五、系统测试

内容提要

1. 系统测试概述
2. 功能性测试
3. 非功能性测试

1 系统测试概述

- 系统测试的概念
  - 将集成测试后的软件，作为计算机系统的一个部分，与计算机硬件、某些支撑软件、数据和人员等系统元素结合起来，在实际运行环境下对计算机系统进行一系列的严格有效的测试来发现软件的潜在问题，保证系统的运行。

- 实施人员：

  独立测试团队

  （引入独立视角，有助于发现遗漏缺陷）

系统测试的入口标准

- 入口标准
  - 所有的单元测试和集成测试成功完成
  - 软件的生成(build)过程没有任何错误
  - 软件版本经过了冒烟测试(版本验证测试)
  - 配套文档完成
  - 缺陷已经修正并且准备重新测试
  - 源代码已经存储在版本控制系统
- 满足以上标准后,测试组才接受软件版本并开始测试

系统测试的出口标准

- 出口标准
  - 已经执行了用来确定系统满足指定的功能性和非功能性需求的测试过程
  - 测试执行过程中没有出现任何重大错误
    - 高优先级的问题已经被修正，并且用回归测试进行了验证
    - 软件发布时可能存在已知的低优先级的缺陷（+若干未知缺陷）
  - 一些度量也可以作为出口标准的一部分
    - 缺陷数量？
    - 缺陷走势？

系统测试的实施原因

1. 在测试中引入独立视角
2. 在测试中引入客户视角
3. 在测试中模拟客户使用环境

   正式、完备和现实的测试环境
4. 测试产品功能和非功能的问题
5. 建立对产品的信心
6. 分析和降低产品发布的风险
7. 保证满足所有需求，产品具备交付确认测试条件

系统测试的内容

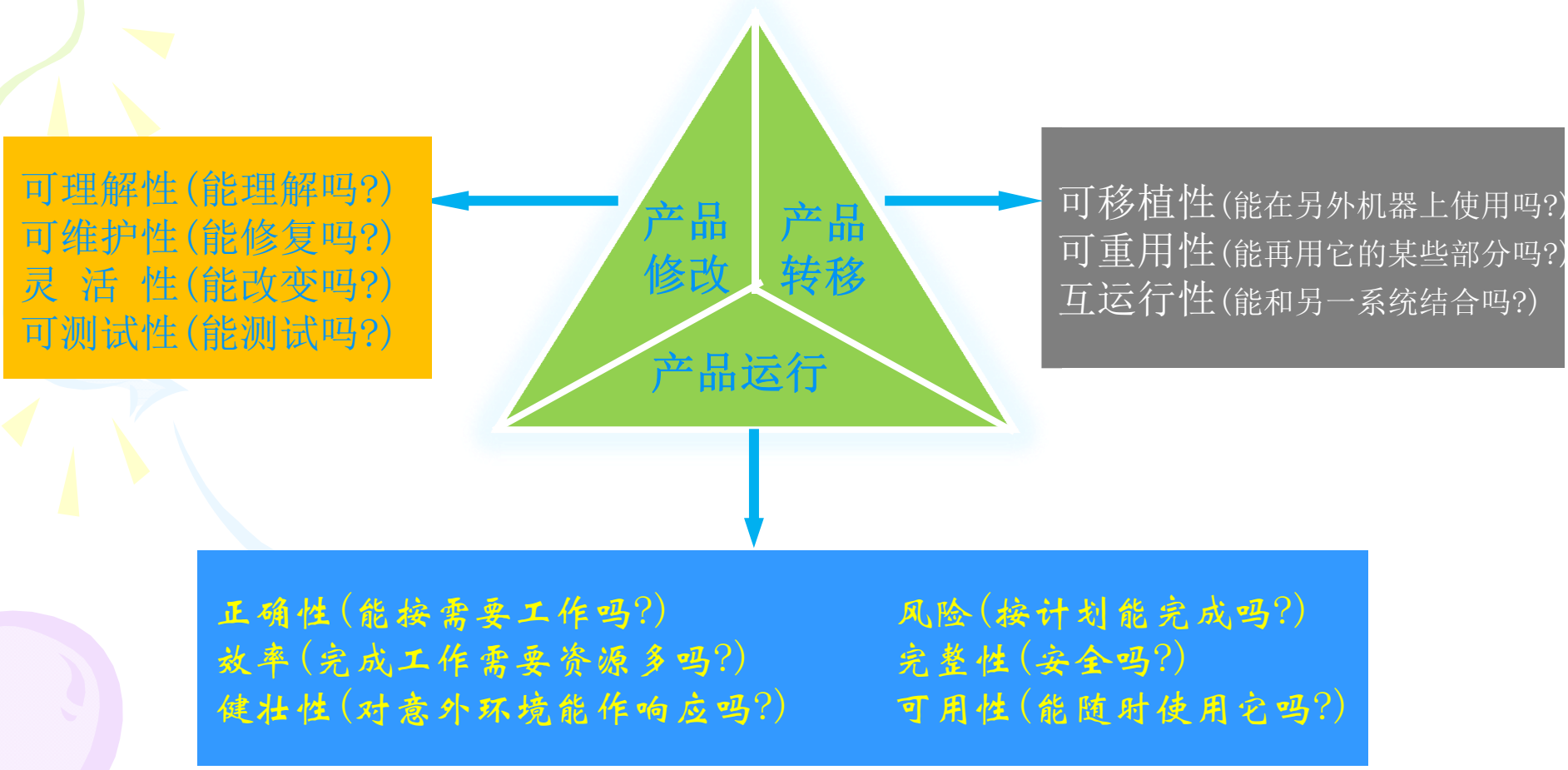
- 系统测试 = 功能测试 + 非功能测试

功能测试VS非功能测试

	功能测试	非功能测试
作用/关注点	验证产品功能和特性	验证产品质量因素
范围	所有测试阶段	系统测试
用例失败原因	代码缺陷	体系结构、设计和代码缺陷
预备知识	产品和领域	产品、领域、设计、体系结构、分析技能

系统测试是既测试产品功能也测试产品非功能的唯一测试阶段

软件质量属性


可理解性(能理解吗?)
可维护性(能修复吗?)
灵 活 性(能改变吗?)
可测试性(能测试吗?)
产品修改
产品转移
产品运行
可移植性(能在另外机器上使用吗?)
可重用性(能再用它的某些部分吗?)
互运行性(能和另一系统结合吗?)
正确性(能按需要工作吗?)
效率(完成工作需要资源多吗?)
健壮性(对意外环境能作响应吗?)
风险(按计划能完成吗?)
完整性(安全吗?)
可用性(能随时使用它吗?)

内容提要

1. 系统测试概述
2. 功能性测试
3. 非功能性测试

功能性测试

- 系统测试的内容
  - 功能符合性的测试
    - 采用黑盒测试技术，检验需求是否得到满足
    - 使用产品级的测试用例
  - 标准的测试
    - 是否符合行业标准（包括术语、流程等）
    - 是否符合法律法规
    - License等是否符合授权要求
    - ...
  - ……

Alpha测试和Beta测试

- 系统测试的阶段
  - **Alpha测试：**
    - 定义：在开发环境下进行的受控测试，尽量模拟用户的使用场景
    - 特点：由开发人员或独立测试人员、用户完成，但开发者会在场（例如，开发完成后，在开发机上展开测试）
  - **Beta测试：**
    - 定义：在实际使用环境下进行测试（可以把产品交给客户在应用环境下实测，收集反馈意见）
    - 特点：开发者通常不在现场

在**Alpha**测试达到一定程度后进行**Beta**测试，**Beta**测试时产品一般相对较为成熟，文档等支持齐备，随时待发布

3 非功能性测试

非功能性测试的常见内容

- 性能测试
- 兼容性测试
- 可用性测试
- 安全性测试
- 可靠性测试
- 国际化测试
- ……

不要事后才考虑到非功能性测试

- 非功能性问题可能难以修复！
  - 非功能性缺陷常常与整体设计相关

- 应在需求阶段就关注非功能性需求

3.1 性能测试

通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行评估

北京奥运订票网站瘫痪

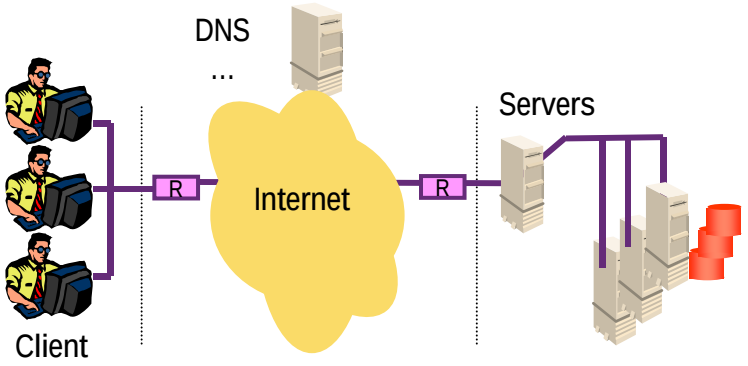
事故过程：订票系统半小时即瘫痪

1、淘宝被挤爆了

昨晚朋友圈截图。有钱花不出去，在线等，挺急的~

性能测试

- 三个方面
  - 应用在客户端性能的测试：考察客户端应用的性能，测试的入口是客户端
    - 并发性能测试：重点
    - 疲劳强度测试
    - 大数据量测试
    - 速度测试等
  - 应用在网络上性能的测试
    - 网络应用性能监控
    - 网络应用性能分析
    - 网络应用性能预测
  - 应用在服务器端性能的测试



并发性能测试

- 并发性能测试的过程是一个负载测试和压力测试的过程，即逐渐增加负载，直到系统的瓶颈或者不能接受的性能点，通过综合分析事务执行指标和资源监控指标来确定系统并发性能的过程。

并发性能测试

- 负载测试（**Load Testing**）
  - 确定各种工作负载下系统的性能，测试当负载逐渐增加时，系统组成部分的相应指标项，如吞吐量、响应时间、CPU负载、内存使用等，来评估系统的性能表现。(能接受情况下的性能)
- 压力测试（**Stress Testing**）
  - 通过确定一个系统的瓶颈或者不能接受的性能点，来获得系统能提供的最大服务级别的测试

例如，针对一个网站进行测试

- 模拟10到50个用户 → 常规性能测试
- 用户逐渐增加到1000乃至上万 → 负载测试
- 测试重负荷下，何时系统的响应会退化或失败 → 压力测试

并发性能测试的目的

- 主要三个方面
  - 以真实业务为依据，选择代表性的、关键的业务操作设计测试案例，以评价系统的当前性能
  - 当扩展应用程序功能，或者新的应用程序将被部署时，帮助确定系统是否还能处理期望的用户负载，以预测系统的未来性能
  - 通过模拟成百上千个用户，重复执行和运行测试，可以寻找到性能瓶颈，进行并优化和调整

举例：电信计费软件测试

- 每月20日左右是市话交费的高峰期，全市几千个收费网点同时启动。收费过程分两步：首先根据电话号码来查询当月产生费用，然后收取现金并将用户修改为已交费状态。
- 看起来简单，但当成百上千的终端同时执行，情况就大不一样了,如此众多的交易同时发生,对应用程序本身、操作系统、中心数据库服务器、中间件服务器、网络设备的承受力都是一个严峻的考验。
- 决策者不可能在发生问题后才考虑系统的承受力，预见软件的并发承受力，这是在软件测试阶段就应该解决的问题。

很多因素都会影响性能

- Windows下能同时启动多少程序？
- 一个文件夹下能保存多少个文件？
- 能同时打开多少个文件？
- 一个程序最多可以分配多大内存？
- 路由器最大网速能到多大？
- 一个服务器程序能同时接受多少个链接？
- 数据库能存储多少数据，每个记录能有多大？

并发性能测试前的准备工作

- 规划和配置测试环境
  - 与用户环境保持一致
  - 单独运行、避免其他软件的干扰
  - 能达到测试执行的技术需求
  - 能得到正确、易理解的测试结果
  - 稳定、可重复
- 选择测试工具
  - 商用工具：**LoadRunner**
  - 开源免费工具：**JMeter**
- 准备测试数据
  - 尽量接近真实
  - 已通过功能性测试

并发性能测试的主要指标

- 主要指标包括事务处理性能指标、资源监控等
  - 事务(Transaction)处理指标：反映业务的应用情况
    - 事务成功率
    - 每分钟(秒)事务数
    - 事务响应时间
      - 最小、平均、最大
      - 标准差：事务处理服务器响应的偏差，值越大，偏差越大
  - 资源监控：反映业务背后的资源使用情况
    - 虚拟并发用户数
    - 每秒点击次数
    - 每秒下载页面数

测试用例表格

功能	
目的	
方法	

并发用户数与事务执行情况															
并发用户数	事务平均响应时间			事务最大响应时间			平均每秒事务数			事务成功率			每秒点击率	平均流量（字节/秒）
	业务1	业务2	业务3	业务1	业务2	业务3	业务1	业务2	业务3	业务1	业务2	业务3		
20														
25														
30														
35														
40														
45														
50														

疲劳强度测试

- 采用系统稳定运行情况下能够支持的最大并发用户数，持续执行一段时间业务，通过综合分析事务执行指标和资源监控指标来确定系统处理最大工作量的性能，分析系统的稳定性.

- 如出现错误导致测试不能成功执行，则及时调整测试指标，例如降低用户数、缩短测试周期等。

测试用例表格

极限名称	“最大并发用户数量”		
前提条件			
测试日期			
输入/动作	测试总时间	并发持续时间	服务器情况
10个并发用户操作			
20个并发用户操作			
……			

大数据量测试

- 例如：测试数据库中已有**10**万个账号时的用户登录情况
  - 数据量非常大时容易出现响应迟钝，假死等情况
- 测试方式分为两种
  - 针对某些系统存储、传输、统计查询等业务进行大数据量的**独立数据量测试**
  - 与压力测试、负载测试、疲劳测试相结合的**综合数据量测试**（数据库性能测试）
- 关键是测试数据的准备，可依靠工具实现

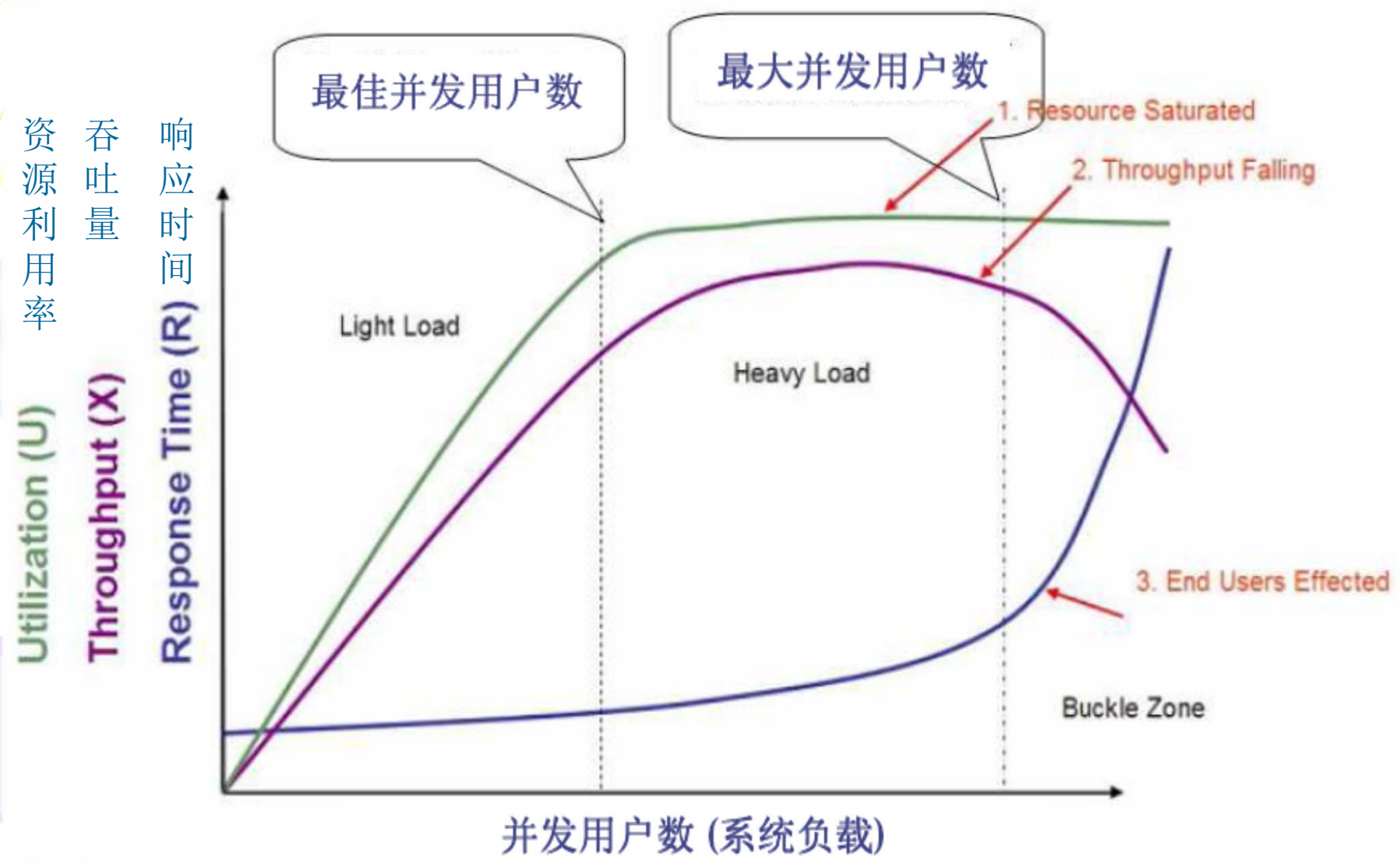
大数据量测试

- 用产品级数据库进行性能测试
  - 有助于及早暴露问题
    （例如，不要用仅有几条记录的数据库来进行测试）
- 获得数据的方式
  - 从客户处获得
  - 模拟方式生成
- 大规模数据不易获得时可在一定测试数据的基础上估算数据量变大后的影响

性能测试实例：“新华社多媒体数据库 V1.0”性能测试

- 目的
  - 模拟多用户并发访问新华社多媒体数据库，执行关键检索业务，分析系统性能。
- 重点
  - 针对系统并发压力负载较大的主要检索业务，进行并发测试和疲劳测试，系统采用B/S运行模式
- 并发测试
  - 设计了特定时间段内分别在中文库、英文库、图片库中进行单检索词检索、多检索词检索、混合检索等业务操作的并发测试案例，从小到大增大负载。
- 疲劳测试
  - 疲劳测试案例为在中文库中并发用户数200，进行测试周期约8小时的持续单检索词检索。

测试结论


最佳并发用户数
最大并发用户数
1. Resource Saturated
2. Throughput Falling
3. End Users Effected
Light Load
Heavy Load
Buckle Zone
资源利用率
Utilization (U)
吞吐量
Throughput (X)
响应时间
Response Time (R)
并发用户数 (系统负载)

测试结论

- 在新华社机房测试环境和内网测试环境中，**100M**带宽下，针对规定的各并发测试案例，系统能够承受并发用户数为200的负载压力，最大每分钟事务数达到78.73，运行基本稳定，但随着负载压力增大，系统性能有所衰减。
- 系统能够承受200并发用户数持续周期约8小时的疲劳压力，基本能够稳定运行。
- 对UNIX系统、Oracle和Apache资源的监控表明，系统资源能够满足上述并发和疲劳性能需求，且系统硬件资源尚有较大利用余地。
- 并发用户数超过200时，监控到HTTP 500、connect和超时错误，且Web服务器报内存溢出错误，系统应进一步提高性能，以支持更大并发用户数。[压力测试]
- 建议进一步优化软件系统，充分利用硬件资源，缩短事务响应时间。

网络上性能的测试

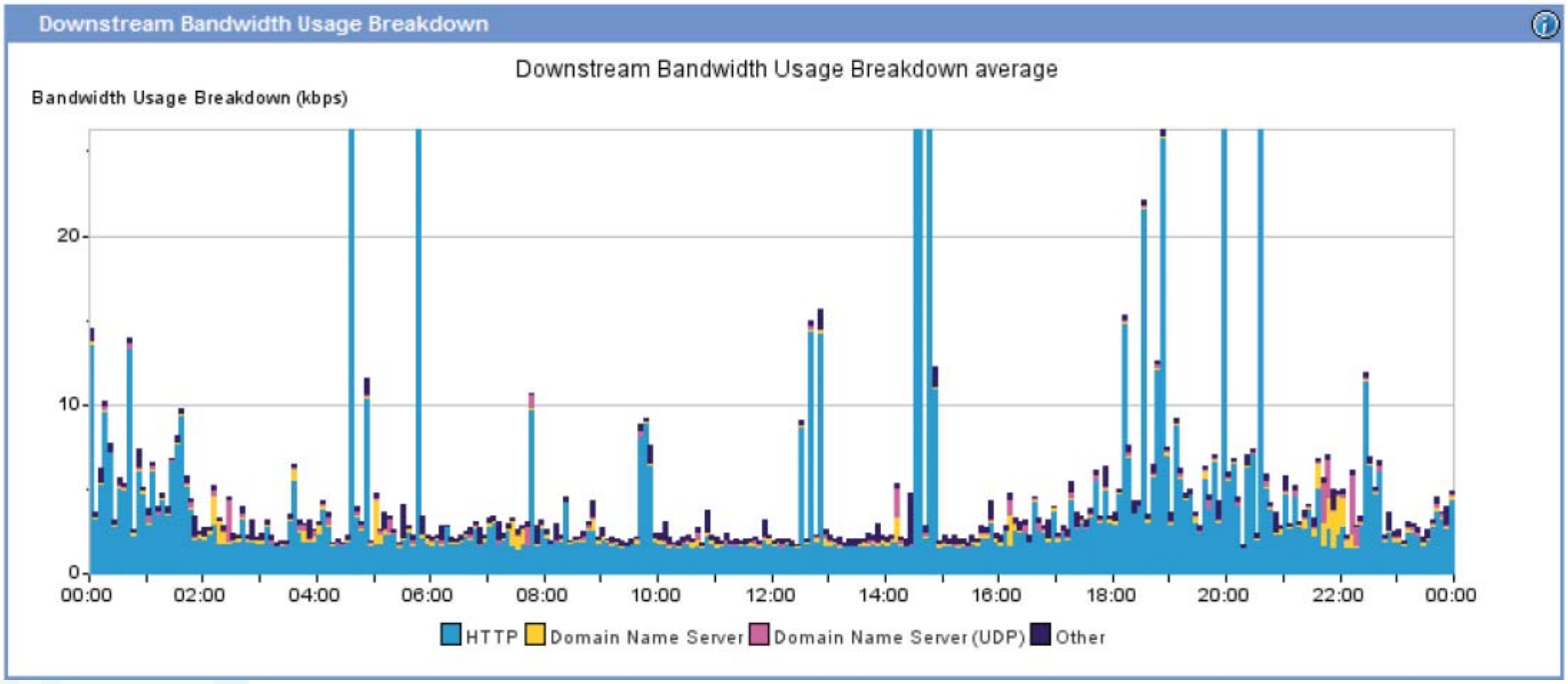
- 网络应用性能监控

  系统试运行之后，需要及时准确地了解网络上正在发生的事情：

  - 什么应用在访问网络；
  - 多少PC正在访问LAN或WAN；
  - 哪些应用程序占用大量带宽
  - 哪些用户产生了最大的网络流量
  - 哪些应用程序导致系统瓶颈或资源竞争？

- 可利用Network Vantage等工具进行网络应用性能监控

Downstream Bandwidth Usage Breakdown



Link View Scorecard: Applications - 10/20/08 12:00 EDT

Higher Level

Link: FastEthernet0/0.100 - ;FastEthernet0/0.100/Layer 2 Virtual LAN using 802.1Q

Network View | Application View | Link View | Reporting Groups | Scorecard | Usage | Performance

Monday, 10/20/08 << Previous Day

Page: 1 2 3 4 5 6 7 >> (Number of Entries: 70)

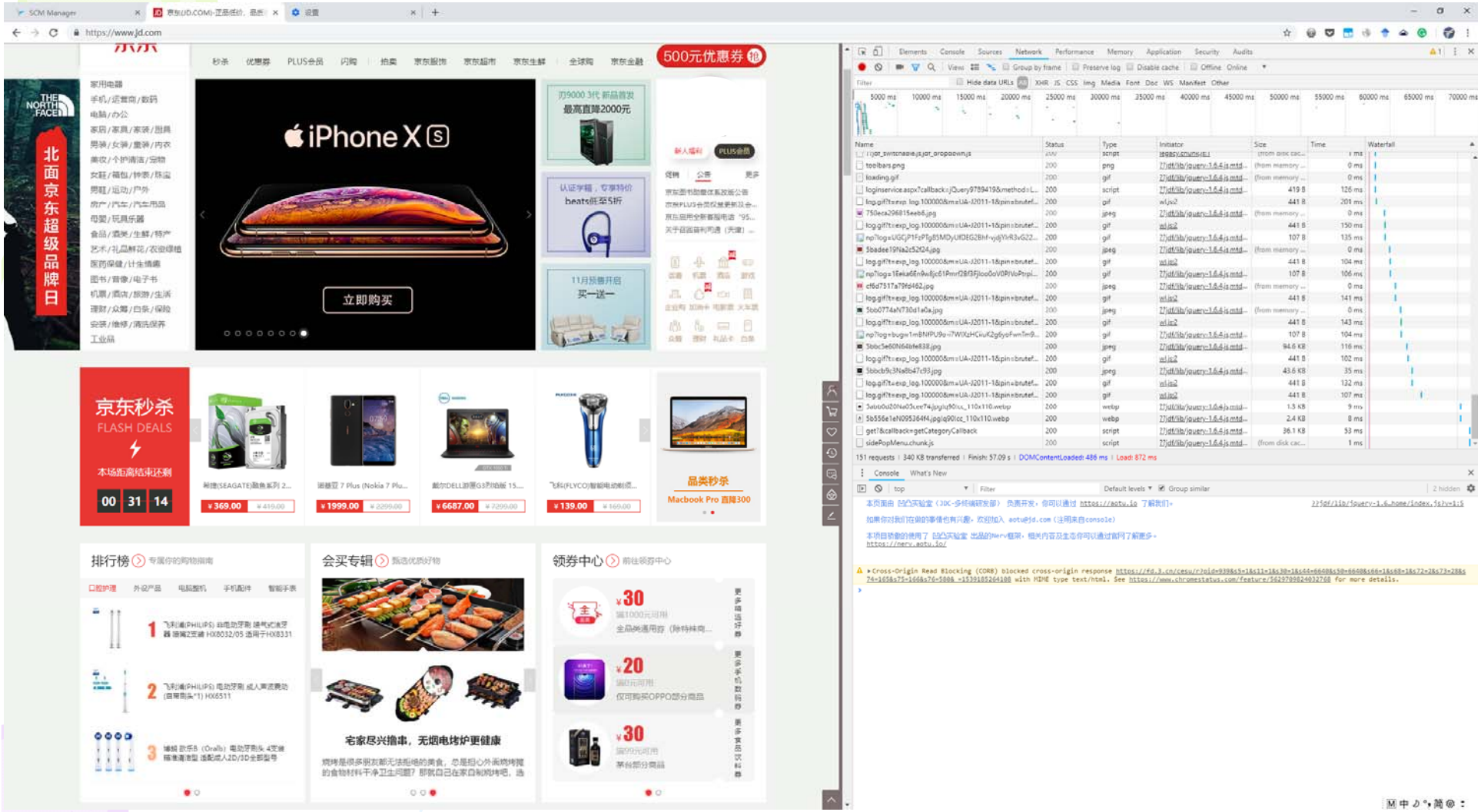
Link View: Applications

Find: in Application Find Reset

Application	Type of Service	Analyzer	Usage					
			Hosts (near end)	Hosts (far end)	Active Hosts	Outgoing Volume	Incoming Volume	Volume
TCP	Default (TOS: 0b:0h:0d)	TCP	2	2	2	3.89 GB	4.2 GB	8.09 GB
SIP (UDP)	Default (TOS: 0b:0h:0d)	UDP	4	3	4	1.35 GB	1.35 GB	2.7 GB
UDP (UDP)	Default (TOS: 0b:0h:0d)	UDP	39	74	87	1.18 GB	1.18 GB	2.36 GB
GRE (IP)	Default (TOS: 0b:0h:0d)	NetFlow	1	1	2	1.45 GB	0B	1.45 GB
HTTP	Default (TOS: 0b:0h:0d)	TCP	24	2	26	899 MB	140 MB	1.01 GB
MS-DS	Default (TOS: 0b:0h:0d)	TCP	56	20	70	429 MB	534 MB	963 MB
HTTPS	Default (TOS: 0b:0h:0d)	TCP	4	3	7	63.3 MB	169 MB	232 MB
SSH Secure Shell	Default (TOS: 0b:0h:0d)	TCP	10	4	12	12.1 MB	156 MB	168 MB
NetBIOS Session Srvc	Default (TOS: 0b:0h:0d)	TCP	46	15	53	95.2 MB	48.1 MB	143 MB
RADIUS	Default (TOS: 0b:0h:0d)	TCP	12	12	13	39.4 MB	39.4 MB	78.8 MB

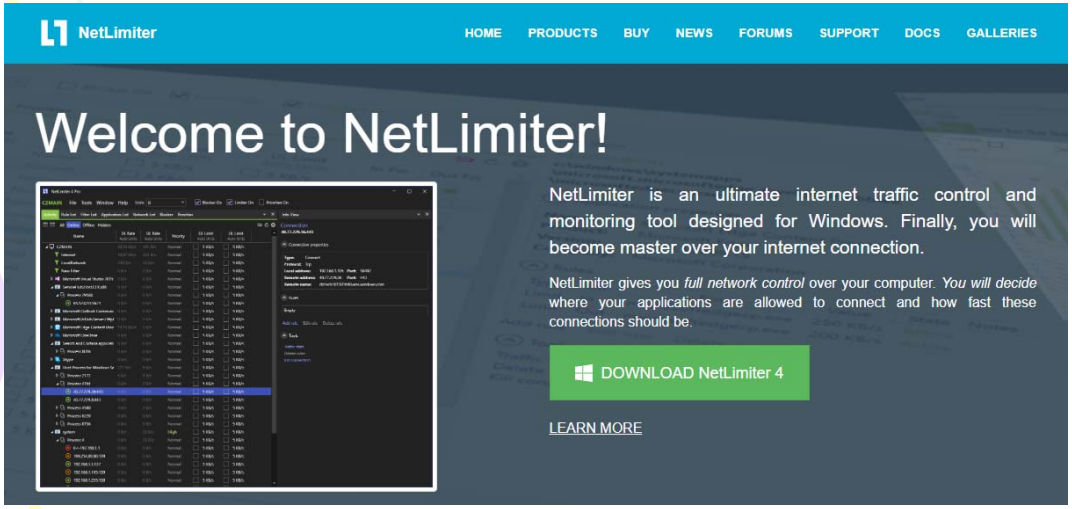
Page: 1 2 3 4 5 6 7 >> (Number of Entries: 70)

用Chrome浏览器进行监控



网络上性能的测试

- 网络应用性能分析
  - 分析网络带宽、延迟、负载（并发用户数）和端口的变化是如何影响用户的响应时间的；
  - 定位性能问题的根源是在客户端、服务器、应用程序还是网络。
  - 检查客户端是否对数据库服务器运行了不必要的请求
  - 检查应用服务器是否花费了不可接受的时间联系数据库服务器
  - ……



可用NetLimit工具限制网速测试应用在不同网速下的表现

网络上性能的测试

- 网络预测
  - 考虑到系统未来发展的扩展性，预测网络流量的变化、网络结构的变化对用户系统的影响非常重要

- 网络预测分析工具**PREDICTOR**
  - 功能：网络容量规划、离线测试网络、网络失效和容量极限分析、预测网络设备迁移和网络设备升级对整个网络的影响
  - 作用：可帮助用户及时升级网络，避免因关键设备超过利用阀值导致系统性能下降；发现哪个网络设备需要升级，减少网络延迟、避免网络瓶颈；根据预测的结果避免不必要的网络升级。

应用在服务器上性能的测试

- 对于应用在服务器上性能的测试，可以采用工具监控，也可以使用系统本身的监控命令.

- 测试的目的是实现服务器设备、服务器操作系统、数据库系统、应用在服务器上性能的全面监控。

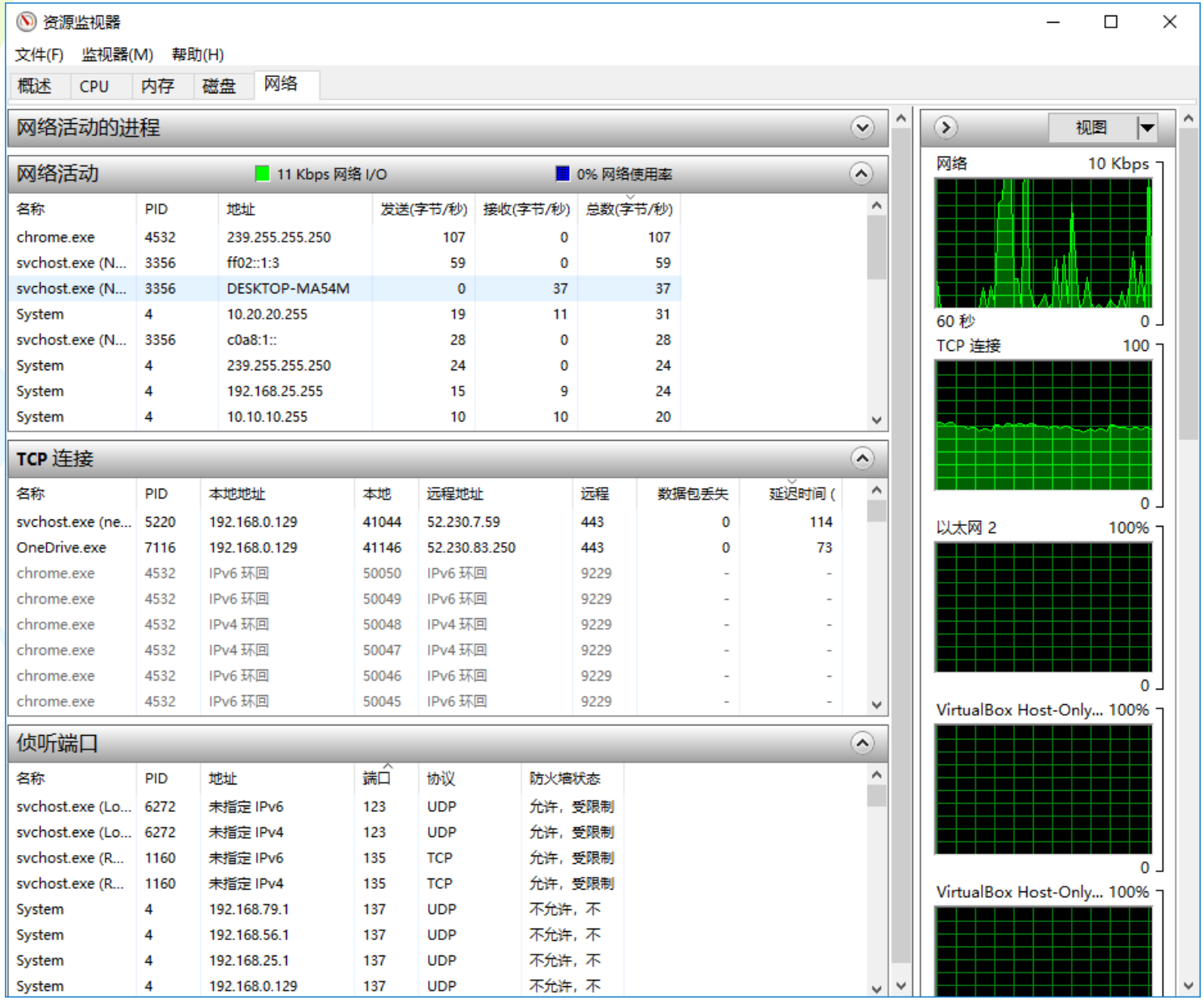
Linux资源监控命令/工具

ps	uptime	pidstat
pstree	ifuser	iostat
pidof	lsof	iotop
top	mpstat	watch
free	vmstst	sar

UNIX操作系统资源监控

度量	描述
Average load	上一分钟同时处于“就位”状态的平均进程数
Collision rate	每秒钟在以太网上检测到的冲突数
Context switches rate	每秒钟在进程或线程之间的切换次数
CPU utilization	CPU的使用时间百分比
Disk rate	磁盘传输速率
Incoming packets error rate	接收以太网数据包时每秒钟接收到的错误数
Incoming packets rate	每秒钟传入的以太网数据包数
Interrupt rate	每秒内的设备中断数
Outgoing packets errors rate	发送以太网数据包时每秒钟发送的错误数
Outgoing packets rate	每秒钟传出的以太网数据包数
Page-in rate	每秒钟读入到物理内存中的页数
Page-out rate	每秒钟写入页面文件和从物理内存中删除的页数
Paging rate	每秒钟读入物理内存或写入页面文件中的页数
Swap-in rate	正在交换的进程数
Swap-out rate	正在交换的进程数
System mode CPU utilization	在系统模式下使用CPU的时间百分比
User mode CPU utilization	在用户模式下使用CPU的时间百分比

用Windows资源监视器



LoadRunner简介

- 轻松创建虚拟用户
  - LoadRunner能够生成虚拟用户，模拟真实用户的业务操作行为。它先记录下业务流程(如机票预定)，然后将其转化为测试脚本（可手工优化）。利用虚拟用户，可以在Windows，UNIX或Linux 机器上同时产生成千上万个用户访问。

- 创建真实的负载
  - LoadRunner提供了一个互动的环境，在其中既能建立起持续且循环的负载，又能管理和驱动负载测试方案。而且，可以利用它的日程计划服务来定义用户在什么时候访问系统以产生负载。

LoadRunner简介

- 实时监测器
  - LoadRunner内含实时监测器，在负载测试过程的任何时候，都可以观察到应用系统的运行性能。监测器可实时显示交易性能数据（如响应时间）和其它系统组件包括application server, web server，网路设备和数据库等的实时性能。

- 分析结果以精确定位问题所在
  - 测试完毕后，LoadRunner汇总所有的测试数据，并提供高级的分析和报告工具，以便迅速查找到性能问题并追溯原由。使用LoadRunner 的Web 交易细节监测器，可了解到将所有的图象、框架和文本下载到每一网页上所需的时间。

手动测试

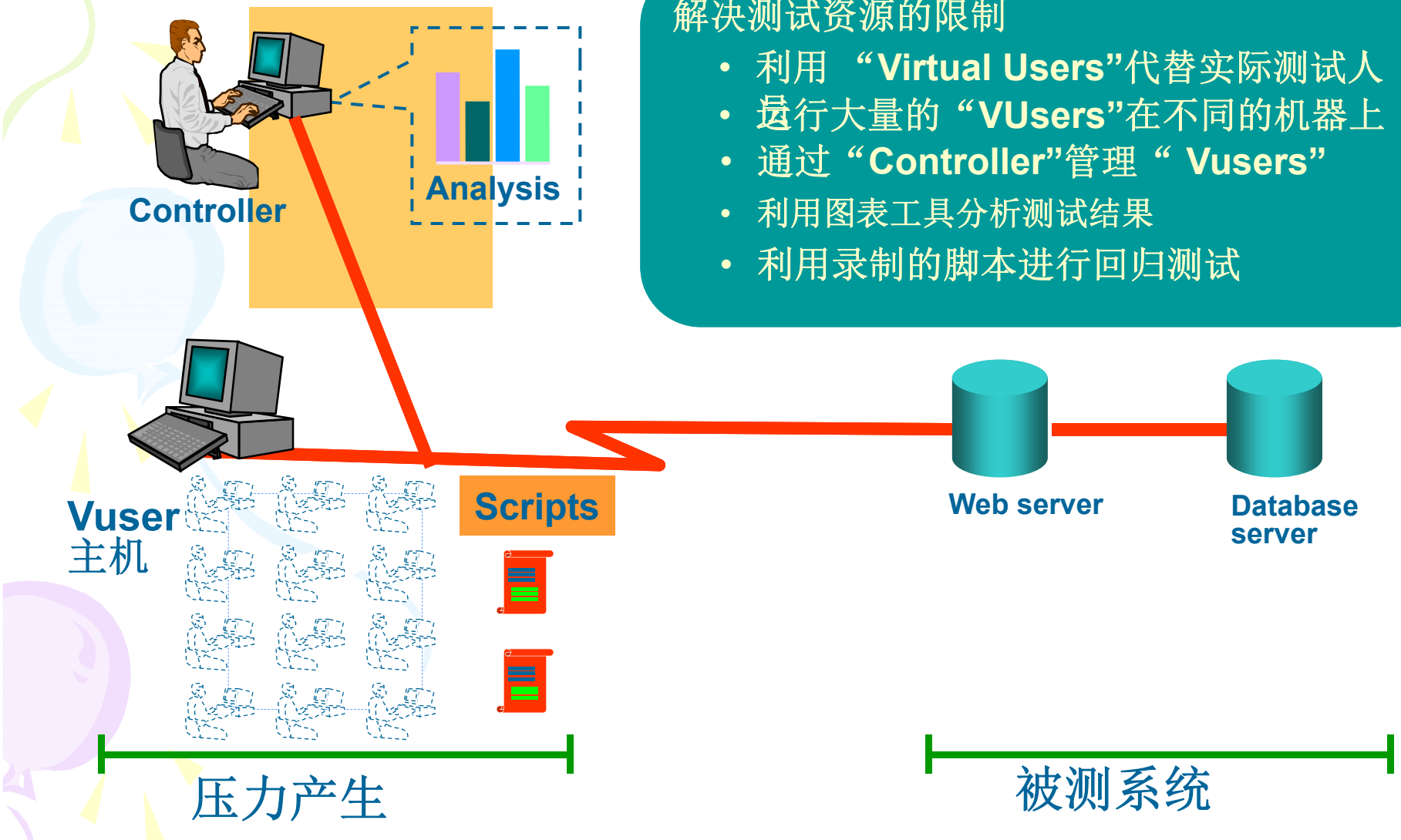
是否有足够的测试资源?
- 测试人员
- 客户机

如何调度和同步测试用户?

如何搜集和分析测试结果?

如何完成回归测试?

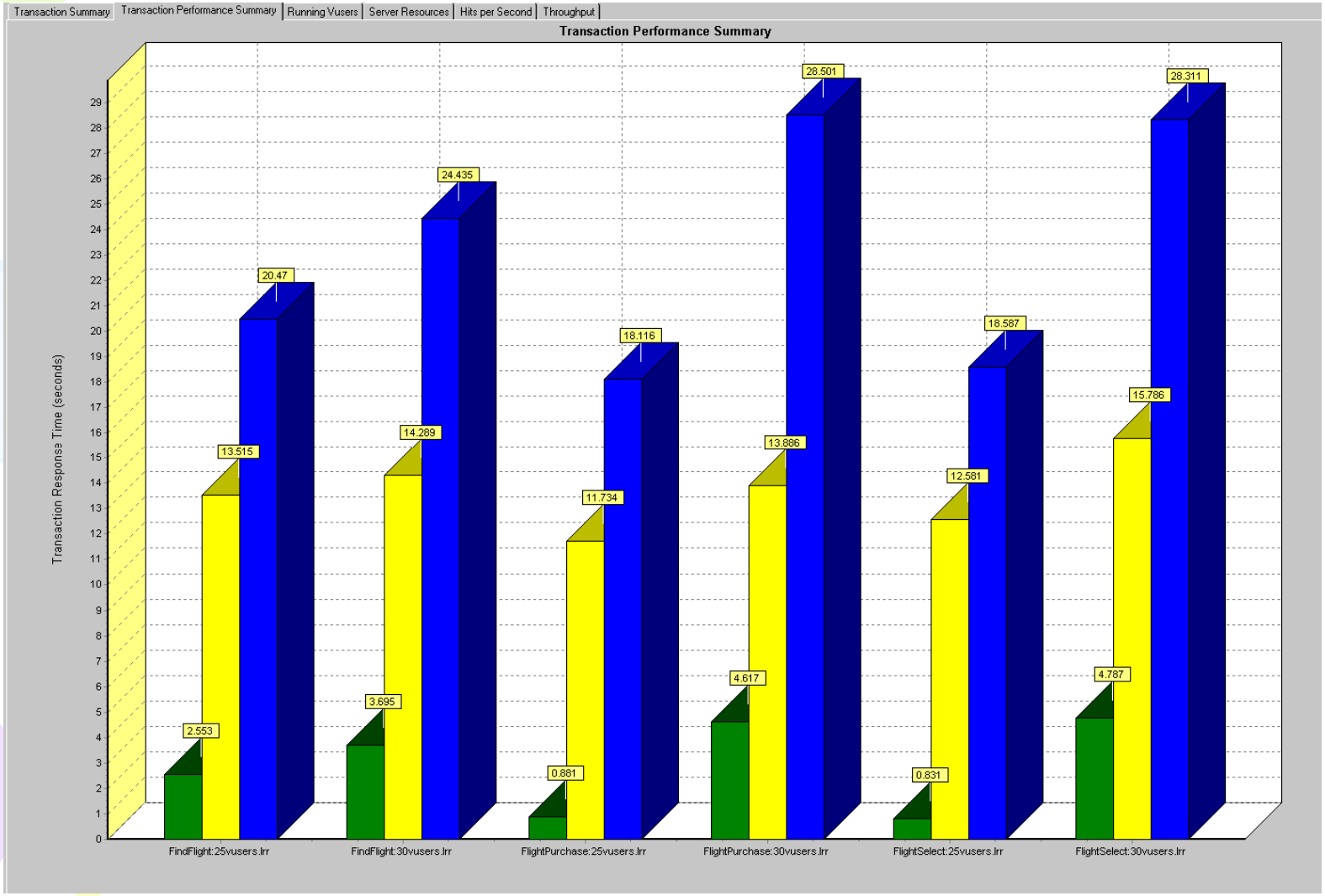
LoadRunner 的解决方案



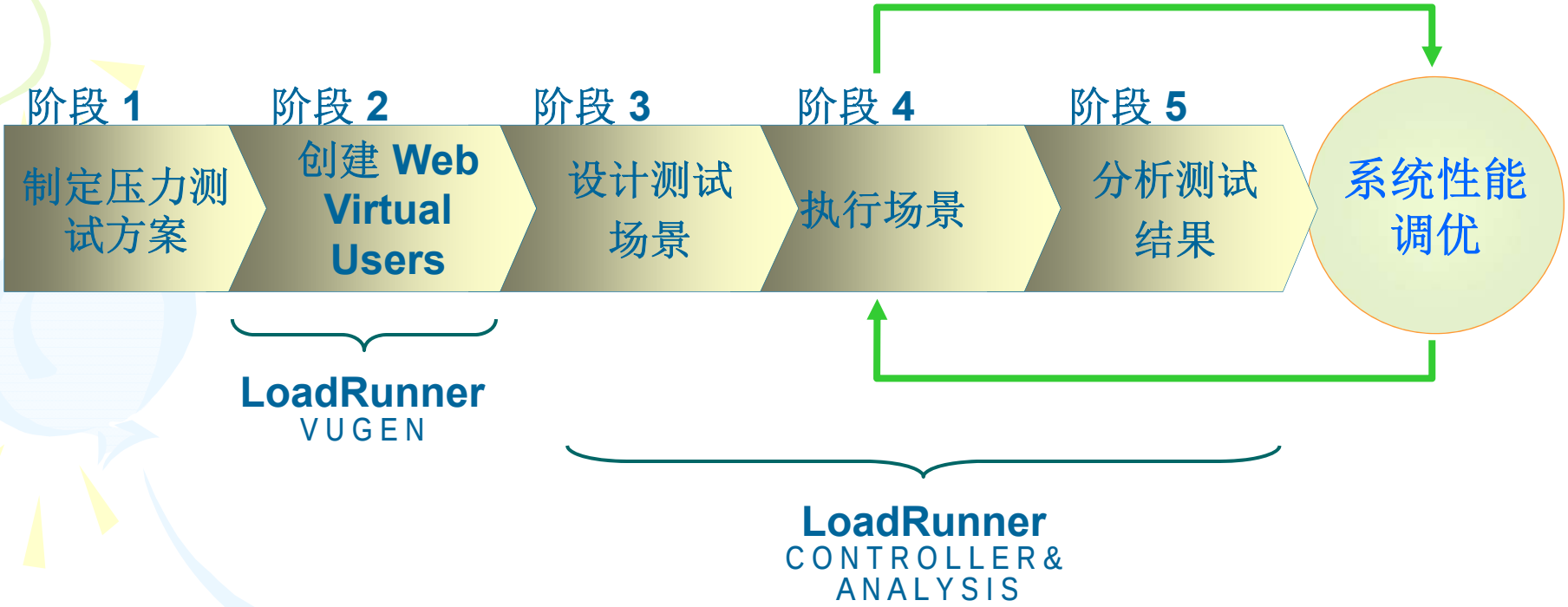
解决测试资源的限制

- 利用 “Virtual Users”代替实际测试人员
- 运行大量的“VUsers”在不同的机器上
- 通过“Controller”管理“ Vusers”
- 利用图表工具分析测试结果
- 利用录制的脚本进行回归测试

LoadRunner 的解决方案



LoadRunner 工作流程



- 确定参与性能测试的具体业务
- 确定总体的负载规模、压力等

- 录制反映业务用法的测试脚本，用脚本来模拟真实用户行为

- 搭配测试脚本
- 参数化测试脚本
- 确定负载的同步方案
- 确定负载和压力的具体变化曲线
- 分配测试主机

invent
LoadRunner

基于Jmeter的性能测试

THE APACHE SOFTWARE FOUNDATION

APACHE JMeter™

About
- Overview
- License

Download
- Download Releases
- Release Notes

Documentation
- Get Started
- User Manual
- Best Practices
- Component Reference
- Functions Reference
- Properties Reference
- Change History
- Javadocs
- JMeter Wiki
- FAQ (Wiki)

Tutorials
- Distributed Testing
- Recording Tests
- JUnit Sampler
- Access Log Sampler
- Extending JMeter

Community
- Issue Tracking
- Security
- Mailing Lists
- Source Repositories
- Building and Contributing
- Project info at Apache
- Contributors

Foundation
- The Apache Software Foundation (ASF)
- Get Involved in the ASF
- Sponsorship
- Thanks

Apache JMeter™

Twitter github

The **Apache JMeter™** application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions.

What can I do with it?

Apache JMeter may be used to test performance both on static and dynamic resources, Web dynamic applications.
It can be used to simulate a heavy load on a server, group of servers, network or o[illegible] performance under different load types.

Apache JMeter features include:

- Ability to load and performance test many different applications/server/protoc
  - Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, ...)
  - SOAP / REST Webservices
  - FTP
  - Database via JDBC
  - LDAP
  - Message-oriented middleware (MOM) via JMS
  - Mail - SMTP(S), POP3(S) and IMAP(S)
  - Native commands or shell scripts
  - TCP
  - Java Objects
- Full featured Test IDE that allows fast Test Plan **recording (from Browsers**
- **Command-line mode (Non GUI / headless mode)** to load test from any Jav
- A complete and **ready to present dynamic HTML report**
- Easy correlation through ability to extract data from most popular response fo
- Complete portability and **100% Java purity**.
- Full **multi-threading** framework allows concurrent sampling by many threads separate thread groups.
- Caching and offline analysis/replaying of test results.
- **Highly Extensible core:**
  - Pluggable Samplers allow unlimited testing capabilities.
  - **Scriptable Samplers** (JSR223-compatible languages like Groovy and
  - Several load statistics may be chosen with **pluggable timers**.
  - Data analysis and **visualization plugins** allow great extensibility as well as personalization.
  - Functions can be used to provide dynamic input to a test or provide data manipulation.
  - Easy Continuous Integration through 3rd party Open Source libraries for Maven, Gradle and Jenkins.

How do I do it?

- Using JMeter to understand how to use it
- Component reference to have detailed information for every Test element
- Functions reference to have detailed information and examples for every function
- Properties reference for all properties that allow you to customize JMeter
- Javadoc API documentation
- JMeter FAQ (Wiki)
- JMeter Wiki
- Building JMeter and Add-Ons for advanced usage

3.2 兼容性测试

兼容性测试

- 检查软件在一个特定的硬件、软件、操作系统、网络环境下能否正常运行，检查软件之间能否正确地交互和共享信息，检查软件版本之间的兼容性问题

- 三类基本测试
  - 平台、设备兼容性：操作系统、硬件
  - 交叉兼容性：不同软件间兼容
  - 版本兼容性：兼容不同版本的数据、功能、配置等

交叉兼容性之：数据共享兼容性

- 应用程序间共享数据可增加软件功能，好的应用程序应支持并遵循公开的接口标准，允许用户与其它软件轻松地传输数据。

- 数据共享兼容的实现
  - 支持数据导入、导出
  - 剪切、复制、粘贴
  - 支持DDE（动态数据交换）和OLE（对象的连接和嵌入）等

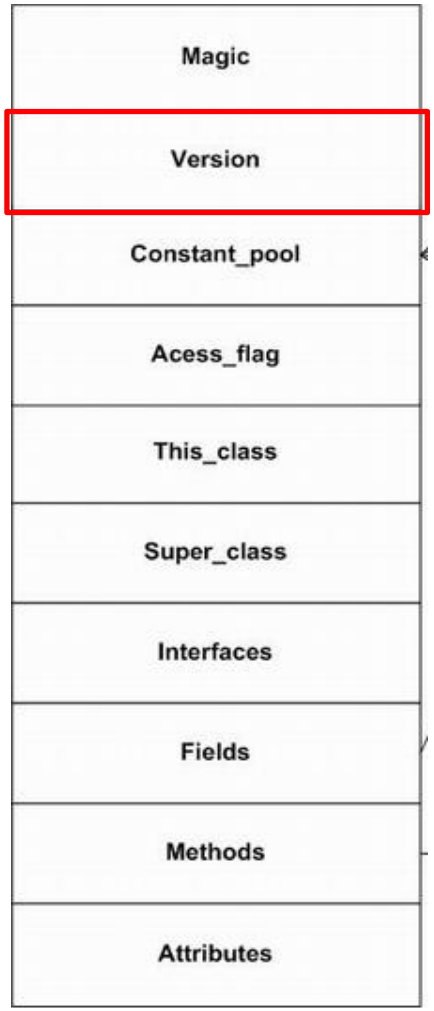
版本兼容性

- 向前兼容和向后兼容
  - 向前：兼容早期版本
  - 向后：兼容后期版本

为实现版本兼容，数据应标识版本，同时考虑扩展需要

e.g., .exe\.class\.pdf

.class文件版本信息



jdk	minor	major
1.0	45	3
1.1	45	3
1.2	46	0
1.3	47	0
1.4	48	0
1.5	49	0
1.6	50	0

确定恰当的测试配置和测试数据

- 测试构成待测软件运行环境的软硬件的多个版本
- 根据版本的重要性进行测试

  参考：流行程度　　发行时间　　厂商　　出现频率
- β测试中请用户帮忙
- 选择与最终用户使用环境相一致、具有代表性的测试配置集合，在开发过程中或者发行后，必须不断地更新测试用例
- 难以全部兼容，需求中应阐明兼容要求

确定恰当的测试用例和测试数据

- 可采用正交测试方法进行组合测试

例：有一Eclipse插件，标称支持Windows/Linux两种操作系统，Eclipse 3.6和Eclipse 4.0版，Java 1.5和Java 1.8。试采用正交测试方法对该插件的兼容性进行测试，并描述整个测试流程如何实施。

因子	水平
操作系统	Windows, Linux
Eclipse	3.6, 4.0
Java	1.5, 1.8

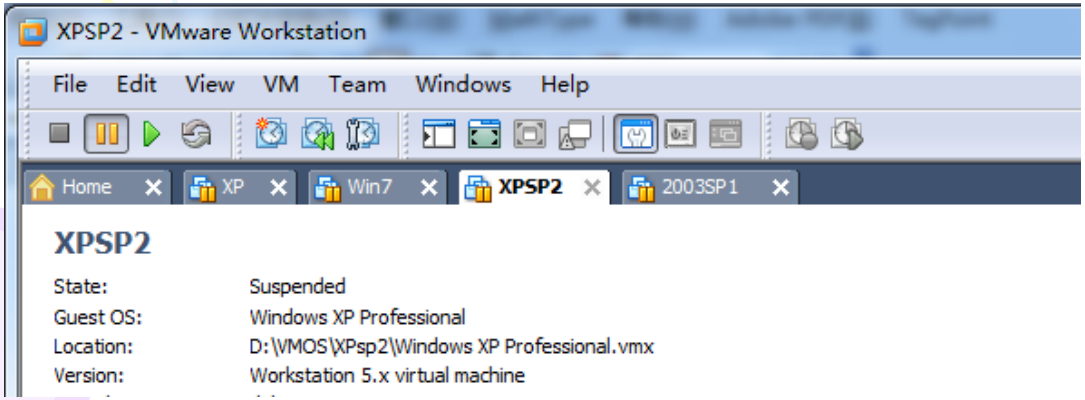
测试用例	操作系统	**Eclipse**	**Java**
1	Windows	3.6	1.5
2	Windows	4.0	1.8
3	Linux	3.6	1.8
4	Linux	4.0	1.5

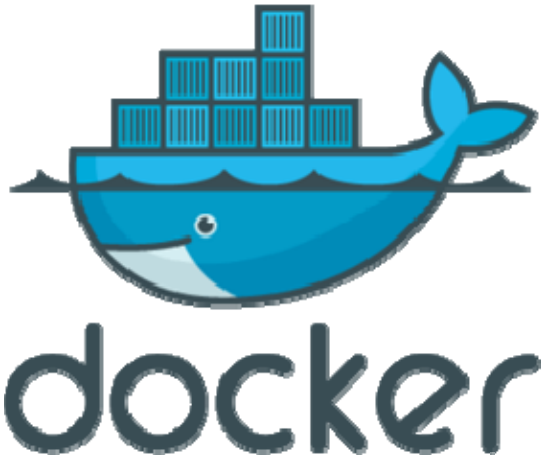
列号 / 试验号	**1**	**2**	**3**
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

$L_4(2^3)$ 正交表

兼容性测试环境的管理

- 测试环境的配置：
  1. 使用脚本运行时搭建和拆除环境 【省空间】
  2. 使用虚拟机和容器【快速简便】
     - VMWare, VirtualBox, Docker





3.3 可用性测试

可用性

- 是否可用

- 是否好用
  - 验证对应用程序有意向的用户能否和应用程序正确地进行交互，同时感到使用起来明确而方便。

界面漂亮≠可用性好

用户界面可用性测试

- 是否符合标准和规范
  - 《Macintosh Human Interface Guideline》
  - 《Microsoft Windows User Experience》

用户界面可用性测试

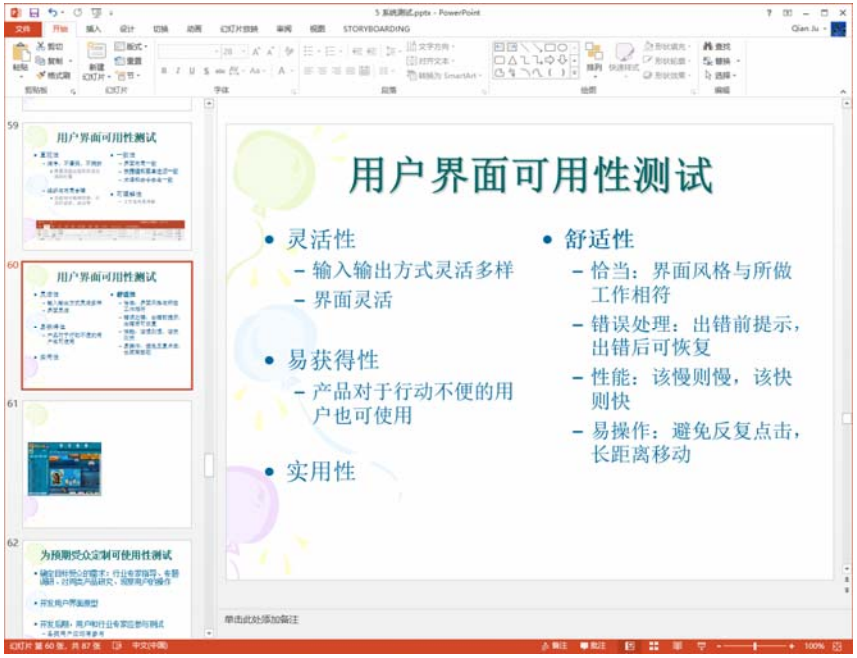
- 直观性
  - 洁净、不唐突、不拥挤
    - 所需功能出现在应该出现的位置
  - 组织与布局合理
    - 功能间可顺利切换，可及时放弃、退出等
- 一致性
  - 界面布局一致
  - 快捷键和菜单选项一致
  - 术语和命令命名一致
- 可理解性
  - 文字说明易理解

用户界面可用性测试

- 灵活性
  - 输入输出方式灵活多样
  - 界面灵活
- 易获得性
  - 产品对于行动不便的用户也可使用
- 实用性
- 舒适性
  - 恰当：界面风格与所做工作相符
  - 错误处理：出错前提示，出错后可恢复
  - 性能：该慢则慢，该快则快
  - 易操作：避免反复点击，长距离移动

恰当的界面


95棋牌游戏
首页
充值
保险箱
客服
下载
全部
新闻公告
用户登录 Login
奖品展示区


用户界面可用性测试
灵活性
输入输出方式灵活多样
界面灵活
易获得性
产品对于行动不便的用户也可使用
实用性
舒适性
恰当：界面风格与所做工作相符
错误处理：出错前提示，出错后可恢复
性能：该慢则慢，该快则快
易操作：避免反复点击，长距离移动

为预期受众定制可使用性测试

- 确定目标受众的需求：行业专家指导、专题调研、对同类产品研究、观察用户的操作
- 开发用户界面原型
- 开发后期，用户和行业专家应参与测试
  - 各类用户应均有参与
  - 系统能否满足专家和业余人士的使用？

3.4 安全性测试

验证应用程序的安全服务，
识别潜在安全缺陷的过程

安全性

- 控制安全性
  - 访问控制是否可能被绕过
  - 是否对输入进行了充分的验证：**SQL Injection、XSS、Stack Smash、Double Free Heap**等大多数漏洞都是由于这个原因
- 数据安全性
  - 保密性：信息不会泄露给非授权用户、实体或者进程
  - 完整性：信息在存储/传输过程中不被修改、破坏或丢失
  - 可用性：当需要时应能存取所需的信息
  - 不可否认性
  - 可控性：对信息的传播及内容具有控制能力

安全性测试应考虑的常见内容

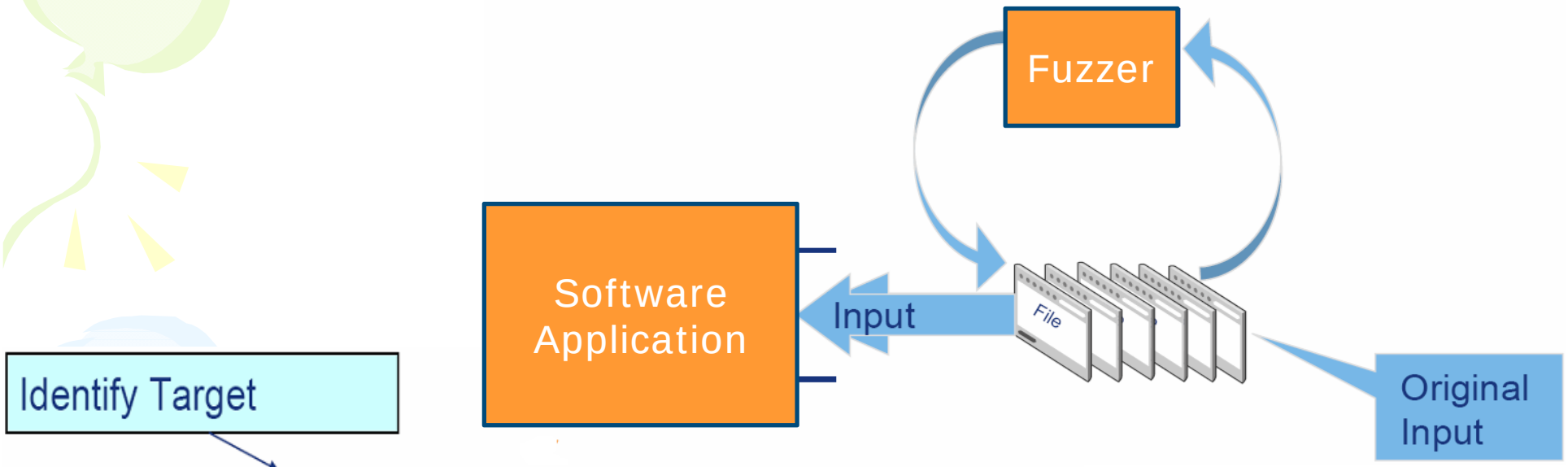
- 常见的方面
  - 用户管理和访问控制
    - 什么人可登录；什么人具有什么操作权限；
    - 是否可能绕开登录
  - 通信和数据加密
    - 如客户私有数据的加密后放数据库
    - 用**SSL**加密浏览器和服务器之间传递的数据
    - 协议是否有漏洞
  - 数据备份和恢复
  - 第三方组件的安全性
    - 是否有数字签名
    - **SHA**、**MD5**是否和官方给出的相一致？
  - 输入数据检查

安全性测试的方法

- 功能测试：
  - 用户界面（异常输入）
  - 灰箱测试特定需求（用户名和密码的加密传输）
  - 分析数据表或者文件（隐私数据是否加密）
- 漏洞扫描
- 模拟攻击试验
- 侦听技术(Sniffer)
- 错误注入
- **Fuzz Testing**
- 静态代码审查
  - 特殊函数，如strcpy，strcat是否有漏洞

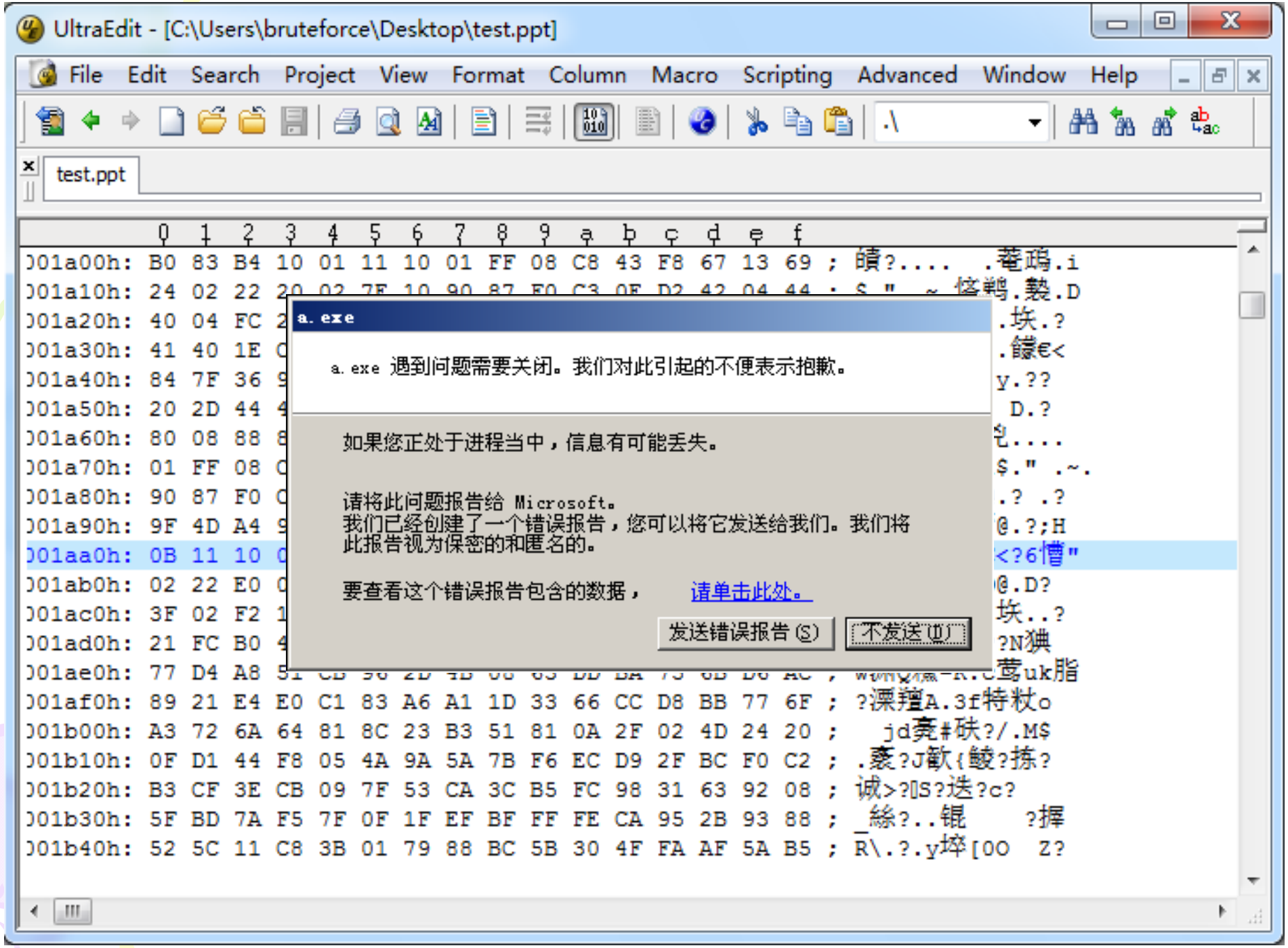
安全性要求特别高的系统，可以外购安全性测试

Fuzz Testing



- Identify Target
- Identify Inputs
- Generate Fuzzed Data
- Execute Fuzzed Data
- Monitor for Exceptions
- Determine Exploitability

- Network Protocols
- Files
- GUI
- Inter Process Communications
- ActiveX
- More


UltraEdit - [C:\Users\bruteforce\Desktop\test.ppt]
File Edit Search Project View Format Column Macro Scripting Advanced Window Help
test.ppt
a.exe
a.exe 遇到问题需要关闭。我们对此引起的不便表示抱歉。
如果您正处于进程当中，信息有可能丢失。
请将此问题报告给 Microsoft。
我们已经创建了一个错误报告，您可以将它发送给我们。我们将此报告视为保密的和匿名的。
要查看这个错误报告包含的数据，
请单击此处。
发送错误报告(S)
不发送(D)