方依赖包，可扩展性更强，脚本参数

收稿日期：2014-11-03
作者简介：于　澎，助理研究员；刘文韬，副研究员。

化也变得灵活。此外，Grinder 架构自身基于 Java 编写，压力系统较轻便，较容易对服务器产生压力，工具容易使用，学习成本较低，因此可以采用 Grinder 性能测试工具对互联网售票系统进行性能测试。

2 平台架构设计

2.1 开源测试框架

Grinder 是一套性能测试框架，如图 1 所示，主要是由多台机器同时运行测试脚本以实现对被测系统产生压力，并实时收集运行场景的响应时间和 TPS 等信息。整个测试框架由 3 部分组成，包括控制台、代理进程和工作进程。其中，控制台负责协调代理和工作进程，收集和显示测试结果，提供测试脚本简单编辑和分发功能；代理进程在后台持续运行，主要负责启动和停止工作进程，并接收控制台发送的测试脚本进行本地缓存；工作进程主要任务是重复执行测试脚本，每个进程可以有多个线程独立的运行脚本，并将测试结果发送给控制台。

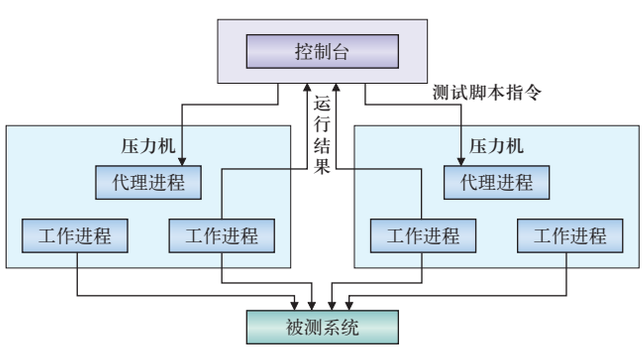

图1 Grinder性能测试框架图

2.2 平台改造

Grinder 性能测试框架中控制台为 C/S 架构，界面不友好，使用过程中测试结果无法实时展现。为了使之在对互联网售票系统进行性能测试和调优过程中能够起到指导作用，有必要对其进行以下 5 方面的开发改造：(1) 对其已有功能进行 Web 化改造；(2) 对测试脚本进行版本控制；(3) 对测试结果进行实时展示和保存；(4) 对历史结果可查看回放；(5) 对系统架构进行优化，使系统易于安装部署，实现可以根据压力需求进行横向拓展的目标。

goTicket 测试平台采用 Grinder 开源测试框架并进行二次开发，是一整套模拟 12306 用户订票业务的性能测试解决方案，平台采用 B/S 架构模式，界面友好，用户操作简单方便。goTicket 性能测试平台对不同级别用户进行角色管理，其中使用者可以进行性能脚本编辑与管理，性能测试场景创建与执行，性能测试结果收集与展示；管理者除具备以上功能外，还可以进行用户管理、压力机管理和系统配置修改等功能操作。

2.3 平台功能逻辑

goTicket 性能测试平台由一个控制台和多台压力机共同组成。其中控制台负责与用户进行交互，用户通过网页可以在线编辑与管理测试脚本，创建与管理测试场景，启动与停止性能测试运行，查看性能测试结果。压力机负责接收控制台指令，执行测试脚本，并将运行结果返回给控制台。另外在被测系统上可以部署监控程序，负责收集被测系统性能指标，并回传给控制台。控制台是整个测试平台的核心部分，负责与用户交互，压力机控制和结果展示，也是整个平台开发的重点。

goTicket 性能测试平台架构如图 2 所示。

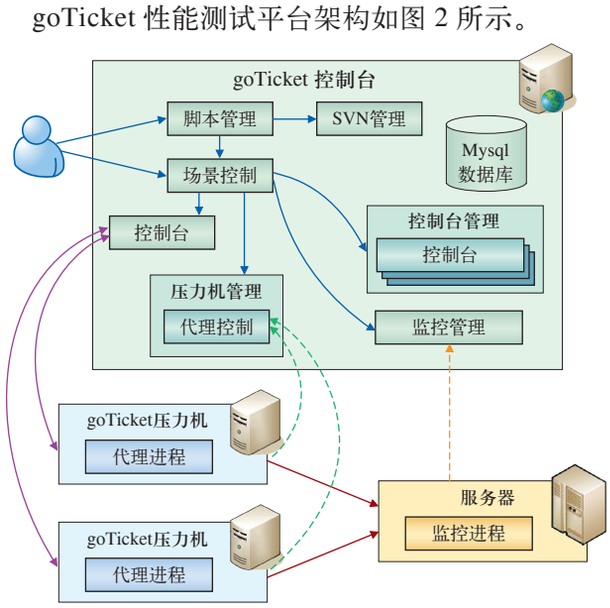

图2 goTicket性能测试平台架构图

goTicket 控制台主要由脚本管理、场景控制、SVN 管理、控制台管理、压力机管理和监控管理组成。(1) 脚本管理模块与用户进行交互，提供列表式的脚本展示，可以记录脚本名称，脚本附加信息，脚本修改时间，版本和大小等信息，可以实现脚本的新建和已有脚本的删除。脚本管理整合 Grinder 脚本调试功能，对于用户已建的脚本可以单次运行，方便

用户查看脚本的正确性，并且可以进行简单编辑修改。脚本通过 SVN 模块进行版本控制，对于线下编辑的脚本可直接上传至 SVN 中，新增脚本显示在脚本列表中，其它模块通过调用统一的 API 接口实现脚本的签出、提交和分发，以及脚本信息查询等操作。(2) 控制台管理模块管理多个控制台进程，当有用户计划测试时，管理模块给用户分配一个可用的控制台，与测试场景绑定，并分配可用的压力机资源。(3) 场景控制模块负责与用户进行交互，如测试定义、编辑和运行等操作，以列表形式展示测试场景概要信息，包括已运行完成、正在运行中和计划运行的测试场景，可以查看测试场景的详细信息。正在运行的场景通过控制台将指定的测试脚本分发到压力机上，并控制压力机上的代理进程产生相应数量的虚拟用户，重复执行测试脚本，从而对被测系统产生压力。(4) 压力机管理模块负责控制平台可用的压力资源，实现在平台内压力资源的注册、分配，以及压力机系统状态的实时监控。(5) 监控管理模块收集被测系统上监控进程发送的信息，包括系统 CPU 使用率、内存使用率和网络流量信息。控制台内部整合了 Mysql 数据库，负责存储用户、脚本和场景信息，为测试场景结果展示提供数据基础。

goTicket 压力机组成相对简单，其采用 Grinder 架构的代理模式，主要包括代理进程，实时与控制台进行交互。当测试运行时，压力机会依据场景设置的虚拟用户数，启动相应数量的进程和线程，重复执行测试脚本中定义的操作，最终实现对目标系统进行不间断的并发请求，并将测试执行的指标，包括运行时间、请求数量、完成数量、压力机状态等信息发送到控制台。

3 关键技术

3.1 操作流程Web化

将 Grinder 测试流程迁移到网页形式是 goTicket 性能测试平台开发的重点，对原控制台程序操作的过程进行易用性优化。主要包括测试脚本编辑和验证，测试场景通过页面形式进行配置，测试运行中实时显示运行结果和对历史结果的查询与展示。整个平台采用 Spring 框架模式开发实现，数据库存储用户定义、场景定义等信息，采用 Mysql 数据库，采用 SVNkit 进行版本管理，支持脚本下载本地调试与上载，使用 FreeMaker 编写模板引擎，用于表现层界面实现。采用第三方 JavaScript 库 Jplot 实现曲线图的展示。平台采用 Tomcat 服务器，部署在 Linux 操作系统上，对系统的配置要求较低。

3.2 用户角色管理

goTicket 性能测试平台对不同级别用户进行角色管理，明确各角色可操作的范围。管理员可以查看平台的运行状况，增加与减少可用的压力机数量，新增、修改和删除用户。普通用户更接近测试，可以编辑、修改和调试脚本，执行测试场景等。在压力资源许可的情况下，多个用户可以同时运行测试场景。

3.3 测试结果展示

goTicket 性能测试平台对实时测试数据进行收集、处理和展示。测试运行时，压力机重复运行测试脚本，记录所定义事务的执行次数、执行时间、运行结果等数据，并将数据发送到控制台进行处理。控制台依据采样间隔时间对收集的数据进行分析处理，通过表格和动态曲线图的方式展示测试的响应时间和 TPS 关键数据，使得用户对运行的测试场景有直观的了解。所有的测试数据和分析处理后的结果都会存储在平台的 Mysql 数据库内，这使得用户查看历史测试结果更加方便。

3.4 Jython脚本支持

goTicket 性能测试平台的学习和使用成本大大降低，其主要原因是其继承了 Grinder 对 Jython 测试脚本的支持。Grinder 架构针对 HTTP 协议有录制脚本工具和丰富的插件，并且有大量的 HTTP 协议处理工具，这些都使得基于 HTTP 协议的测试脚本容易编写与维护。Jython 是 Python 解释器在 Java 中的完整重现，其继承 Python 语法风格，编写简单，同时又继承了 Java 的类库，再加上测试脚本支持引用第三方 Java 类库，这使得脚本功能极大地提升。

4 平台应用

4.1 手机订票性能测试

手机订票作为互联网售票系统的拓展，其共用

一部分互联网售票系统组件，但也新增部分应用服务器，为确保上线后手机服务器稳定运行，有必要模拟多客户端访问请求产生对应用服务器的压力。由于移动设备访问应用服务器需要进行多步确认，模拟操作处理过程相对复杂，单个业务逻辑内请求较多，所以测试脚本编写较困难，维护也有较大难度。测试过程经过了多天多轮次的压力测试，又经过数天的系统调优过程。测试模拟 Android 用户直压 WorkLight，压测 WorkLight 负载均衡，直压 Nginx，压测 Nginx 负载均衡多个测试场景，并模拟不同并发用户数量请求。测试场景模拟 10 000 个并发用户，思考时间为 100 ms，其中使用 10 台压力机，每台压力机启动 10 个工作进程，每个进程产生 100 个线程，总 TPS 为 38 200，平均响应时间 150 ms。

手机订票性能测试 TPS 曲线图和响应时间曲线图如图 3、图 4 所示。

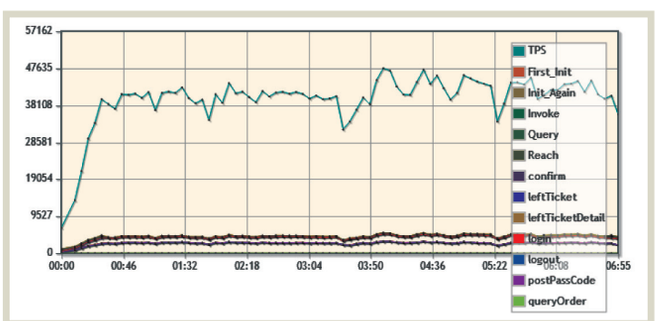

图3　手机订票性能测试TPS曲线图

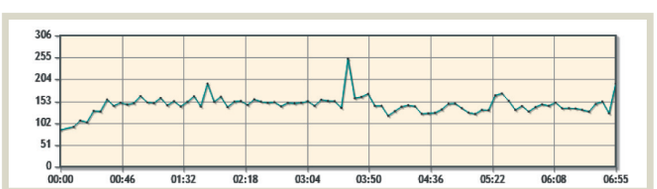

图4　手机订票性能测试响应时间曲线图

4.2　互联网售票扩能性能测试

为保证互联网售票系统在春运期间平稳运行，系统进行应急扩能，扩展 A5、A6、A7 共 3 组售票节点，且首次使用 X86 架构平台承载售票核心数据库。goTicket 测试平台对线上功能进行压力测试，模拟用户登入、订票、查询订单和登出操作。测试场景模拟 200 个并发用户订票，思考时间为 100 ms，其中使用 5 台压力机，每台压力机启动 2 个工作进程，每个进程产生 20 个线程，总 TPS 为 160，平均响应时间 680 ms。

互联网售票扩能性能测试 TPS 曲线图和响应时间曲线图如图 5、图 6 所示。

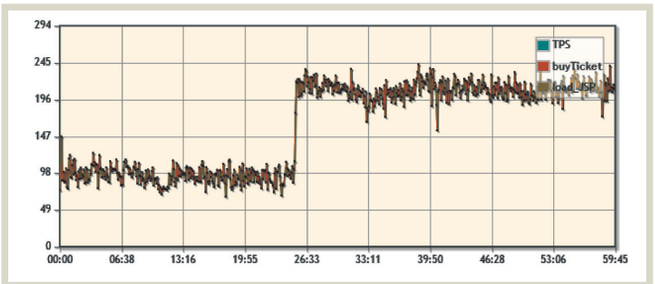

图5　互联网售票扩能性能测试TPS曲线图

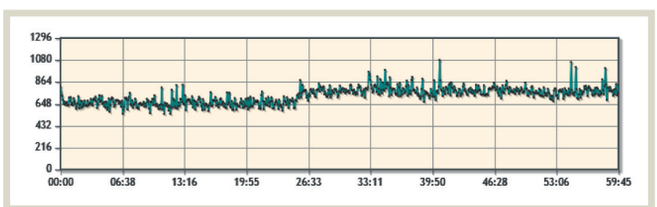

图6　互联网售票扩能性能测试响应时间曲线图

5　结束语

goTicket 性能测试平台作为客票系统的一个组成部分，目前已用于模拟 HTTP 协议，实现对互联网售票系统的性能测试，并且在平台持续产生压力的情况下对被测系统进行性能调优，平台将持续提供对系统不同需求的测试服务。客票系统是一个大型分布式系统，其数据库的稳定运行和客票核心业务的正确运行同样重要，因此有必要对 goTicket 性能测试平台进行拓展，为以上功能的测试与调优提供更强的技术保障。另一方面，基于互联网业界迅速发展的云服务，可以研究和部署互联网云服务器压力测试环境，模拟互联网用户对系统所产生的压力。

参考文献：

[1] 赵　斌．软件测试技术经典教程 [M]．北京：科学出版社，2011.

[2] 陈能技，郭柏雅．性能测试诊断分析与优化 [M]．北京：电子工业出版社，2012.

责任编辑　陈　蓉