# 性能测试的原理及其自动化工具的实现

谭 浩， 关 昕， 马 力
（华北计算技术研究所， 北京 100083）

**摘 要**：软件组织在对其开发的系统进行性能测试时，可能需要自己编写一些性能测试的工具。但是，性能测试工具的编写是一件复杂的事情，性能测试工具需要满足以下要求：占用系统资源少，可扩展性好，使用性强，能够模拟大规模的用户，能模拟真实的环境等。探讨了在一般的企业级应用中性能测试的原理及基本要求，根据这些需求提出了一个性能测试工具的实现架构，分析了其中的一些关键技术。

**关键词**：性能测试； 自动化工具； 架构； 插件； 压力
**中图法分类号**：TP311.5 **文献标识码**：A **文章编号**：1000-7024 (2006) 19-3660-03

## Principle of performance test and realization of its automatic tools

TAN Hao, GUAN Xin, MA Li
(North China Institute of Computing Technology, Beijing 100083, China)

**Abstract**: When the software organization tests the performance of their systems, some tools need be developed for the performance test. But it is complex to develop a tool for the performance test. It should meet such requirements: need the resource of systems as little as possible, easy to extend, easy to use, simulate large amount users, simulate real environment etc. The principle and the essential requirement in enterprise scale application is discussed, and then a framework and realization method of a tool for testing the performance of applications is put forward, Finally some pivotal techniques in realization the automatic tools are analyzed.

**Key words**: testing the performance of applications; automatic testing tools; framework; plug-in; stress the application

## 0 引 言

随着企业需求的日益增长以及计算技术的进步，企业级系统的应用已经从早期的一个人使用的单机时代转换到了服务成千上万个用户的跨广域网的庞大时代。随着企业业务量的增加，企业的应用系统可能同时要为成百上千，乃至上万的用户提供服务，其承载的负荷越来越重，系统性能的好坏严重影响了企业对外提供服务的质量。现在，对企业的应用系统的性能进行测试和调优越来越引起企业的重视。目前，典型的企业应用系统的架构如图 1 所示。

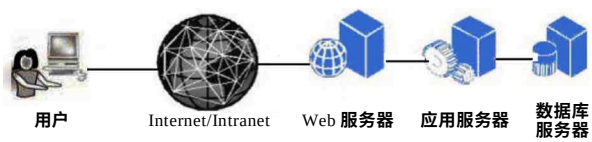


图 1 典型企业应用系统架构

这样的系统通常由客户端、网络、防火墙、负载均衡器、Web 服务器，应用服务器(中间件)、数据库等环节组成，根据木桶原理，木桶所能盛水的多少取决于最短的那块木板，如果要调优系统的性能，则必须找到整个系统的瓶颈。

## 1 性能测试的原理

一般而言，对企业的应用系统进行测试包括下面的两个部分：查看系统在一定的负载条件下运行是否合乎需求及通过系统在一定负载下运行的所得的数据分析它的瓶颈。因此，能够对当前的系统产生一定的负载，并且能够分析系统在一定负载下的表现是实现性能测试的关键。一般来说，使系统产生负载有以下 4 种方法：

(1)重复：性能测试对系统产生压力的基本手段。换句话说，测试的重复就是一遍又一遍地执行某个操作或功能，比如重复调用一个 Web 服务。功能验证测试可以用来被弄清楚一个操作能否正常执行。而性能测试将确定一个操作能否正常执行，并且能否继续在每次执行时都正常。这对于推断一个产品是否适用于某种生产情况至关重要。客户通常会重复使用产品，因此压力测试应该在客户之前发现代码错误。但是仅仅只有这么一个条件是不行的，简单地扩展功能验证测试来多次重复并不能构成一个有效的压力测试。当与下面的一些原则结合起来使用时，重复就可以发现许多隐蔽的代码错误。

(2)并发：并发是同时执行多个操作的行为。换句话说，就是在同一时间执行多个测试，例如在同一个服务器上同时调

收稿日期：2005-08-06。
作者简介：谭浩 (1982－)，男，湖北天门人，硕士研究生，研究方向为软件度量； 关昕，女，研究方向为软件项目管理和软件测试； 马力，女，研究员，硕士生导师，研究方向为软件工程理论和质量控制。

用许多 Web 服务。这个原则不一定适用于所有的产品 (比如无状态服务),但是多数软件都具有某个并发行为或多线程行为元素,这一点只能通过同时执行多个代码示例才能测出来。压力系统要能够同时遍历多条代码路径。例如,一个 Web 服务压力测试需要一次模拟多个客户机。Web 服务通常会访问多个线程实例间的一些共享数据。因额外方面的编程而增加的复杂性通常意味着代码会具有许多因并发引起的错误。由于引入并发性意味着一个线程中的代码有可能被其它线程中的代码中断,所以错误只在一个指令集以特定的顺序(例如以特定的定时条件) 执行时才会被发现。把这个原则与重复原则结合在一起,您可以发现程序中的许多问题。

(3)量级 :性能测试应该应用于产品的方面是每个操作中的负载量。压力测试可以重复执行一个操作,但是操作自身也要尽量给产品增加负担。例如,一个 Web 服务允许客户机输入一条消息,您可以通过模拟输入超长消息的客户机来使这个单独的操作进行高强度的使用。换句话说就是,您增加了这个操作的量级。这个量级总是特定于应用的,但是可以通过查找产品的可被用户计量和修改的值来确定它——例如,数据的大小、延迟的长度、资金数量的转移、输入速度以及输入的变化等等。单独的高强度操作自身可能发现不了代码错误 (或者仅能发现功能上的缺陷),但与其它条件结合在一起时,您将可以增加发现问题的机会。

(4)随机变化 :任何性能测试都多多少少应该具有一些随机性。如果您对系统中需要输入的一些参数作出一些变化,您就能够在每次测试运行时应用许多不同的代码路径。比如:使用重复时,在重新启动或重新连接服务之前,您可以改变重复操作间的时间间隔、重复的次数,或者也可以改变被重复的 Web 服务的顺序。使用并发,您可以改变一起执行的 Web 服务、同一时间运行的 Web 服务数目,或者也可以改变关于是运行许多不同的服务还是运行许多同样的实例的决定。量级或许是最容易更改的,每次重复测试时都可以更改应用程序中出现的变量(例如,发送各种大小的消息或数字输入值)。如果测试完全随机的话,很难一致地重现压力下的错误,可以使用基于一个固定随机种子的随机变化。这样,用同一个种子,重现错误的机会就会更大。

同时,性能测试应该记录下系统在压力下运行时的一些参数,例如响应时间、最大/最小并发数、失败的次数、正常连续运行的最长/最短时间、并发数与失败的关系等,这些数据能够帮助测试者很快地找到系统地瓶颈,并且性能测试应该具备自动分析这些数据的功能,否则,一次性能测试完毕后,记录下了一些数据,这些数据的量非常大,往往包括几千个变量的运行曲线,大小可能达到上 G 的规模。靠人工去分析这些数据几乎是不可能的,性能测试必需提供数据分析工具,帮助性能测试人员去阅读、解读和分析数据,辅助测试人员定位系统的瓶颈。

## 2 自动化工具的实现

现在,随着软件的规模越来越大,现在手工编写性能测试的测试程序已经变得越来越不现实,因为大量的测试数据需要输入,大规模的程序需要维护。而且,市面上的一些测试软件如 Loadrunner 又太贵,因此,编写专门的性能测试工具已是企业的急切所需了。根据上述原理,一个好的性能测试工具必须满足以下几个方面:

(1)提供产生压力的手段 :产生压力的手段,主要是通过编写压力脚本,这些脚本以多个进程或者线程的形式在客户端进行运行,来模拟多用户对被测系统的并发访问,以此达到产生压力的目的。由于一台普通的 PC 机可以轻易产生几百乃至上千个进程或线程,通过使用若干台 PC 机,就可以轻易模拟出成千上万个并发用户。压力脚本执行的功能和被测系统客户端软件执行的功能应该一样,从而产生真正的业务压力。编写压力脚本的工作实际上就是重新编写客户端软件。为了快速达到编写脚本的目的,采用的最有效的方式是通过性能测试工具录制客户端软件和服务器之间的通讯包,自动产生脚本,然后在自动生成的脚本的基础上进行少量修改,如 :关联动态内容,指定批量测试数据等。

(2)能够对后台系统进行监控 :压力脚本的方式给我们提供了模拟各种压力状况的有力手段,通过人为制造各种类型的压力,我们可以观察被测系统在各种压力状况下的表现,从而定位系统瓶颈,作为系统调优的基础。因此,提供丰富的对后台系统进行监控的手段是性能测试工具第 2 个最主要的评价指标。监控应该不在被测系统上安装任何软件,否则的话,为了监控而引入的“代理”之类的小软件将给被测系统引入新的可变因素,一方面造成了测试结果的不准确,另一方面会给用户的系统的稳定性可能造成影响,从而导致用户的反感和拒绝。目前,各种监控手段大都采用所谓“无代理”的方式,即不在被测系统上安装任何软件,仅仅通过改变被测系统的配置,就可以对被测系统进行监控。需要监控的部件多种多样,包括操作系统、数据库、中间件、应用系统、安全模块、网络、防火墙等等。

(3)对压力测试产生的数据能够进行分析,快速找出被测系统的瓶颈 :一组压力测试运行完毕后,我们会得到详尽的性能数据。这些数据包括最终用户的响应时间,后台系统各个部件的运行数据。这些数据的量非常大,往往包括几千个变量的运行曲线,大小可能达到上 G 的规模。靠人工去分析这些数据几乎是不可能的,性能测试工具必需提供数据分析工具,帮助性能测试人员去阅读、解读和分析数据,辅助测试人员定位系统的瓶颈。数据分析工具是保证最终测试成果的手段,因此它是性能测试工具中最重要的部分之一。

通过上面的分析,我们设计出系统的体系结构如图 2 所示。其中,整个系统分 4 个部分组成:

· 虚拟用户脚本产生器 (virtual user generator,Vugen) :主要作用是进行脚本的录制,脚本录制的方式是通过代理的方式录制,通过代理录制客户端和后台服务器之间的通讯包,分析其中的协议,自动产生脚本。用户可以在自动产生的脚本的基础上进行修改,从而快速开发出一个逻辑功能和客户端软件完全一样的压力脚本程序。如图 3 所示。

Vugen 首先接收到用户发送的报文,对其进行分析,转换成对应的 API 调用 (根据脚本所用语言的不同,API 调用函数也不同),然后再转发给服务器,同理,从服务器传回的报文也同样通过 Vugen,Vugen 对其进行分析记录,然后转发给用户。

· 压力调度和监控系统 Conductor 和压力产生器 Player:

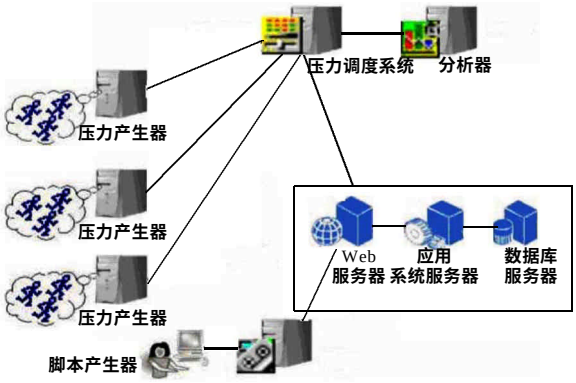


图 2　系统体系结构


图 3　Vugen 的原理

在我们的工具中，压力产生器 Player 主要用来产生一系列进程或线程，这些进程或线程运行由 Vugen 生成的脚本，而 Conductor 主要负责 Player 怎么来运行脚本，控制 Player 的个数，以下是 Conductor 和 Player 的交互图，如图 4 所示。

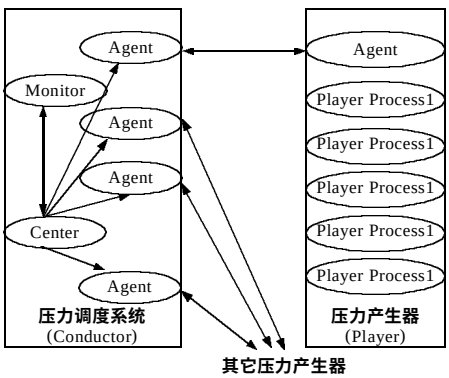


图 4　Conductor 和 Player 的交互图

如图 4 所示，Conductor 上面有若干进程/线程。每种进程/线程的作用如下：Center 进程是整个调度的核心进程，它负责联系和用户界面打交道的工作。Agent 进程负责和远端的 Player 机器中对应的 Agent 进程通讯。负责把编译好的脚本传送到 Player 机器上。在脚本运行的时候，定期从 Player 机器上获取 Player 的运行状态，每个虚拟用户运行的日志。Monitor 进程负责对被测试系统的各个环节进行监控，并把监控的内容一方面写入 Conductor 机器的本地磁盘，另外一方面把监控的内容传送给 Center 进程，实时地显示在用户界面上。Player 的进程有两种：Agent 进程和 Player 进程。Agent 负责和 Conductor 机器通讯，它根据 Conductor 的指示，在本机器上派生出指定数目的 Player 进程，这些 Player 进程负责具体执行相应的脚本。Player 进程个数就是虚拟用户的个数。

·压力结果分析工具 Analysis：该工具是一个纯数学工具软件，我们采用了目前市场上已经存在的数据处理的软件 Matlab。我们将压力测试所产生的数据直接导入其中进行处理。

## 3　实现所需的关键技术

Vugen 在分析和记录所得的数据报文时，如何根据捕获的数据包来反解析成对应的网络协议。在解决这个问题时，我们主要采用单个进程来侦听以太网上所有 IP 报文，分析其报头以判断出这个报文是哪个方面的报文，根据分析所得的结果(主要是所属协议，数据包大小)产生 Vugen 所用语言的相对应的 API 调用。

在设计 Vugen 的体系结构时，我主要采取了插件的设计模式，因为刚开始不可能开发出能录用多种协议的报文，在以后对其功能进行扩展时，我们可以逐渐地加入新的协议而不必改动应用程序，而且也可以让任何对开发协议解析器感兴趣的开发者在一个统一的框架下开发。整个 Vugen 的体系结构如图 5 所示。

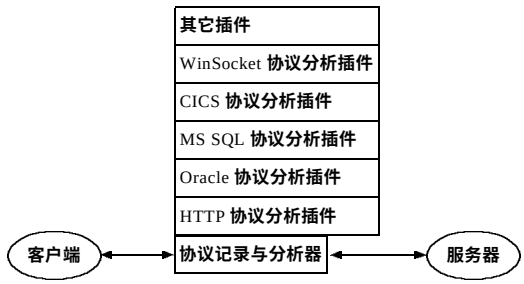


图 5　Vugen 的体系结构

## 4　结束语

性能需求在当今的企业用户的需求中所占的比例越来越大，同时开发企业自己的性能测试工具也是企业的当务之急，本文所提出的原理和方法能够适应大多数的企业需要。在现代的企业应用系统中，每个环节的都是一个很复杂的子系统，对其调优都是一门专门的技能。对于整个 IT 系统的调优，其复杂程度更是急剧增加。因此仅仅只开发高效的工具是不够的，企业除了开发适合自己的性能测试工具之外，还要不断提高测试人员的基本素质，这样才能保证企业开发出合乎用户需求的应用系统。

## 参考文献：

[1] Ron Patton.软件测试[M].北京:机械工业出版社, 2002.
[2] Mercury Interactive.LoadRunner7.6 用户手册[EB/CD]. 2004.
[3] Mercury Interactive.WinRunner7.5 用户手册[EB/CD]. 2004.
[4] John Watkins. 实用软件测试过程[M]. 北京: 机械工业出版社, 2004.
[5] 郑人杰,王伟,王方德.基于软件能力成熟度模型(CMM)的软件过程改进[M].北京:清华大学出版社, 2003.
[6] Roger S Pressman.软件工程——实践者的研究方法[M].北京: 清华大学出版社, 2002.
[7] 陈建勋,马于涛,谢敏,等. 软件过程的模型化研究[J]. 计算机工程与设计, 2004,25(1):33-35.
[8] 文昌辞,王昭顺. 软件测试自动化静态分析研究[J]. 计算机工程与设计, 2005,26(4): 987-989.