

# 测试技术 – 常用测试

# 测试技术 – 常用测试

---

- ▶ 几种常见系统测试
- ▶ 用户文档测试
- ▶ 配置测试
- ▶ 本地化测试
- ▶ 易用性测试
- ▶ 基于应用服务器的测试
  - ▶ Client/Server 测试
  - ▶ 网站测试
  - ▶ 对Web进行压力测试
- ▶  $\alpha$  测试和  $\beta$  测试
- ▶ 实时系统测试
- ▶ 其他测试
- ▶ 调试

# 系统测试

---

## ▶ 以下分别简要说明几种系统测试：

### ▶ 性能测试、负载测试、压力测试

性能测试是为获取或验证系统性能指标而进行的测试。

负载测试是通过改变系统软件负载方式、增加负载等来发现系统软件中所存在的性能问题。主要测试软件系统的性能。

压力测试通常是在高负载情况下来对系统的稳定性进行测试，更有效地发现系统稳定性的隐患和系统在负载峰值的条件下功能隐患等。

目的虽不同，但方法类似，通常会用特定的测试工具，来模拟超常的数据量、负载等，监视系统的各项性能指标。

### ▶ 安全测试、可靠性测试

### ▶ 兼容性测试

# 系统测试

---

## ▶ 性能测试

性能测试常常与负载测试、压力测试结合进行，确定在各种工作负载下系统的性能，及系统能提供的最大服务级别的测试。并常常要求同时进行硬件和软件检测。

## ▶ 性能测试的目的：

为了验证系统是否达到用户提出的性能指标，同时发现系统中存在的性能瓶颈，起到优化系统的目的。

包括以下几个方面：评估系统的能力，识别体系中的弱点，系统调优，检测软件中的问题，验证稳定性（resilience）可靠性（reliability）等。

# 系统测试

---

## ▶ 性能测试的依据：

需求说明书中规定的性能。特别是对于实时系统或嵌入式系统。

如用户没有提出性能指标，则根据用户需求、测试设计人员的经验来设计各项测试指标。（需求+经验）

## ▶ 主要的性能指标：

通常，对软件性能的检测表现在以下几个方面：服务器的各项指标（CPU、内存占用率等），后台数据库的各项指标，网络流量，响应时间，并发性能，辅助存储区（如缓冲区，工作区的大小等），处理精度等。

为记录性能需要在系统中安装必要的量测仪表或是为度量性能而设置的软件（性能测试工具）或程序段。如：IBM Rational Performance Tester、HP Loadrunner、Compuware QALoad等。

# 系统测试

---

## ▶ 性能测试要点

- ▶ 测试环境应尽量与产品运行环境保持一致，应单独运行尽量避免与其他软件同时使用。
- ▶ 性能测试一般使用测试工具和测试人员编制测试脚本来完成。
- ▶ 性能测试的重点在于前期数据的设计与后期数据的分析。
- ▶ 性能测试的用例主要涉及到整个系统架构的问题，所以测试用例一旦生成，改动一般不大，所以做性能测试的重复使用率一般比较高。



# 性能测试的方法和技巧

---

## ▶ 两种负载类型

- ▶ “flat” 测试
- ▶ ramp-up测试

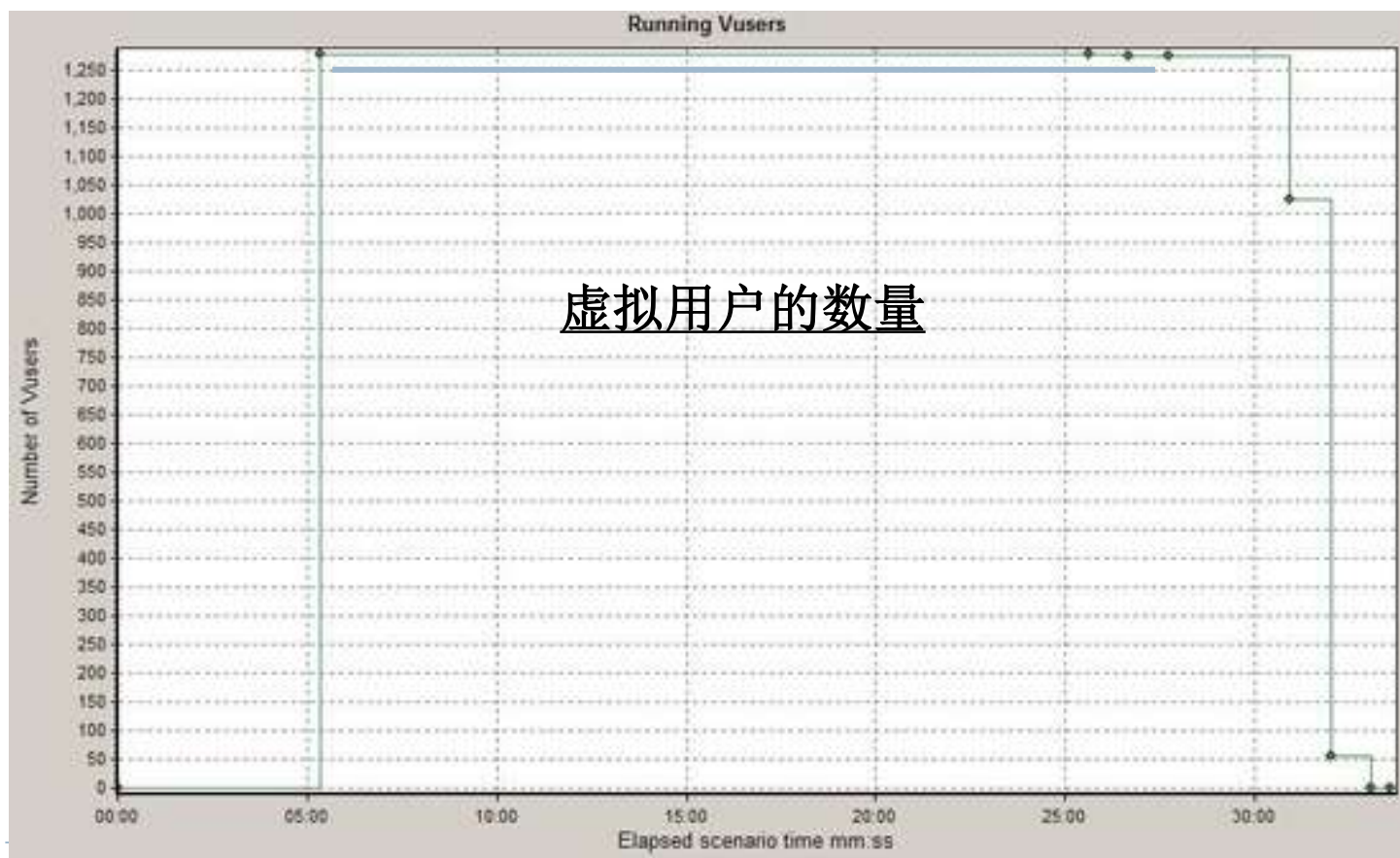
## ▶ 对于企业级的系统，性能测试的方法主要有：

- ▶ 基准测试
- ▶ 性能规划测试
- ▶ 渗入测试
- ▶ 峰谷测试



# 两种负载类型

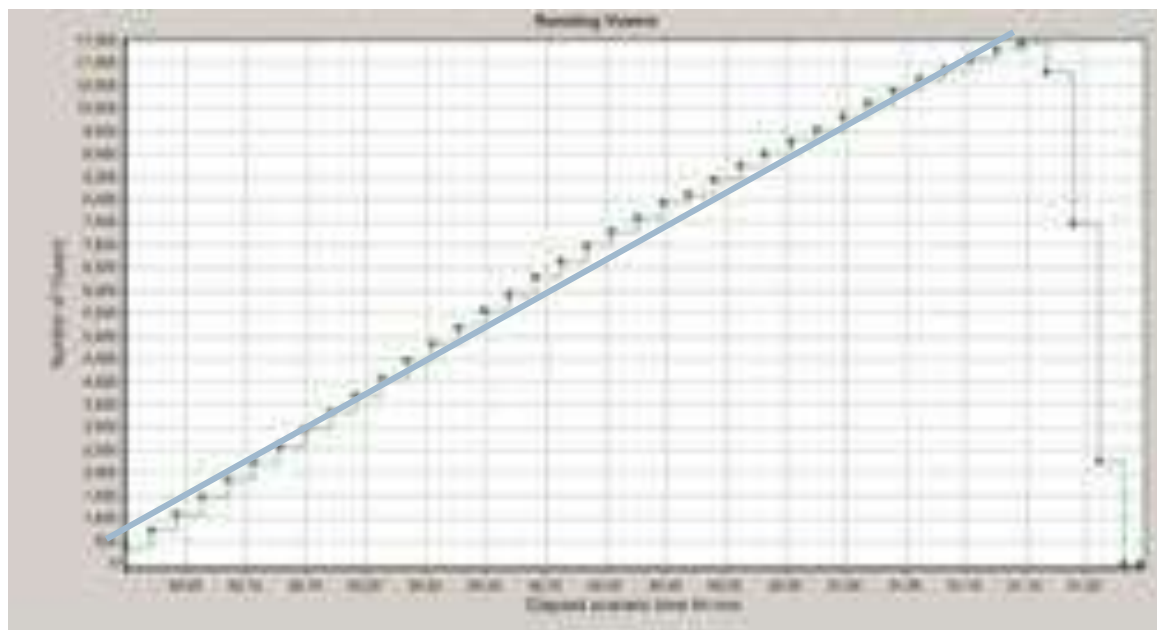
- ▶ “Flat” 测试: 一次加载所有负载, 然后在预定的时间段内持续运行, 然后取测试性能指标的平均值。





# 两种负载类型

- ▶ Ramp-up测试: 负载是交错上升的 (每几秒增加一些新负载)。ramp-up测试不能产生精确和可重现的平均值, 这是因为由于负载的增加是每次一部分, 系统的负载在不断地变化。其优点是, 可以看出随着系统负载的改变, 测量值是如何改变的→据此选择要运行的flat测试的范围。

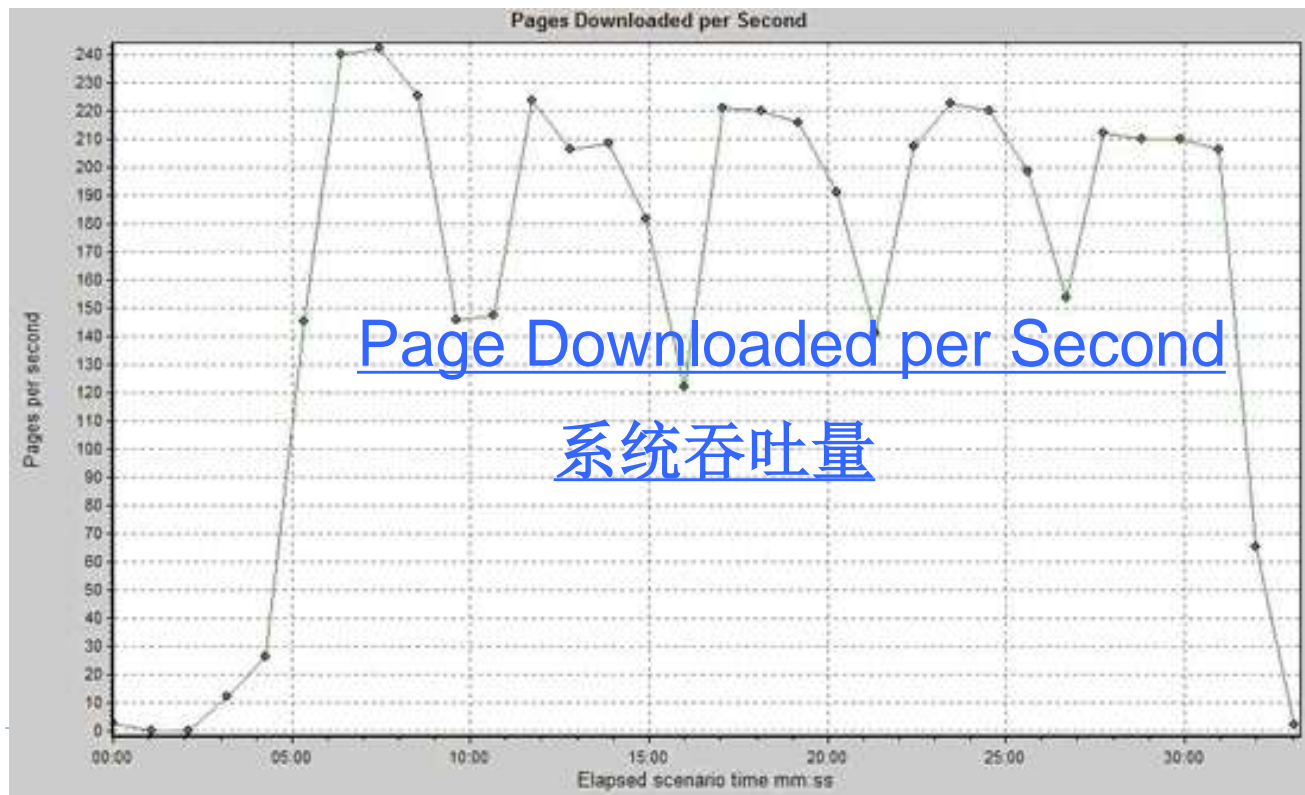


# Flat测试 “波动” 效应

## ► Flat测试的问题是系统会遇到“波动”效应。

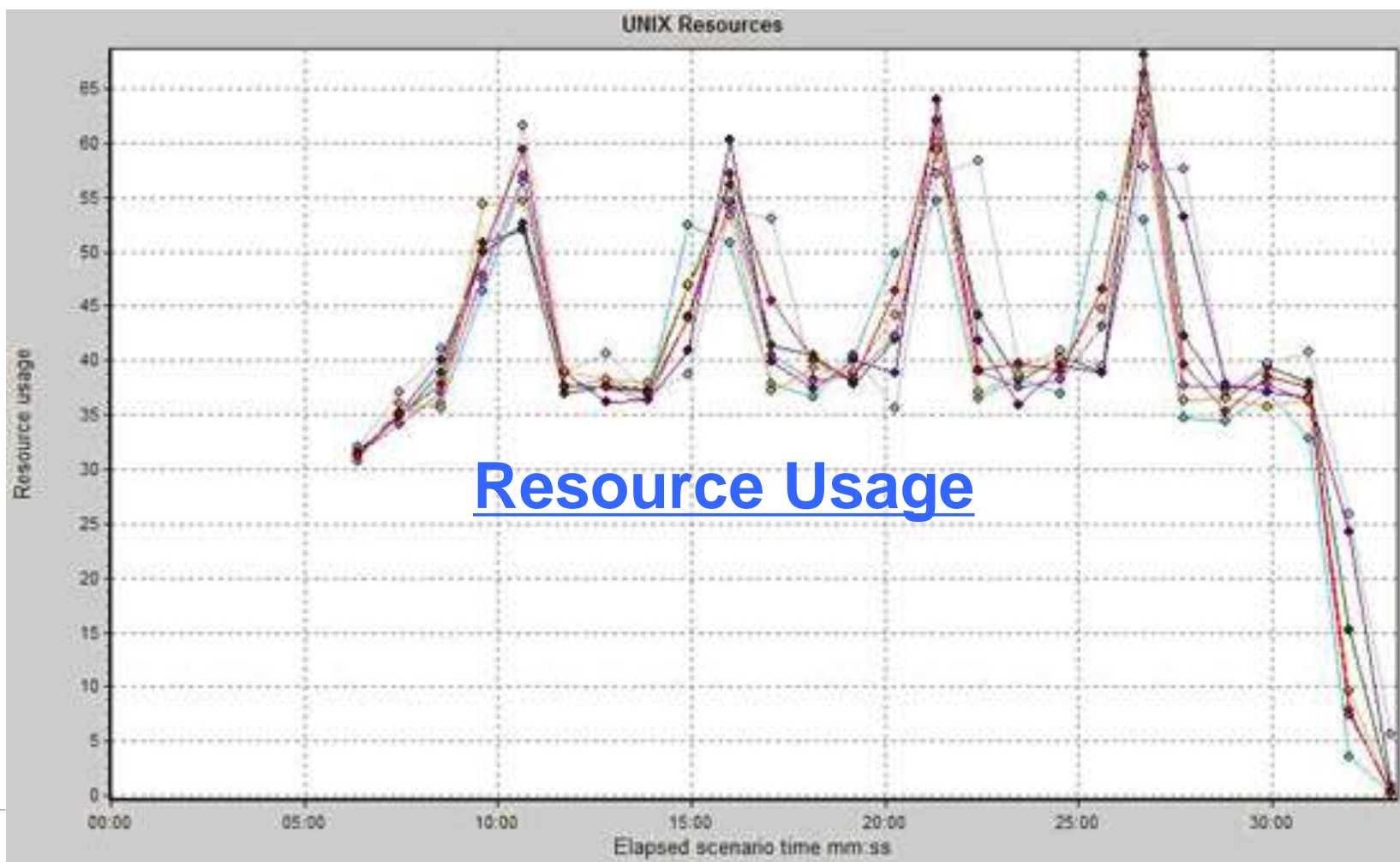
如，当在网站测试中所有的用户都同时执行几乎相同的操作时，可能会发生这种现象，这将会产生非常不可靠和不精确的结果

波动的出现，吞吐量不再是平滑的，获得的值也不精确。



# Flat测试 “波动” 效应

- ▶ 这在系统的各个方面都有所体现。



# Flat测试 “波动” 效应

---

## ► 解决方法

必须采取一些措施防止这种情况的出现。

有两种方法可以从这种类型的结果中获得精确的测量值：

如果测试可以运行相当长的时间（有时是几个小时，取决于用户的操作持续的时间），最后由于随机事件的本性使然，服务器的吞吐量会被“拉平”。

或者，可以只选取波形中两个平息点之间的测量值。该方法的缺点是可以捕获数据的时间非常短。



# 基准测试（Benchmark Test, BMT）

---

## ▶ 基准测试

基准测试是对测试对象的某些性能指标进行定量和可对比的测试。

基准测试强调要获得一致的、可再现的结果。

可再现的结果有两个好处：减少重新运行测试的次数；对测试的产品和产生的数字更为确信。

如，假定测试的两个指标是服务器的响应时间和吞吐量，两指标会受到负载的影响，而负载又受两个因素影响：（如下二页）

同时与服务器通信的连接（或虚拟用户）的数目，

每个虚拟用户请求之间间隔时间的长短。

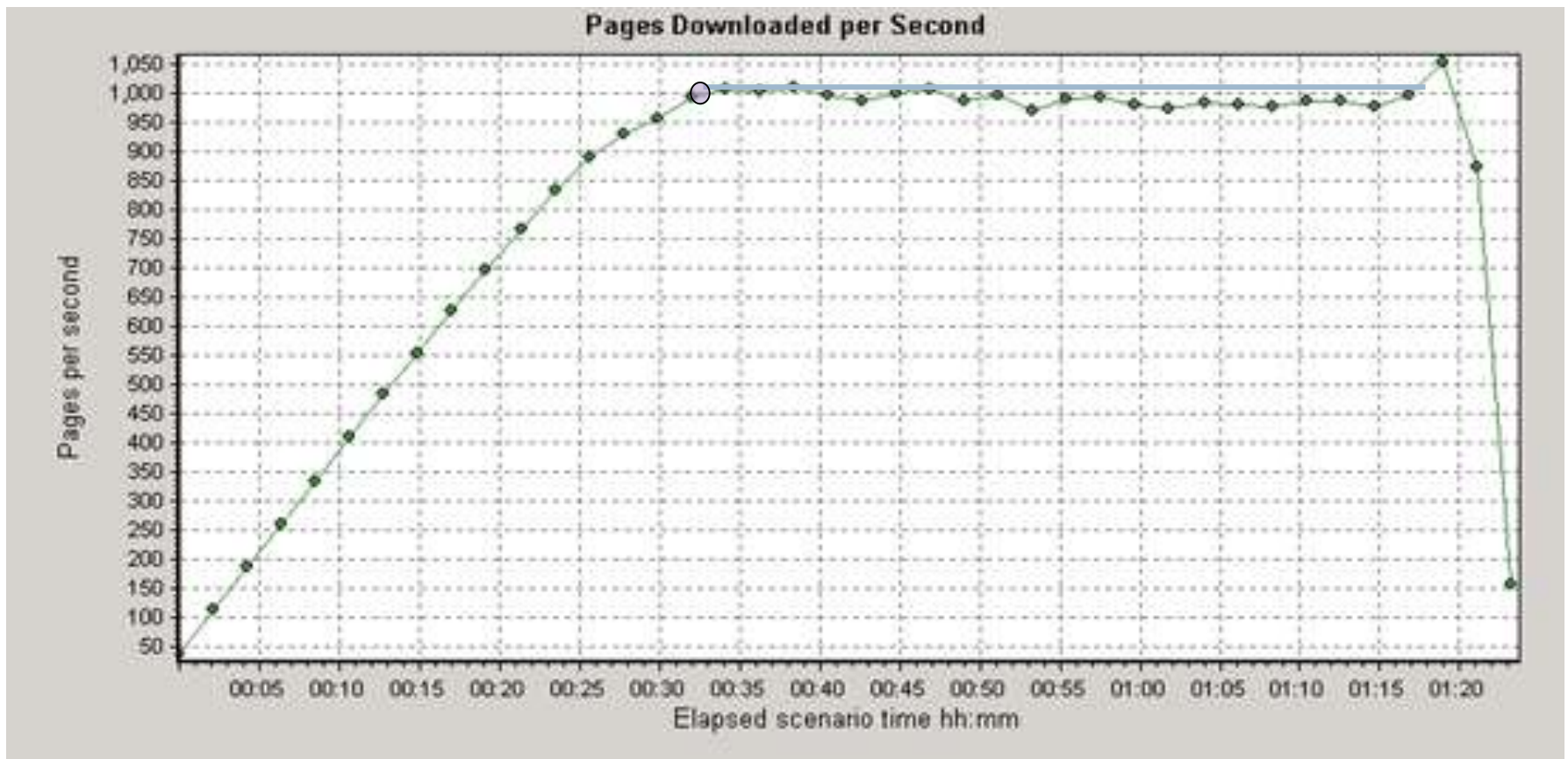
与服务器通信的用户越多，负载就越大。同样，请求之间的间隔时间越短，负载也越大。这两因素的不同组合会产生不同的服务器负载等级。

这时用基准测试，就可对响应时间和吞吐量获得可再现的结果。

---

# 基准测试（Benchmark Test, BMT）

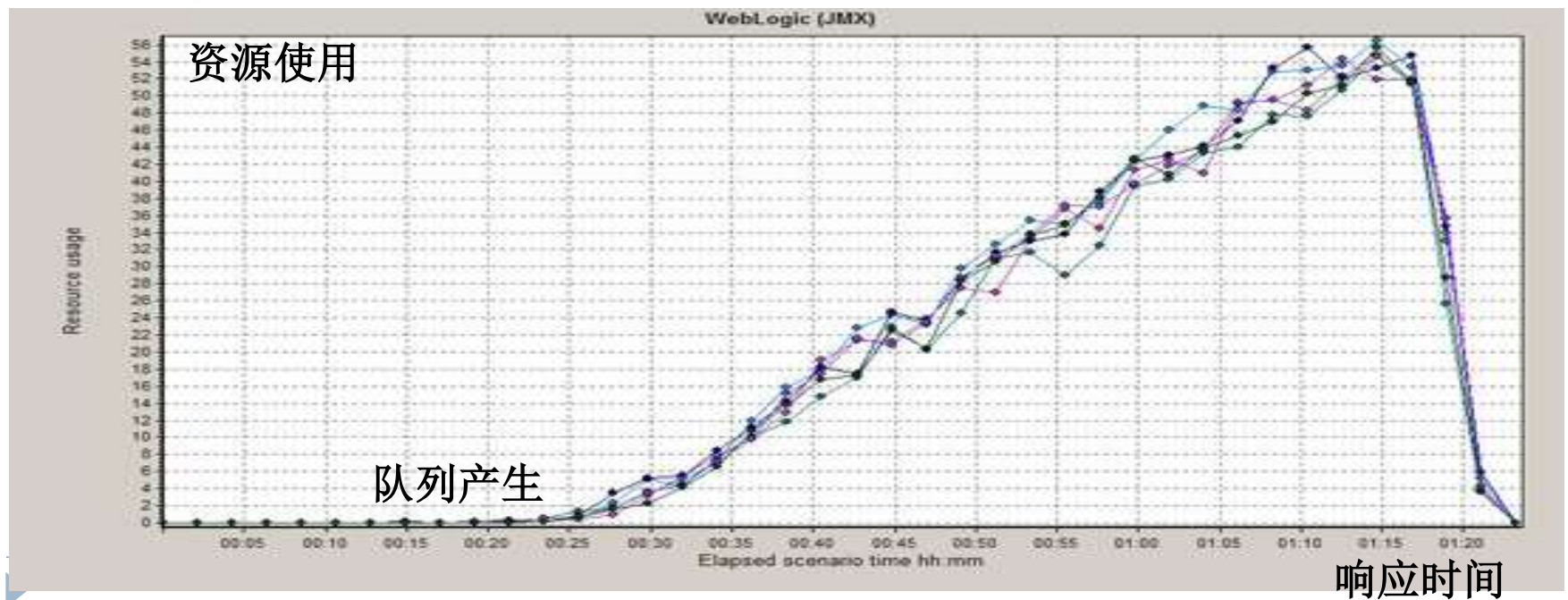
- 随着服务器上负载的增加，吞吐量会不断攀升，直到到达一个点，并在这个点上稳定下来





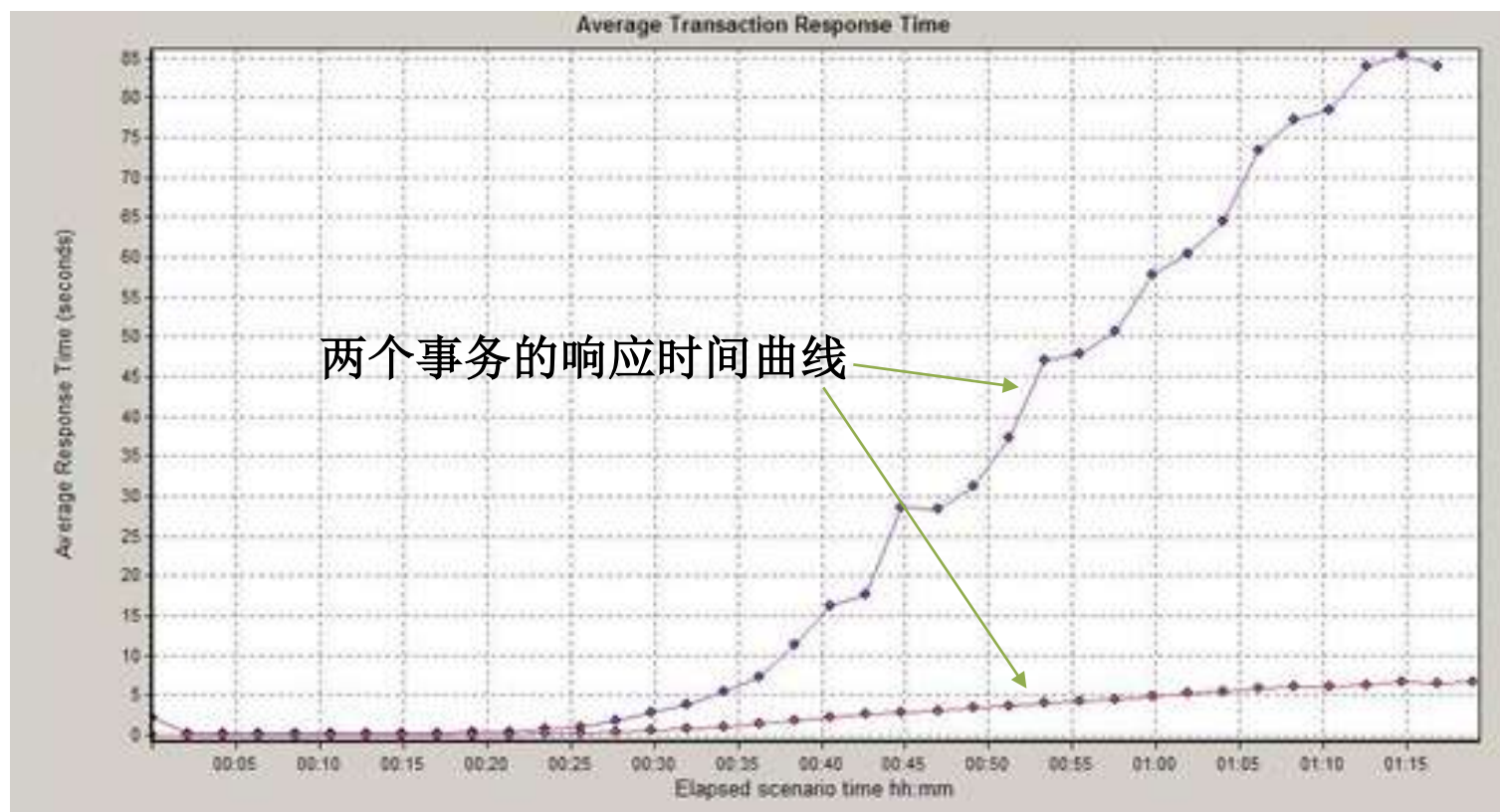
# 基准测试 (Benchmark Test, BMT)

- ▶ 在上一稳定点上，执行队列开始增长，因为服务器上所有的线程都已投入使用，传入的请求不再被立即处理，而是放入队列中，当线程空闲时再处理。
- ▶ 当系统达到稳定点，服务器吞吐量保持稳定后，就达到了给定条件下的系统上限。但是，随着服务器负载的继续增长，响应时间也随之延长，虽然吞吐量保持稳定。



# 基准测试（Benchmark Test, BMT）

- 为获得可再现结果，将系统置于相同的高负载下，将请求之间间隔时间设为零。这样服务器会立即超载，并开始构建执行队列。如果请求（虚拟用户）数保持一致，基准测试的结果会非常精确 → flat运行是获得基准测试数据的理想模式





# 性能规划测试

---

## ▶ 目标

其目标是找出，在特定的环境下，给定应用程序的性能可以达到何种程度。通常，具体的目标是找出系统在特定的服务器响应时间下支持的当前用户的最大数。

如，设有4个服务器，在5秒或更少的响应时间下支持当前用户的最大数是多少？

或，如果要以5秒或更少的响应时间支持5000个当前用户，需要多少个服务器？

## ▶ 要确定系统的容量，需要考虑几个因素：

- ▶ 用户中有多少是并发与服务器通信的。
- ▶ 每个用户的请求时间间隔是多少。



# 性能规划测试

---

## ▶ 如何加载用户以模拟负载状态？

最好的方法是模拟高峰时间用户与服务器通信的状况。

- ▶ 如果用户负载状态是在一段时间内逐步达到的,选择ramp-up测试,每隔几秒增加X个用户;
- ▶ 如果所有用户是在一个非常短的时间内同时与系统通信,就应该使用flat测试,将所有的用户同时加载到服务器

## ▶ 什么是确定容量的最好方法？

结合两种负载类型的优点,并运行一系列的测试

如:首先使用ramp-up测试确定系统支持的用户范围→该范围内不同的并发用户负载进行一系列的flat测试,更精确地确定系统的容量。



# 渗入测试

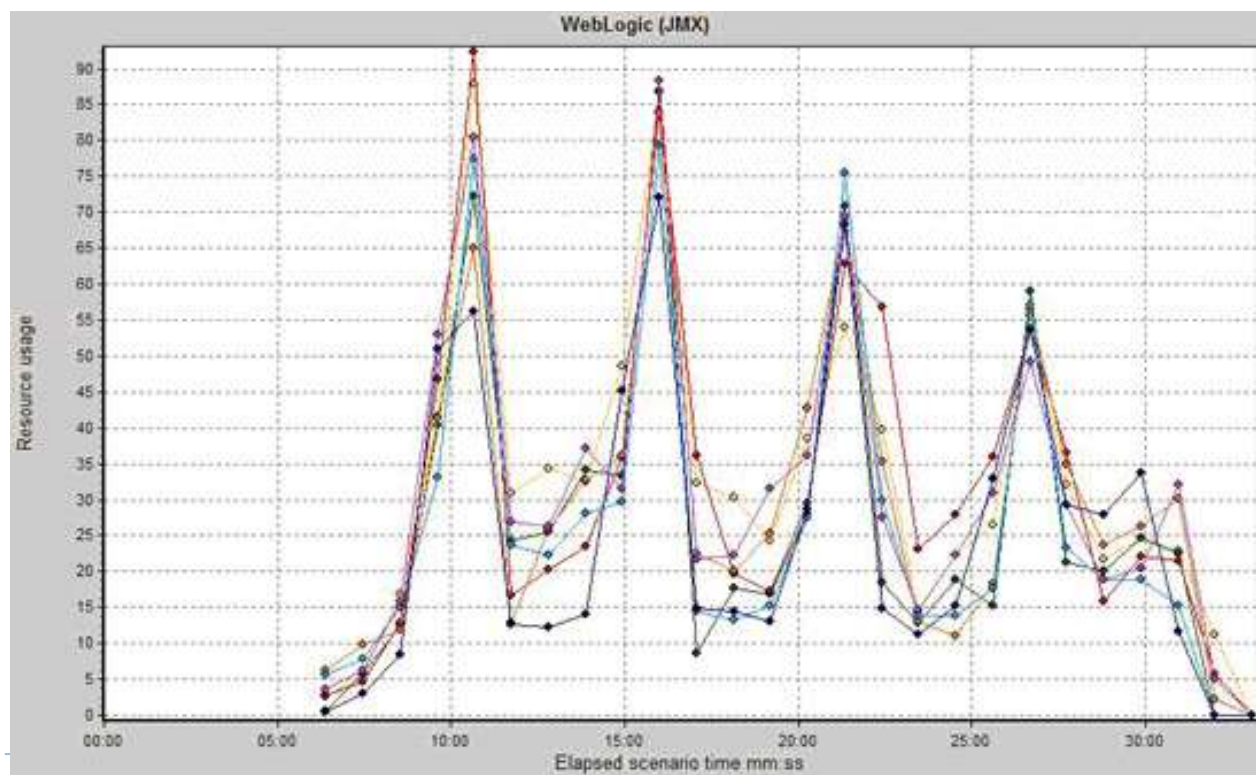
---

- ▶ 渗入测试是一种比较简单的性能测试。渗入测试所需时间较长，它使用固定数目的并发用户测试系统的总体健壮性。这些测试将会通过内存泄漏、增加的垃圾收集(GC)或系统的其他问题，显示因长时间运行而出现的任何性能降低。
- ▶ 建议运行两次测试——一次使用较低的用户负载（要在系统容量之下，以便不会出现执行队列），一次使用较高的负载（以便出现积极的执行队列）。



# 峰谷测试

- 兼有容量规划ramp-up测试和渗入测试的特征,目标是确定从高负载(例如系统高峰时间的负载)恢复、转为几乎空闲、然后再攀升到高负载、再降低的能力。查看系统是否显示了内存或GC性能降低的有关迹象?



# 系统测试

---

## ▶ 负载测试

负载测试是模拟实际软件系统所承受的负载条件的系统负荷，通过不断加载（如逐渐增加模拟用户的数量）或其它加载方式来观察不同负载下系统的响应时间和数据吞吐量、系统占用的资源（如CPU、内存）等，以检验系统的行为和特性，以发现系统可能存在的性能瓶颈、内存泄漏、不能实时同步等问题。

## ▶ 负载测试的目标

确定在各种工作负载下系统的性能。

确定并确保系统在超出最大预期工作量的情况下仍能正常运行。

此外，负载测试还要评估性能特征。例如，响应时间、事务处理速率和其他与时间相关的方面。



# 系统测试

---

## ▶ 强度测试（压力测试）

- ▶ 检验系统能力的最高实际限度，是检查在系统运行环境不正常乃至发生故障的情况下，系统可以运行到何种程度的测试。
- ▶ 进行强度测试时，让系统的运行处于资源的异常数量、异常频率和异常批量的条件下。

从本质上来说，测试者是想要破坏程序。关注这几方面：

测试压力估算、测试环境准备、问题的分析、累积效应

测试压力估算时遵循的一些准则为：

- ▶ 把输入数据速率提高一个数量级，确定输入功能将如何响应。
- ▶ 设计需要占用最大存储量或其它资源的测试用例进行测试。
- ▶ 设计出在虚拟存储管理机制中引起“颠簸”的测试用例进行测试。
- ▶ 设计出会对磁盘常驻内存的数据过度访问的测试用例进行测试。

# 系统测试 - 强度测试

---

## ▶ 测试环境

- ▶ 测试环境包括硬件环境（服务器、客户端等）、网络环境（通信协议、带宽等）、测试程序、数据准备等。
- ▶ 分析强度测试中易出现瓶颈处，从而有目的地调整测试环境或测试策略，使强度测试反映出软件的性能。
  - ▶ 压力稳定测试：在选定压力下，持续24小时以上进行稳定性测试。
  - ▶ 破坏性加压测试：不断加压，造成系统崩溃或让问题暴露。

# 系统测试 - 强度测试

---

## ▶ 问题分析

强度测试常采用黑盒测试方法，测试人员难定位问题根源，所以适当的分析和详细记录十分重要：

- ▶ 查看服务器上的进程及相应的日志文件可能立刻找到问题的关键；
- ▶ 查看监视系统性能的日志文件，找出问题出现的关键时间，系统状态；
- ▶ 检查测试运行参数，适当调整，重新测试，看问题能否再现；
- ▶ 对问题进行分解、屏蔽某些因数或功能，试着重现问题。

## ▶ 累积效应

测试中最好不要重做系统，因为这会忽略累积效应，使一些缺陷无法被发现。



# 系统测试 - 强度测试

---

## ▶ 压力测试类型

- ▶ 并发性能测试
- ▶ 疲劳强度测试
- ▶ 大数据量测试



# 系统测试 - 强度测试

---

## ▶ 并发性能测试

考察客户端应用的性能，测试的入口是客户端。

并发性能测试的过程，是一个负载测试和压力测试的过程，即逐渐增加并发虚拟用户数负载，直到系统的瓶颈或者不能接收的性能点，通过综合分析交易执行指标、资源监控指标等来确定系统并发性能的过程。

并发性能测试是负载压力测试中的重要内容。

ramp-up测试

# 系统测试 - 强度测试

---

## ▶ 疲劳强度测试

通常是采用系统稳定运行情况下能够支持的最大并发用户数或者日常运行用户数，持续执行一段时间业务，通过综合分析交易执行指标和资源监控指标来确定系统处理最大工作量强度性能的过程。

疲劳强度测试案例制定的原则是保证系统长期不间断运行的业务量，并应该尽量去满足该条件。

flat测试

# 系统测试 - 强度测试

---

## ▶ 大数据量测试

### ▶ 独立的数据量测试

针对某些系统存储、传输、统计、查询等业务进行大数据量测试。

### ▶ 综合数据量测试

和压力性能测试、负载性能测试、并发性能测试、疲劳性能测试相结合的综合测试方案

## ▶ 测试后的一些瓶颈分析举例



# 系统瓶颈分析

---

- ▶ 交易的响应时间如果很长，远远超过系统性能需求，表示耗费CPU的数据库操作，例如排序，执行聚合函数（如sum、min、max、count）等较多，可考虑是否有索引以及索引建立的是否合理；尽量使用简单的表联接；水平分割大表格等方法来降低该值。
- ▶ 分段排除错误。测试工具可以模拟不同的虚拟用户来单独访问Web服务器、应用服务器和数据库服务器，这样，就可以在Web端测出的响应时间减去以上各个分段测出的时间就可以知道瓶颈在哪并着手调优。
  -
- ▶ SQLServer资源监控中指标缓存点击率（Cache Hit Ratio），该值越高越好。如果持续低于80%，应考虑增加内存。注意该参数值是从SQL Server启动后，就一直累加记数，所以运行经过一段时间后，该值将不能反映系统当前值。



# 系统瓶颈分析

---

- ▶ OS资源监控指标中的内存页交换速率（Paging rate），如果该值偶尔走高，表明当时有线程竞争内存。如果持续很高，则内存可能是瓶颈。也可能是内存访问命中率低。“Swap in rate”和“Swap out rate”也有类似的解释。
- ▶ OS资源监控指标中的CPU占用率（CPU utilization），如果该值持续超过95%，表明瓶颈是CPU。可以考虑增加一个处理器或换一个更快的处理器。合理使用的范围在70%以内。
- ▶ OS资源监控指标中的指标磁盘交换率（Disk rate），如果该参数值一直很高，表明I/O有问题。可考虑更换更快的硬盘系统、重新部署业务逻辑等，另外设置Tempdb in RAM（内存临时表空间），减低“max async IO”（最大异步IO）等措施都会降低该值。



# 例1：“新华社多媒体数据库 V1.0”性能测试

---

## ► 概述

中国软件评测中心（CSTC）根据新华社技术局提出的《多媒体数据库（一期）性能测试需求》和GB/T 17544《软件包质量要求和测试》的国家标准，对“新华社多媒体数据库 V1.0”进行性能测试。

## ► 性能测试的目的

是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。



# 例1：“新华社多媒体数据库 V1.0”性能测试

---

## ▶ 性能测试的重点

是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。

## ▶ 方案

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、多检索词以及变检索式、混合检索业务等并发测试案例。

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小时的单检索词检索。在进行并发和疲劳测试的同时，监测的测试指标包括交易处理性能以及UNIX (Linux)、Oracle、Apache资源等。

---



# 例1：“新华社多媒体数据库 V1.0”性能测试

---

## ► 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，系统能承受并发用户数为200的负载压力，最大交易数/分钟达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。

通过对系统UNIX（Linux）、Oracle和Apache资源的监控，系统资源能满足上述并发和疲劳性能需求，且系统硬件资源尚有较大余地。

当并发用户数超过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。

建议进一步优化软件系统，充分利用硬件资源，缩短交易响应时间。

---

## 例2

---

针对某公司办公自动化（OA）系统的负载压力测试，采用专业的负载压力测试工具来执行测试。系统采用Browse/Server 架构，服务器是一台 PC Server（4 路2.7GHz 处理器，4GB 内存），安装的平台软件包括 Microsoft Internet Information Server 5.0，ASP.NET，SQLServer 2000。使用2 台笔记本电脑安装测试工具模拟客户端执行“登录”业务操作。

测试目标分别为以下两个：

- 第一，测试系统分别在2M、4M 网络带宽下，能够支持用户登录的最大并发用户数；
- 第二，测试服务器的吞吐量（即：每秒可以处理的交易数），主要包括服务器CPU平均使用率达到85%时系统能够支持的最大吞吐量和服务器CPU 平均使用率达到100%时系统能够支持的最大吞吐量。



## 例2

---

本次测试的性能需求是：指标“响应时间”合理范围为 0~5 秒。

测试结果如下：

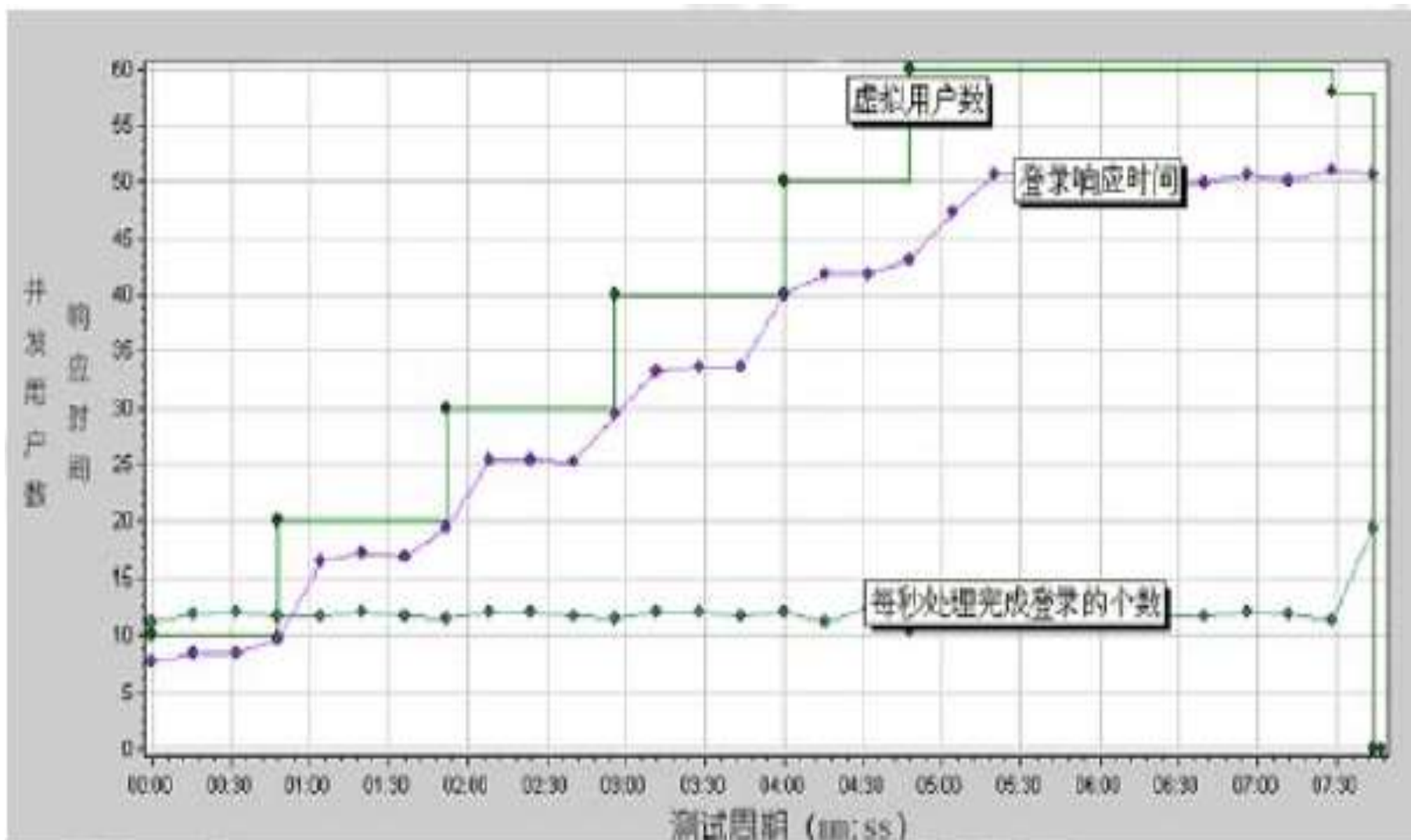
案例 1

网络环境：2M 带宽

客户端性能测试结果：

测试指标	平均值
登录响应时间	3.391 秒
虚拟用户数	N/A
每秒处理完成登录的个数	11.897 交易/秒

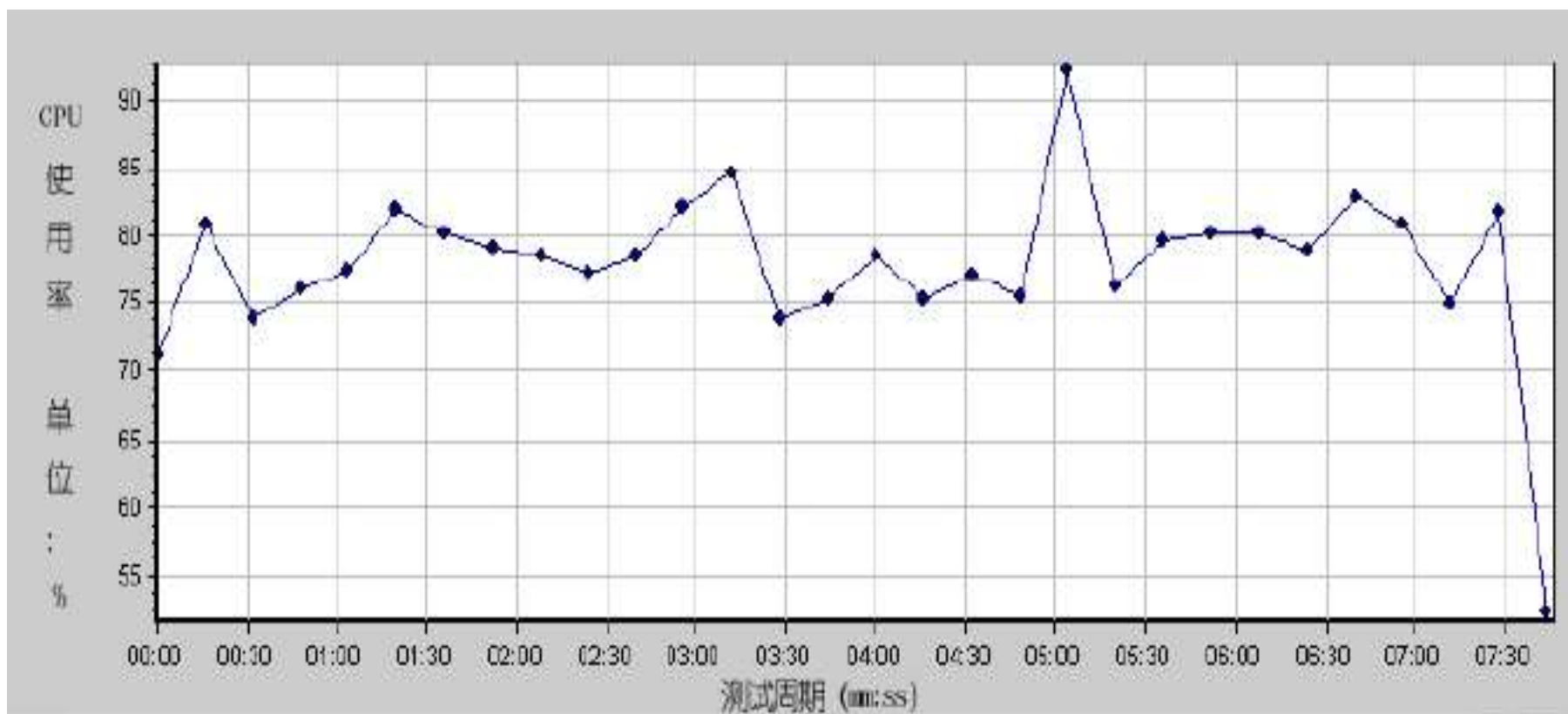




注：图中登录响应时间的纵坐标单位是0.1秒

## ► 服务器资源使用结果

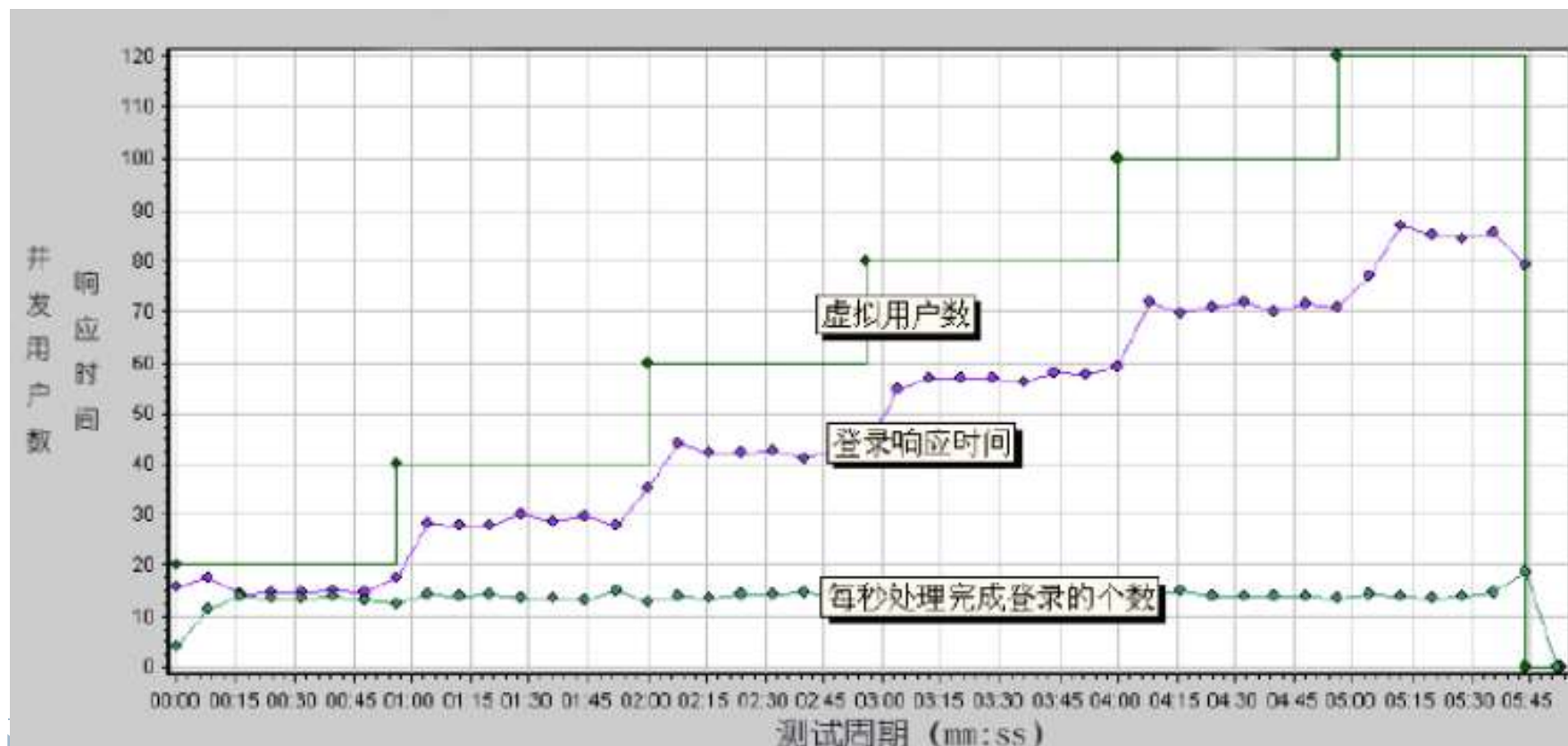
测试指标	平均值
CPU 使用率	78%



## 案例 2：网络环境：4M 带宽

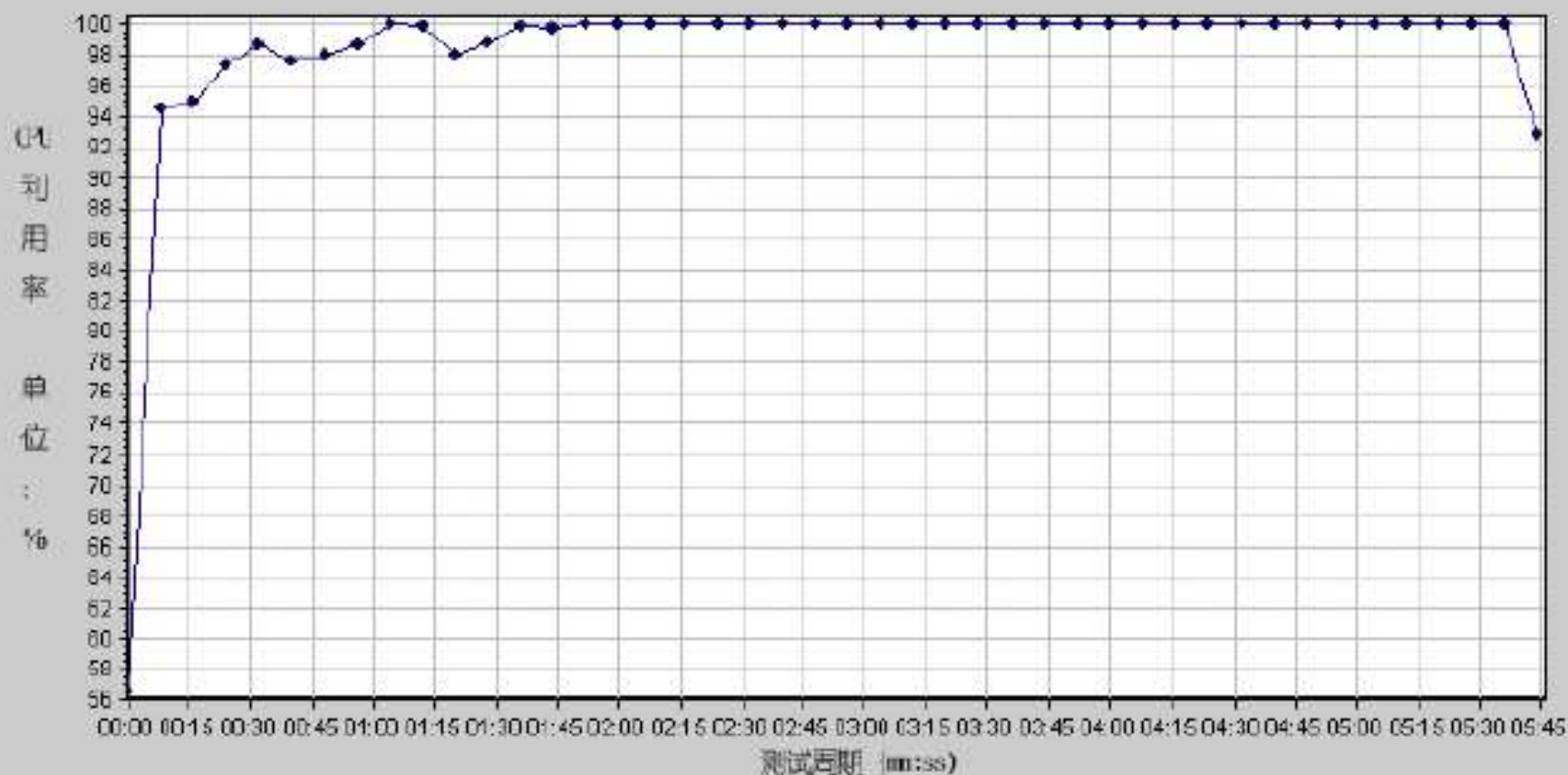
### 客户端性能测试结果

测试指标	平均值
登录响应时间	4.795 秒
虚拟用户数	N/A
每秒处理完成登录的个数	13.447 交易/秒



## 服务器资源使用结果

测试指标	平均值
CPU 使用率	98%



► 问1: “响应时间” (“RT”, Response Time) 的概念。

响应时间是系统完成事务执行准备后所采集的时间戳和系统完成待执行事务后所采集的时间戳之间的时间间隔，是衡量特定类型应用事务性能的重要指标，标志了用户执行一项操作大致需要多长时间。

► 问2: 分析案例1的测试结果数据，指出满足系统的性能指标需求时，系统能够承受的并发用户登录的最大数量，并说明理由。

系统能够承受的并发用户登录的最大数量为50。

题中指出“响应时间合理范围为0~5秒。”。案例1中，登录响应时间随虚拟并发用户数增加而增长。在50个虚拟并发用户的负载下，登录响应时间达到5秒。当负载超过50个虚拟并发用户，响应时间超过5秒。所以案例1中最合理的并发用户数为50。



▶ 问3：分析案例1的测试结果数据，说明服务器CPU资源使用率是否合理，以及带宽是否是系统瓶颈，并陈述理由。

服务器CPU资源使用率是合理的。2M带宽是系统处理业务的瓶颈。

理由是对比“4M带宽登录”案例，4M带宽下，系统每秒处理完成的登录个数固定在13.5个左右，登录响应时间随虚拟用户数增加而增长。在60个虚拟用户的压力下，登录响应时间在4.2秒左右。在80个虚拟用户的压力下，登录响应时间>5秒，所以在合理登录响应时间（5秒）内预计同时登录用户数是70左右。服务器CPU使用率成为系统处理的瓶颈。说明随着带宽的提高，系统的处理能力进一步提高，同时高吞吐量造成了系统资源的紧张，带来了新的系统性能瓶颈。

- 问4：分析案例 2 的测试结果数据，说明服务器CPU 资源使用率是否合理，以及增加带宽是否是提高系统性能的有效方法，并陈述理由。

服务器CPU资源使用率不合理，其平均值超过85%。

4M带宽的网络测试环境与2M带宽的网络测试环境相比，带来了新的系统瓶颈（CPU资源使用率平均值超过85%），所以增加带宽不是提高系统性能的有效方法。

在此基础上，继续提高带宽，系统的处理能力将进一步提高，高的处理能力会使服务器的资源瓶颈进一步加重，带来更加严重的后果。

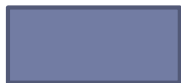
► 问5：论述 CPU 使用率成为系统性能瓶颈时，如何制定解决方案？论述网络带宽成为系统性能瓶颈时，如何制定解决方案？

当CPU资源使用成为系统瓶颈时的解决方案可以概括为：

1. 增加CPU的个数；
2. 提高CPU的主频；
3. 将web服务器与数据库服务器分开部署；
4. 调整软件的设计与开发；

当带宽成为系统瓶颈时的解决方案可以概括为：

1. 增加带宽；
2. 压缩传输数据。



# 安全性测试，可靠性

---

## ▶ 安全性测试、可靠性测试

目的不同，其手段和方法也不同，但都属于系统测试的范畴，有一定的联系，如软件可靠性要求通常包括了安全性的要求。

安全性测试、可靠性测试的技术比较深、实施比较难，但在应用系统中越来越重要。



# 系统测试 - 安全性测试

---

## ▶ 目录

- ▶ 术语
- ▶ 二种级别的安全性
- ▶ 威胁模式分析
- ▶ 软件安全概述
- ▶ 安全性防护策略、测试方法
- ▶ owasp top 10
- ▶ 计算机取证



机械工业出版社 ISBN:9787111185260

# 软件安全性测试

---

- 术语

- 软件安全性测试

不安全的软件，是有着巨大缺陷的软件。

ISO/IEC 9126 中安全性定义：

是与防止对程序及数据的非授权的故意或意外访问的能力有关的软件属性；

GB/T 15532-2008 中，关于安全性测试

从安全保密性方面，可测试系统及其数据访问的可控制性。

测试系统防止非法操作的模式，包括防止非授权的创建、删除或修改程序或信息，必要时做强化异常操作的测试。

测试系统防止数据被讹误和被破坏的能力。

测试系统的加密和解密功能

# 软件安全性测试

---

- 术语

- 软件安全性测试

安全测试：检查系统对不安全因素的防范能力。

不安全的因素有：黑客、病毒、蠕虫、间谍软件、后门程序、木马、拒绝服务攻击等；

驾驶攻击：随着在城域网中普及无线高保真（WiFi）网络，黑客们可驾驶车子，带着笔记本，在城市的街道上兜圈子，一旦搜索到未受保护的无线网络，即进行攻击，这种技术就是“驾驶攻击”。

# 软件安全性测试

---

- ▶ **安全产品：**是指产品在系统的所有者或管理员的控制下，保护用户信息的保密性、完整性、可获得性，以及处理资源的完整性和可获得性。  
([www.microsoft.com/technet/community/chats/trans/security/sec0612.mspx](http://www.microsoft.com/technet/community/chats/trans/security/sec0612.mspx))
- ▶ **安全漏洞：**是指使产品不可行的缺陷——即使是正确地使用产品时仍不能防止攻击者窃取系统的用户权限、调节操作、破坏数据，或建立未授权的信任等。  
([www.microsoft.com/technet/archive/community/columns/security/essays/vulnrbl.mspx](http://www.microsoft.com/technet/archive/community/columns/security/essays/vulnrbl.mspx))
- ▶ **黑客：**精通计算机编程和使用的人，电脑玩家。使用编程技能来获得对计算机网络或文件的非法访问的人。



# 软件安全性测试

---

## ▶ 为什么有人要攻击你的软件。

了解动机能帮助软件测试员考虑到测试的软件中有哪些安全方面的漏洞。

- 1) 挑战/成名
- 2) 好奇
- 3) 使用/借用
- 4) 恶意破坏
- 5) 偷窃

# 软件安全性测试

---

## ▶ 二种级别的安全性

- ▶ 应用程序级的安全性
- ▶ 系统级别的安全性

## ▶ 测试目标

- ▶ 应用程序级的安全性：核实操作者只能访问其所属的用户类型已被授权访问的那些功能和数据。
- ▶ 系统级别的安全性：核实只有具备权限的用户才能访问系统和应用程序。

# 软件安全性测试

---

## ► 威胁模式分析 (threat modeling)

(来源 《Writing Secure Code》 (Microsoft Press, 2003, second edition) )

由评审小组查找产品特性设置方面可能会引起安全漏洞的地方。

根据这些信息，开发小组对产品做修改，花更多的努力设计特定的功能，或集中精力测试潜在的故障点。最终使产品更加安全。

执行威胁模式分析并非软件测试员的责任。这个责任应是项目经理，并非项目小组每个成员都要参与。

威胁模型分析的步骤如下：

# 软件安全性测试

---

## ► 威胁模式分析步骤（7步）

### 1) 构建威胁模型分析小组

对于小组来说，重要的一点是了解他们的最初目标**不是解决安全问题，而是确定安全问题**。在后期可以隔离安全威胁，设计解决方案。

### 2) 确认价值

考虑系统所有的东西对于一个入侵者来说价值有多大。

### 3) 创建一个体系结构总体图

确认计划用在软件中及如何实现的技术。

创建一个体系结构图表示出主要的技术模块和它们之间如何通信，确认不同技术和其之间的信任边界，以及为了访问数据必须发生的授权。

# 软件安全性测试

---

## ► 威胁模式分析步骤（续）

### 4) 分解应用程序

确认数据所在位置，及系统如何处理。

### 5) 确认威胁

一旦完全理解了所有的部分（价值、体系结构、数据），威胁模型分析小组可以转向确认威胁。每一个部分都应该考虑成为威胁目标，并且应假设它们会受到攻击。

### 6) 记录威胁

每个威胁都必须用文档记录，并应进行跟踪以确保其被解决。

文档是一种简单方式，用于描述威胁、目标、攻击可能采用的方式、系统用于防御攻击有哪些反制手段。

# 软件安全性测试

---

## ► 威胁模式分析步骤（续）

### 7) 威胁等级评定

理解并非所有的威胁生来就平等。可用恐怖公式来确定每个威胁的等级。

## ► 恐怖公式（DREAD Formula）

1) 潜在的危害：如果被黑，损害多大？

2) 可反复性：被黑的几率？（黑客多少次尝试，能成功一次）

3) 可利用性：黑的技术难度？

4) 受影响的用户：有多少？

5) 可发现性：黑客发现漏洞的可能性？

在以上5个方面打分，1表示低，2表示中等，3表示高，然后加起来，获得5~15间的一个值，作为每个威胁的安全等级。

# 软件安全性测试-概述

---

## ▶ 软件安全是一项功能吗？软件漏洞是一个缺陷吗？

软件安全可以简单地看做是软件产品或系统的另外一项功能。

软件测试员不需要拿到一份清楚明白地定义软件安全性是如何实现的产品说明书。

软件测试员也不能假设威胁模型分析是完全和准确的。

基于上述二原因，测试者要用“失效性测试”，像黑客那样攻击被测软件，尽可能地找到各式安全漏洞。

# 安全系统防护体系

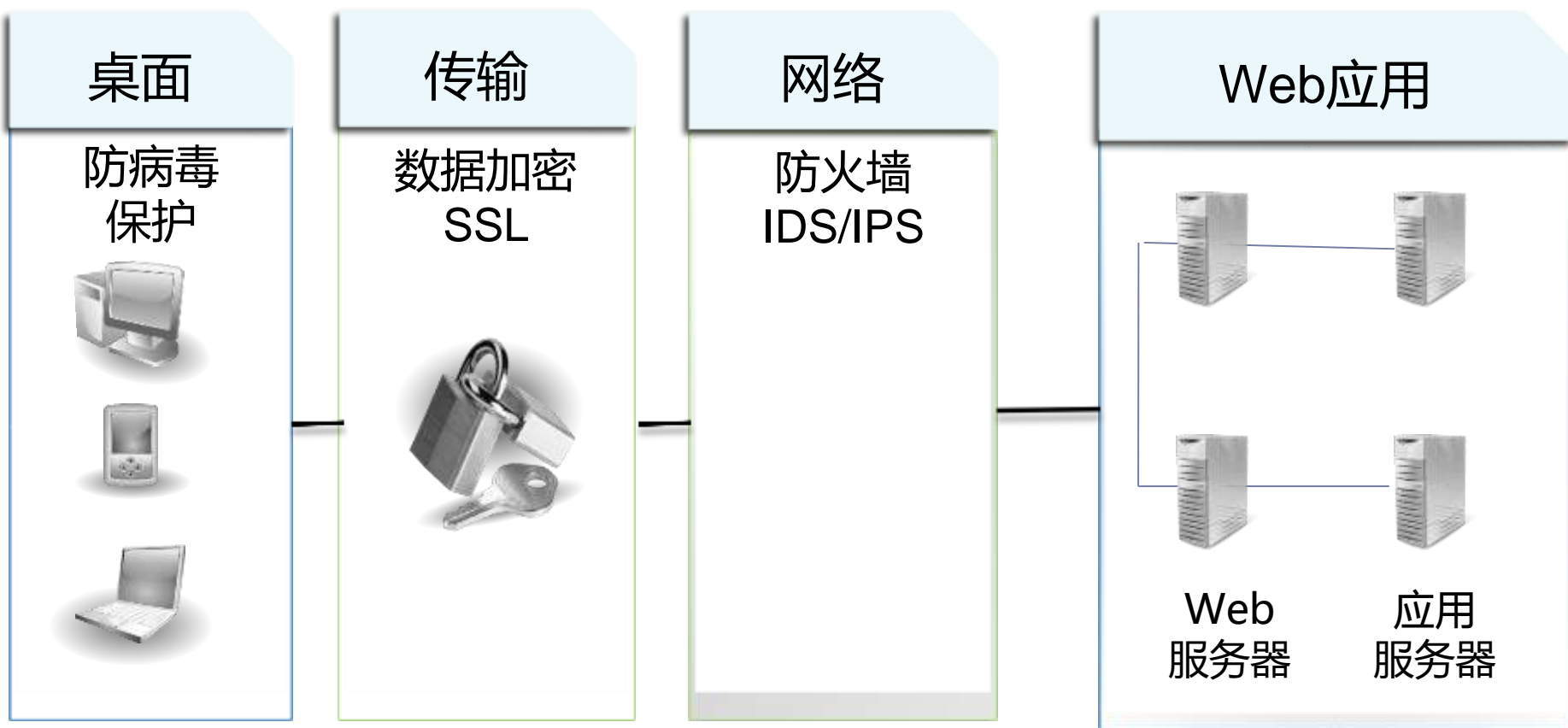
---

1	实体安全	基础设施的物理安全。
2	平台安全	网络平台、操作系统、基础通用应用平台(服务/数据库等)的安全。
3	数据安全	系统数据的机密性、完整性、访问控制和可恢复性。
4	通信安全	系统之间数据通信和会话访问不被非法侵犯。
5	应用安全	业务运行逻辑安全/业务资源的访问控制；业务交往的不可抵赖性/业务实体的身份鉴别/业务数据的真实完整性。
6	运行安全	对系统安全性的动态维护和保障，控制由于时间推移和系统运行导致安全性的变化。
7	管理安全	对相关人员、技术和操作进行管理，总揽以上各安全要素进行控制。





# 信息安全全景



IDS(入侵诊断系统)

IPS(入侵防御系统)

# 安全防护策略

---

安全防护策略	说明
安全日志	记录非法用户的登录名、操作时间、IP地址及内容等。
入侵检测	从系统内部和各种网络资源中主动采集信息，从中分析可能的网络入侵或攻击。
隔离防护	将系统中的安全部分与非安全部分进行隔离的措施，主要是防火墙和协议隔离。
漏洞扫描	对软件系统及网络系统进行与安全相关的检测，找出安全隐患和可被黑客利用的漏洞。
病毒防治	采用集中式管理，分布式杀毒等防毒技术。



# 安全性测试的方法

方法	说明
功能测试	对涉及到安全的软件功能：用户管理、权限管理、加密系统、认证系统等进行测试。采用黑盒测试方法。
漏洞扫描	使用特定的漏洞扫描软件完成。漏洞扫描软件分为主机漏洞扫描（Host Scanner）和网络漏洞扫描（Network Scanner）。
安全日志测试	测试日志的完整性、正确性。
模拟攻击试验	冒充、重演、消息篡改、服务拒绝、内部攻击、外部攻击、陷阱门、木马等。



# 软件安全性测试

---

## ► 攻击的几种方法

安全测试期间，测试人员假扮非法入侵者，采用各种办法试图突破防线。如：

- 以系统输入为突破口，利用输入的容错性进行正面攻击；
- 申请和占用过多的资源压垮系统，以破坏安全措施，从而进入系统；
- 故意使系统出错，利用系统恢复的过程，窃取用户口令及其它有用的信息；
- 浏览那些逻辑上不存在，但物理上还存在的各种记录和资料等。
- 通过浏览残留在计算机各种资源中的垃圾（无用信息），以获取如口令，安全码，译码关键字等信息；

（续下页）

# 软件安全性测试

---

## ► 攻击的几种方法(续)

- 浏览全局数据，期望从中找到进入系统的关键字；

理论上，只要有足够的时间和资源，没有无法进入的系统。因此安全设计的准则是使非法入侵的代价超过被保护信息的价值。此时非法侵入者已无利可图。

# OWASP TOP 10 Risks

---



## OWASP Top 10 2017

10项最严重的 Web 应用程序安全风险



<https://owasp.org>  
<http://www.owasp.org.cn>



本文档的发布基于《Creative Commons Attribution-ShareAlike 4.0 International License》



# OWASP TOP 10 Risks

---

## ▶ OWASP(开放Web软体安全项目 - Open Web Application Security Project):

是一个开源、非盈利的全球性安全组织，致力于应用软件的安全研究。使命是使应用软件更加安全，使企业和组织能够对应用安全风险作出更清晰的决策。

OWASP研究内容：目前有几个进行中的计划，包括最知名的OWASP Top 10(十大Web弱点)、安全PHP/Java/ASP.Net等，针对不同的软件安全问题进行讨论与研究。

目前OWASP全球拥有130个分会近万名会员，共同推动了安全标准、安全测试工具、安全指导手册等应用安全技术的发展。

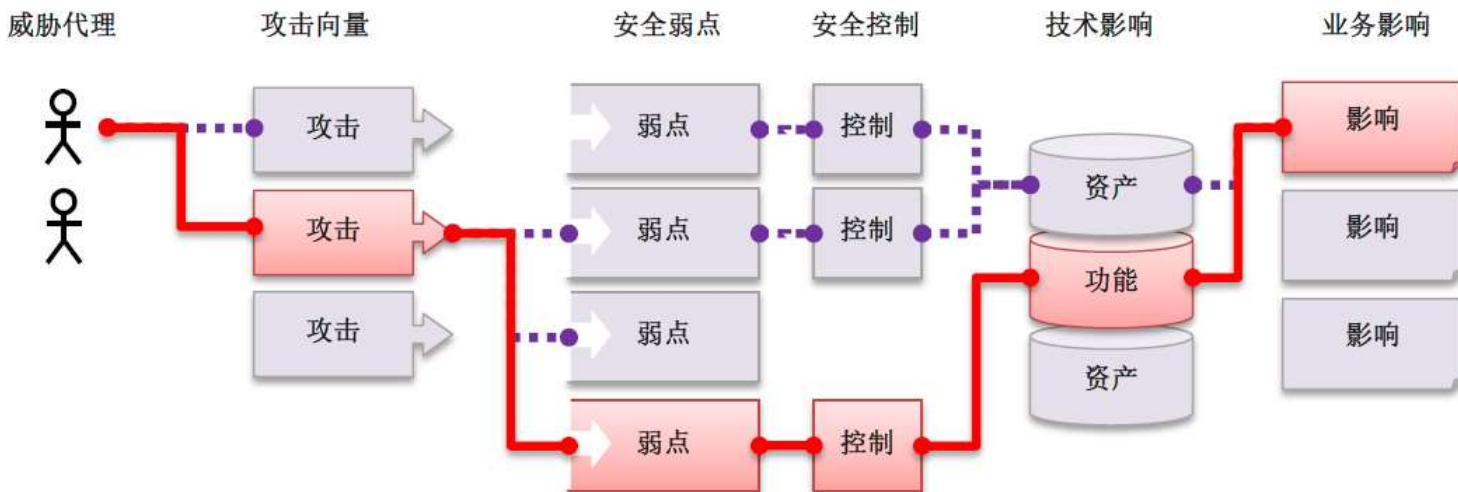
## ▶ OWASP在业界影响力：

美国联邦贸易委员会(FTC)强烈建议所有企业需遵循OWASP所发布的十大Web弱点防护守则；美国国防部亦列为最佳实务；国际信用卡数据安全标准PCI标准更将其列为必要组件。

---

# 应用程序安全风险

攻击者可以通过应用程序中许多不同的路径方法去危害您的业务或者企业组织



应用描述	可利用性：3	普遍性：2	可检测性：3	技术：3	业务？
几乎任何数据源都能成为注入载体，包括环境变量、所有类型的用户、参数、外部和内部Web服务。当攻击者可以向解释器发送恶意数据时， <a href="#">注入漏洞</a> 产生。		注入漏洞十分普遍，尤其是在遗留代码中。注入漏洞通常能在SQL、LDAP、XPath或是NoSQL查询语句、OS命令、XML解析器、SMTP包头、表达式语句及ORM查询语句中找到。  注入漏洞很容易通过代码审查发现。扫描器和模糊测试工具可以帮助攻击者找到这些漏洞。		注入能导致数据丢失、破坏或泄露给无授权方，缺乏可审计性或是拒绝服务。注入有时甚至能导致主机被完全接管。  您的应用和数据需要受到保护，以避免对业务造成影响。	



# OWASP TOP 10

## 应用安全风险- 2017

---

### ▶ A1-注入

将不受信任的数据作为命令或查询的一部分发送到解析器时，会产生诸如SQL注入、NoSQL注入、OS注入和LDAP注入的注入缺陷。攻击者的恶意数据可以诱使解析器在没有适当授权的情况下执行非预期命令或访问数据。

### ▶ A2失效的身份认证

通常，通过错误使用应用程序的身份认证和会话管理功能，攻击者能够破译密码、密钥或会话令牌，或者利用其它开发缺陷来暂时性或永久性冒充其他用户的身份。

### ▶ A3敏感数据泄露

许多Web应用程序和API都无法正确保护敏感数据，例如：财务数据、医疗数据和PII数据。攻击者可以通过窃取或修改未加密的数据来实施信用卡诈骗、身份盗窃或其他犯罪行为。未加密的敏感数据容易受到破坏，因此，我们需要对敏感数据加密，这些数据包括：传输过程中的数据、存储的数据以及浏览器的交互数据。

---

# OWASP TOP 10 -2017

---

## ▶ A4 - XML 外部实体 (XXE)

许多较早的或配置错误的XML处理器评估了XML文件中的外部实体引用。攻击者可以利用外部实体窃取使用URI文件处理器的内部文件和共享文件、监听内部扫描端口、执行远程代码和实施拒绝服务攻击。

## ▶ A5 - 失效的访问控制

未对通过身份验证的用户实施恰当的访问控制。攻击者可以利用这些缺陷访问未经授权的功能或数据，例如：访问其他用户的帐户、查看敏感文件、修改其他用户的数据、更改访问权限等。

## ▶ A6 安全配置错误

安全配置错误是最常见的安全问题，这通常是由于不安全的默认配置、不完整的临时配置、开源云存储、错误的HTTP标头配置以及包含敏感信息的详细错误信息所造成的。因此，我们不仅需要对所有的操作系统、框架、库和应用程序进行安全配置，而且必须及时修补和升级它们。

---



# OWASP TOP 10 -2017

---

## ▶ A7 - 跨站脚本 (XSS)

当应用程序的新网页中包含不受信任的、未经恰当验证或转义的数据时，或者使用可以创建 HTML 或 JavaScript 的浏览器 API 更新现有的网页时，就会出现 XSS 缺陷。XSS 让攻击者能够在受害者的浏览器中执行脚本，并劫持用户会话、破坏网站或将用户重定向到恶意站点。

## ▶ A8 不安全的反序列化

不安全的反序列化会导致远程代码执行。即使反序列化缺陷不会导致远程代码执行，攻击者也可以利用它们来执行攻击，包括：重播攻击、注入攻击和特权升级攻击。



# OWASP TOP 10 -2017

---

## ▶ A9 使用含有已知漏洞的组件

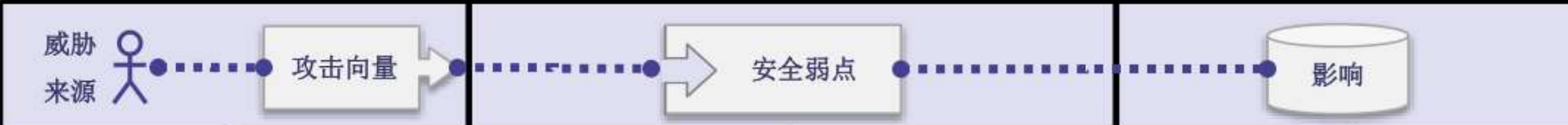
组件（例如：库、框架和其他软件模块）拥有和应用程序相同的权限。如果应用程序中含有已知漏洞的组件被攻击者利用，可能会造成严重的丢失数据或服务器接管。同时，使用含有已知漏洞的组件的应用程序和API可能会破坏应用程序防御、造成各种攻击并产生严重影响。

## ▶ A10 - 不足的日志记录和监控

不足的日志记录和监控，以及事件响应缺失或无效的集成，使攻击者能够进一步攻击系统、保持持续性或转向更多系统，以及篡改、提取或销毁数据。大多数缺陷研究显示，缺陷被检测出的时间超过200天，且通常通过外部检测方检测，而不是通过内部流程或监控检测。



# A1注入

					
应用描述	可利用性：3	普遍性：2	可检测性：3	技术：3	业务？
几乎任何数据源都能成为注入载体，包括环境变量、所有类型的用户、参数、外部和内部Web服务。当攻击者可以向解释器发送恶意数据时， <a href="#">注入漏洞</a> 产生。	注入漏洞十分普遍，尤其是在遗留代码中。注入漏洞通常能在SQL、LDAP、XPath或是NoSQL查询语句、OS命令、XML解析器、SMTP包头、表达式语句及ORM查询语句中找到。  注入漏洞很容易通过代码审查发现。扫描器和模糊测试工具可以帮助攻击者找到这些漏洞。			注入能导致数据丢失、破坏或泄露给无授权方，缺乏可审计性或是拒绝服务。注入有时甚至能导致主机被完全接管。  您的应用和数据需要受到保护，以避免对业务造成影响。	

# A1注入

---

## ► 攻击案例场景

场景#1: 应用程序在下面存在漏洞的SQL语句的构造中使用不可信数据:

```
String query = "SELECT * FROM accounts WHERE custID=" +  
               request.getParameter("id") + "";
```

场景#2: 同样的, 框架应用的盲目信任, 仍然可能导致查询语句的漏洞。(例如: Hibernate查询语言(HQL)):

```
Query HQLQuery= session.createQuery( "FROM accounts  
WHERE custID=' " + request.getParameter("id") + "'");
```

在这两案例中, 攻击者在浏览器中将“id”参数的值修改成‘or’ 1’ =’ 1。如:

```
http://example.com/app/accountView?id=' or '1'='1
```

这样查询语句的意义就变成了从accounts表中返回所有的记录。更危险的攻击可能导致数据被篡改甚至是存储过程被调用。

---

---

更多参看

《OWASP Top 10 2017》 v1.1

# 安全测试下一步做什么？

---

## ▶ 建立持续性的应用安全测试

安全编码很重要。但验证你想达到的安全性是否真实存在、是否正确、是否像我们想的那样也很关键。应用程序安全测试的目标是提供这些证据。这项工作困难而复杂，因此，强烈建议你思考如何专注于整个应用程序组合中重要的地方，并且低成本高收益。

当前安全风险变化很快，每年进行一次扫描或渗透测试的日子已经过去了。现代软件开发需要在整个软件开发生命周期中进行持续的应用安全测试。





# 安全测试下一步做什么？

---

## ► 理解威胁模型

在你测试之前，请了解业务中需要耗时的重要部分。优先级来源于威胁模型，所以如果你还没有威胁模型，那么需要在测试开始前建立一个。考虑使用《OWASP应用安全验证标准》和《OWASP测试指南》作为指导标准，而不依赖工具厂商的结果来判断哪些是重要业务。

## ► 理解你的软件开发流程

你的应用安全测试方法必须与你们软件开发流程（SDLC）中的人员、工具和流程高度匹配。试图强制推动额外的步骤、门限和审查可能会导致摩擦、绕行和一定范围内的争议。寻找自然而然的机会去收集安全信息，然后将融合进你的流程。



# 安全测试下一步做什么？

---

## ▶ 测试策略

选择最简单、快速、准确的方法去验证每项需求。

OWASP 基准测试项目可以有助于评估各安全工具对OWASP TOP 10 风险的检测能力，从而为你的特定需求挑选出最合适的工具。注意考虑用于工具误报的人力成本和漏报的严重危害。

## ▶ 实现全面性和准确性

你不需要一切都要立刻测试。先关注那些重要的方面，然后随着时间扩展你的全面性。这意味着逐步扩展安全防御库和自动验证库，以及扩展应用系统和API本身的覆盖。目标是所有的应用程序和API基本安全性都能获得持续性的验证。



# 安全测试下一步做什么？

---

## ▶ 体现报告的价值

不管你测试得多么专业，若不有效与别人沟通都等于白做。展示你对程序运行的理解，从而建立互信关系。不必用晦涩难懂专业用语，清楚描述漏洞的滥用风险，然后在某场景下真实展现攻击。要对漏洞发现与利用难度及引发的后果做真实的评估。最后提交结果时请使用开发团队正在使用的文档工具格式，而不是简单的PDF。



## 举例（2014年软件评测师考试下午题三）

阅读下列说明，回答问题1至问题4，将解答填入答题纸的对应栏内。

**【说明】**某大型披萨加工和销售商为了有效管理披萨的生产和销售情况，欲开发一套基于Web的信息系统。其主要功能为销售、生产控制、采购、运送、存储和财务管理等。系统采用Java EE平台开发，页面中采用表单实现数据的提交与交互，使用图形（Graphics）以提升展示效果。

**【问题1】**（6分）设计两个表单项输入测试用例，以测试XSS（跨站点脚本）攻击。系统设计时可以采用哪些技术手段防止此类攻击。

**【问题2】**（3分）简述图形测试的主要检查点。

**【问题3】**（5分）简述页面测试的主要方面。

**【问题4】**（6分）系统实现时，对销售订单的更新所用的SQL语句如下：

```
PreparedStatement pstmt = connection.prepareStatementC( "Update  
SalesOrder Set status=? Where OrderID=?;" )
```

然后通过setString(...)；的方式设置参数值后加以执行。

设计测试用例以测试SQL注入，并说明该实现是否能防止SQL注入。

---

**【问题1】**（6分）设计两个表单项输入测试用例，以测试XSS（跨站点脚本）攻击。系统设计时可以采用哪些技术手段防止此类攻击。

用例1 `<script>alert<'xss'></script>`

用例2 `<IMG SRC="javascript:alert('XSS');">`

如看到弹框，表示漏洞存在。

防御XSS攻击方法：验证所有输入数据，有效检测攻击；对所有输出数据进行适当的编码，以防止任何已成功注入的脚本在浏览器端运行。

防御规则：

不要在允许位置插入不可信数据； 在向HTML元素内容插入不可信数据前对HTML解码； 在向HTML常见属性插入不可信数据前进行属性解码； 在向HTML JavaScript DATA Values插入不可信数据前，进行JavaScript解码； 在像HTML样式属性插入不可信数据前，进行CSS解码； 在向HTML URL属性插入不可信数据前，进行URL解码。

# 举例

---

**【问题2】** (3分) 简述图形测试的主要检查点。

图形测试，主要检查点如下：

颜色饱和度和对比度是否合适

需要突出的链接颜色是否容易识别

是否正确加载所有的图形

**【问题3】** (5分) 简述页面测试的主要方面。

页面的一致性如何

在每个页面上是否设计友好的用户界面和直观的导航系统

是否考虑多种浏览器的需要

是否建立了页面文件的命名体系

是否充分考虑了合适的页面布局技术，如层叠样式表、表格和帧结构等

# 举例

【问题4】 (6分)系统实现时，对销售订单的更新所用的SQL语句如下：

```
PreparedStatement pstmt = connection.prepareStatementC( "Update  
SalesOrder Set status=? Where OrderID=?;" )
```

然后通过setString(...); 的方式设置参数值后加以执行。

设计测试用例以测试SQL注入，并说明该实现是否能防止SQL注入。

```
Pstmt.setString ('1' or '1=1', status)
```

```
Pstmt.setString ('1' or '1=1', orderID)
```

能防止SQL注入

PreparedStatement把参数中包含的单引号做了转义处理,相当于:

```
Update SalesOrder Set status='1 \' or \' 1=1' ...
```

整个字符串当成一个状态，而非：

```
Update SalesOrder Set status='1' or '1=1' ...
```

# 软件安全性测试

---

## 计算机取证

用户变更时未被删除的保留数据叫做潜在数据。

潜在数据是否是潜在的安全漏洞，需要在小组采用的任何威胁模型分析中进行讨论：也许这些数据不会被看成是产品的问题，也许会被看成是一个大问题。

潜在数据的更复杂的例子是由计算机安全专家用来发现可以用做犯罪调查的证据。





# 可靠性测试

---

## ► 软件的可靠性

是指：“在规定的一段时间和条件下，软件维持其性能水平的能力有关的一组属性，可用成熟性、容错性、易恢复性三个基本子特性来度量”。（ISO9126）

可靠性测试是从验证的角度出发，检验系统的可靠性是否达到预期的目标，同时给出当前系统可能的可靠性增长情况。

## ► 成熟性度量

成熟性：与由软件故障引起失效的频度有关的软件属性（ISO9126）

可通过错误发现率DDP（Defect Detection Percentage）来表现：

$$DDP = \text{测试发现的错误数} / \text{估算的错误总数}$$

在测试中查找出的错误越多，实际应用中出错的机会就越小，软件也就越成熟。

GB/T 15532-2008: 测试系统的平均无故障时间。

# 可靠性测试 - 容错性测试

- ▶ 容错性:与在软件故障或违反指定接口的情况下,维持规定的性能水平的能力有关的软件属性; (ISO 9126)
- ▶ 容错性测试是检查软件在异常条件下自身是否具有防护性的措施。
  - ▶ 当系统出错时,能否在指定时间间隔内修正错误并重启功能。
  - ▶ 当输入异常数据或进行异常操作,系统是否提供保护。

如果系统的容错性好的话,系统只给出提示或内部消化掉,而不会导致系统出错甚至崩溃。

- ▶ GB/T 15532-2008:从容错性方面考虑,可测试:
  - a) 系统对中断发生的反应;
  - b) 系统在边界条件下的反应;
  - c) 系统的功能、性能的降级情况;
  - d) 系统的各种误操作模式;
  - e) 系统的各种故障模式(如数据超范围、死锁);
  - f) 测试在多机系统出现故障需要切换时系统的功能和性能的连续平稳性。

# 可靠性测试 - 易恢复性测试

- ▶ 易恢复性:与在失效发生后,重建其性能水平并恢复直接受影响数据的能力以及为达此目的所需的时间和能力有关的软件属性; (ISO 9126)
- ▶ 易恢复测试
  - 是要证实在克服硬件故障(包括掉电、硬件或网络出错等)后,系统能否正常地继续进行工作,并不对系统造成任何损害。
- ▶ 方法
  - 可采用各种人工干预的手段,模拟硬件故障,故意造成软件出错,不能正常工作,进而检验系统的恢复能力。
  - 如: 往移动盘读写时不插入盘, 或写保护;
    - 不接打印机就打印;                      客户端或服务器断电;
    - 网络通信中断; 等

# 系统测试 - 可靠性测试 - 易恢复性测试

---

## ▶ 检查包括：

- ▶ 错误探测功能——系统能否发现硬件失效与故障；
- ▶ 能否切换或启动备用的硬件；
- ▶ 在故障发生时能否保护正在运行的作业和系统状态；
- ▶ 在系统恢复后能否从最后记录下来的无错误状态开始继续执行作业，等等。

如系统本身能够自动地进行恢复，则应检验：

重新初始化，检验点设置机构、数据恢复以及重新启动是否正确。

如这一恢复需要人为干预，应考虑平均修复时间是否在限定的范围以内。

# 系统测试 - 可靠性测试 - 易恢复性测试

---

► GB/T 15532-2008：从易恢复性方面考虑，可测试：

- a) 具有自动修复功能的系统的自动修复的时间；
- b) 系统在特定的时间范围内的平均宕机时间；
- c) 系统在特定的时间范围内的平均恢复时间；
- d) 系统的可重新启动并继续提供服务的能力；
- e) 系统的还原功能的还原能力。



# 兼容性测试

---

## ▶ 兼容性测试

软件兼容性测试是检测各软件之间能否正确地交互和共享信息，其目标是保证软件按照用户期望的方式进行交互，使用其它软件检查软件操作的过程。

## ▶ 软件兼容的实例：

- ▶ 从Web页面剪切文字，然后在文字处理程序中打开的文档中粘贴。
- ▶ 从电子表格程序保存账目数据，然后在另一个完全不同的电子表格程序中读入这些数据。
- ▶ 使图形处理软件在同一操作系统下的不同版本正常工作。
- ▶ 使文字处理程序从联系人管理程序中读取姓名和地址，打印个性化的邀请函和信封。
- ▶ 升级到新的数据库程序，读入现存所有数据库，并能够像老版本一样对其中的数据进行处理。

# 兼容性测试（续）

---

► 兼容性的测试通常需要解决以下问题：

- 1) 新开发的软件需要与哪种平台（操作系统、Web浏览器或操作环境）和应用软件保持兼容？如果要测的软件是一个平台，那么要求什么应用程序能在其上运行？
- 2) 应该遵守哪种定义软件之间交互的标准或者规范。
- 3) 软件使用何种数据与其它平台、与新的软件进行交互和共享信息。

也就是说，兼容性通常有4种

向前兼容与向后兼容

不同版本间的兼容

标准和规范兼容

数据共享兼容

# 兼容性测试（续）

---

## (1) 向前兼容和向后兼容

向前兼容是指可以使用软件的**未来版本**。

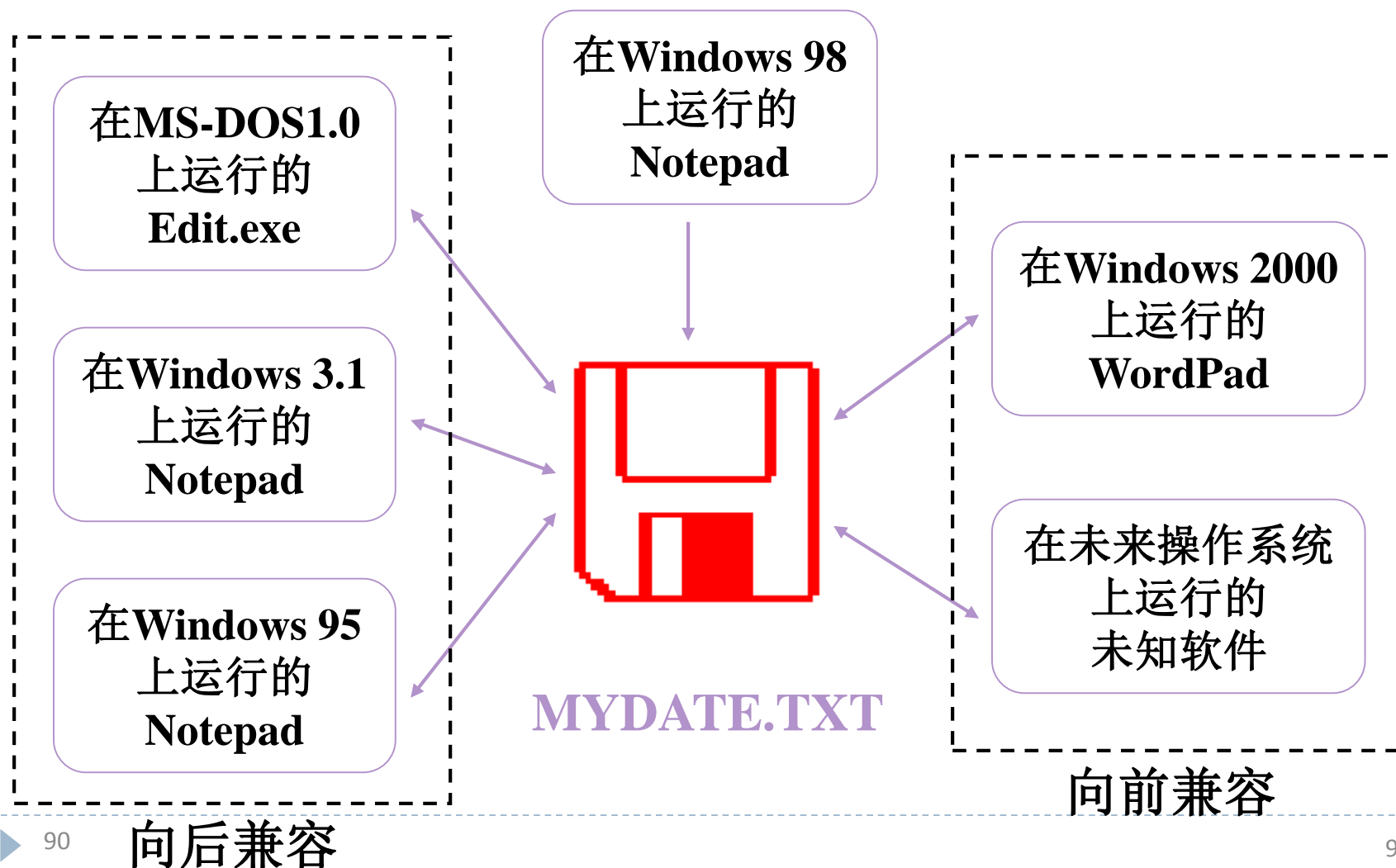
向后兼容是指可以使用软件的**以前版本**。

并非所有的软件都要求向前兼容和向后兼容，这是软件设计者需要决定的**产品特性**。

例：使用文本文件可以对向前兼容和向后兼容作一个简单的演示：在Windows 98上用Notepad创建的文本文件，它可以向后兼容MS-DOS 1.0后的所有版本，它还可以向前兼容Windows XP甚至以后的版本。



# 兼容性测试（续）



# 兼容性测试（续）

---

## （2）不同版本间的兼容

不同版本间的兼容是指多种平台和应用软件的多个版本之间是否能够正常工作。

如：测试一个应用软件对于操作系统的兼容性。可能涉及：

- 跨平台测试

- 同OS的不同版本的兼容性测试

- 同OS不同语言版的兼容性测试

- 不同厂家、相同类型的OS兼容性测试

再如：测试一个流行操作系统的新版本，当前版操作系统上可能有成千上万的程序，则新操作系统目标是与它们百分之百兼容。

因为不可能在一个操作系统上测试所有的软件程序，因此需要决定哪些是最重要的、必须进行的。

（续）

# 兼容性测试（续）

---

## （2）不同版本间的兼容（续）

对于测试新应用软件也一样，需要决定在哪个平台版本上测试，以及和什么应用程序一起测试。

选择“重要”的程序原则是：

流程度：利用销售记录选择前100或1000个最流行的程序。

年头：应该选择近3年内的程序和版本。

类型：把软件分为画图、书写、财务、数据库、通信等类型。从每一种类型中选择测试软件。

生产厂商：另一个原则是根据制作软件的公司来选择软件。

# 兼容性测试（续）

---

## (3) 标准和规范

适用于软件平台的标准和规范有两个级别——高级标准和低级标准。

### 高级标准

高级标准是产品应当普遍遵守的，如软件能在何种操作系统上运行？是互联网上的程序吗？它运行于何种浏览器？每一项问题都关系到平台，假若应用程序声明与某个平台兼容，就必须接受关于该平台的标准和规范。

如，Microsoft Windows的认证徽标，即WHQL(Windows Hardware. Quality Labs)。为了得到这个徽标，软件必须通过独立测试实验室的兼容性测试。其目的是确保软件在操作系统上能够平稳可靠地运行。

# Windows的认证徽标



On computers



On software and devices



# 兼容性测试（续）

---

## （3）标准和规范（续）

### 低级标准

低级标准是对产品开发细节的描述，如文件格式和网络通讯协议等。从某种意义上说，低级标准比高级标准更加重要。

如，一个图形软件，把文件保存为.pict文件格式（Macintosh标准图形文件格式），而程序不符合.pict文件的标准，用户就无法在其他程序中查看该文件。该软件与标准不兼容，很可能成为短命产品。

同样，通信协议、编程语言语法以及程序员用于共享信息的任何形式都必须符合公开标准和规范。

低级兼容性标准可以视为软件说明书的扩充部分。如果软件说明书有：“本软件以.bmp,jpg,gif格式读写图形文件”，就要找到这些格式的标准，并测试以确保软件符合标准。

# 兼容性测试（续）

---

## （4）数据共享兼容

数据共享兼容是指要在应用程序之间共享数据，它要求支持并遵守公开的标准，允许用户与其他软件无阻碍的传输数据。

如：

在Windows环境下，程序间通过剪切、复制和粘贴实现数据共享。在此状况下，传输通过剪贴板的程序来实现。若对某个程序进行兼容性测试就要确认其数据能够利用剪贴板与其他程序进行相互复制。

通过读写移动外存实现数据共享，如软磁盘、U盘、移动硬盘等，但文件的数据格式必须符合标准，才能在台计算机上保持兼容。

文件导入/导出的测试：是否正确转换为新格式。

